SQE Final Project

Project Title: Comprehensive Quality Engineering for the Open-Source Crater Application

Testers: Uzair Majeed 23i-3063, Hussnain Haider 23i-0695, Faez Ahmed 23i-0598

Section: SE-B

Unit Testing Report (IEEE 829-2008 Standard)

Index of Test Files

42.	CustomFieldValueResource-Test.txt
43.	CustomPathGenerator-Test.txt
44.	CustomerCollection-Test.txt
45.	CustomerEstimateStatusRequest-Test.txt
46.	CustomerLoginRequest-Test.txt
47.	CustomerMailResetPasswordNotification-Test.txt
48.	CustomerPolicy-Test.txt
49.	CustomerPortalMiddleware-Test.txt
50.	CustomerProfileRequest-Test.txt
51.	CustomerRequest-Test.txt
52.	CustomersController-Test.txt
53.	DashboardController-Test.txt
54.	DatabaseEnvironmentRequest-Test.txt
55.	DeleteExpensesRequest-Test.txt
56.	DeleteItemsRequest-Test.txt
57.	DiskEnvironmentRequest-Test.txt
58.	DomainEnvironmentRequest-Test.txt
59.	DownloadModuleController-Test.txt
60.	DropboxServiceProvider-Test.txt
61.	EmailLog-Test.txt
62.	EnableModuleController-Test.txt
63.	EnvironmentManager-Test.txt
64.	EstimateCollection-Test.txt
65.	EstimateItem-Test.txt
66.	EstimateItemCollection-Test.txt
67.	EstimateItemResource-Test.txt
68.	EstimatePdfController-Test.txt
69.	EstimateResource-Test.txt
70.	EstimateTemplatesController-Test.txt
71.	EstimateViewedMail-Test.txt
72.	EstimatesRequest-Test.txt
73.	EventServiceProvider-Test.txt
74.	ExchangeRateLog-Test.txt
75.	ExchangeRateLogCollectionAndResource-Test.txt
76.	ExchangeRateLogRequest-Test.txt
77.	ExchangeRateProvider-Test.txt
78.	ExchangeRateProviderCollection-Test.txt
79.	ExchangeRateProviderController-Test.txt
80.	ExchangeRateProviderRequest-Test.txt
81.	Expense-Test.txt
82.	ExpenseCategoriesController-Test.txt
83.	ExpenseCategory-Test.txt
84.	ExpenseCategoryCollection-Test.txt
85.	ExpenseCategoryPolicy-Test.txt

86.	ExpenseCategoryRequest-Test.txt
87.	ExpenseCategoryResource-Test.txt
88.	FileDisk-Test.txt
89.	FileDiskCollection-Test.txt
90.	FileDiskResource-Test.txt
91.	FilePermissionChecker-Test.txt
92.	FilesystemDisks-Test.txt
93.	FinishController-Test.txt
94.	FinishUpdateController-Test.txt
95.	ForgotPasswordController-Test.txt
96.	GenerateEstimatePdfJob-Test.txt
97.	GenerateInvoicePdfJob-Test.txt
98.	GeneratePaymentPdfJob-Test.txt
99.	GeneratesMenuTrait-Test.txt
100.	GetSettingRequest-Test.txt
101.	GetSettingsRequest-Test.txt
102.	GetSupportedCurrenciesController-Test.txt
103.	Handler-Test.txt
104.	InstallModuleCommand-Test.txt
105.	InstallationMiddleware-Test.txt
106.	Invoice-Test.txt
107.	InvoiceCollection-Test.txt
108.	InvoiceItem-Test.txt
109.	InvoicePdfController-Test.txt
110.	InvoiceViewedMail-Test.txt
111.	Invoicing-Test.txt
112.	ItemCollection-Test.txt
113.	ItemPolicy-Test.txt
114.	ItemResource-Test.txt
115.	ItemSalesReportController-Test.txt
116.	Kernel-Test.txt
117.	Listener-Test.txt
118.	LoginController-Test.txt
119.	LoginRequest-Test.txt
120.	MailEnvironmentRequest-Test.txt
121.	MailResetPasswordNotification-Test.txt
122.	MigrateUpdateController-Test.txt
123.	ModuleCollection-Test.txt
124.	ModuleDisabledEvent-Test.txt
125.	ModuleEnabledEvent-Test.txt
126.	ModuleFacade-Test.txt
127.	ModulesPolicy-Test.txt
128.	NextNumberController-Test.txt
129.	Note-Test.txt

130.	NumberPlaceholdersController-Test.txt
131.	OnboardingWizardController-Test.txt
132.	OwnerPolicy-Test.txt
133.	PathToZip-Test.txt
134.	PaymentCollection-Test.txt
135.	PaymentMethod-Test.txt
136.	PaymentMethods-Test.txt
137.	PdfMiddleware-Test.txt
138.	ProfileControllerAndRequest-Test.txt
139.	RecurringInvoice-Test.txt
140.	Redirect-Test.txt
141.	RegisterController-Test.txt
142.	RelationNotExist-Test.txt
143.	Request-Test.txt
144.	RequirementsController-Test.txt
145.	ResetApp-Test.txt
146.	ResetPasswordController-Test.txt
147.	RetrospectiveEditsController-Test.txt
148.	RoleCollection-Test.txt
149.	RolePolicy-Test.txt
150.	RouteServiceProvider-Test.txt
151.	SendInvoiceRequest-Test.txt
152.	SendMail-Test.txt
153.	SettingKeyRequest-Test.txt
154.	SettingRequest-Test.txt
155.	SettingsPolicy-Test.txt
156.	ShowReceiptController-Test.txt
157.	SiteApi-Test.txt
158.	TaxCollectionAndResource-Test.txt
159.	TestMail-Test.txt
160.	TimeZones-Test.txt
161.	Transaction-Test.txt
162.	TrimStrings-Test.txt
163.	TrustProxies-Test.txt
164.	Unit-Test.txt
165.	UnitPolicyReqAndResource-Test.txt
166.	UnzipModuleController-Test.txt
167.	UnzipUpdateController-Test.txt
168.	UnzipUpdateRequest-Test.txt
169.	UpdateController-Test.txt
170.	UpdateFinished-Test.txt
171.	UpdateSettingsRequest-Test.txt
172.	UpdateUserSettingsController-Test.txt
173.	UploadModuleRequest-Test.txt

174.	UploadReceiptRequest-Test.txt
175.	UserPolicy-Test.txt
176.	UserRelated-Test.txt
177.	VerificationController-Test.txt
178.	VerifyCsrfToken-Test.txt
179.	helpers-Test.txt

File: AbilitiesController-Test.txt

Test Case ID	AC-001
Title	AbilitiesController returns list of abilities from config
Objective	Verify that AbilitiesController returns the correct list of abilities configured in the application.
Preconditions	- Application running - Database seeded - Laravel config system available - Abilities are defined in config under 'abilities.abilities'
Test Steps	1. Set config key 'abilities.abilities' with the values: - 'users.view' => 'View Users' - 'users.create' => 'Create Users' - 'roles.manage' => 'Manage Roles' 2. Instantiate AbilitiesController. 3. Create a GET request to '/test-abilities'. 4. Call AbilitiesController with the created request and capture JSON response. 5. Wrap response with TestResponse helper. 6. Assert HTTP response is OK (200). 7. Assert response JSON matches: ['abilities' => mockAbilities]. 8. Assert response is instance of JsonResponse.
Test Data	- Config: 'abilities.abilities' set to array of three abilities - Request: GET, path '/test-abilities' - Expected JSON: {'abilities': ['users.view' => 'View Users', 'users.create' => 'Create Users', 'roles.manage' => 'Manage Roles']}
Expected Result	- HTTP response status is 200 OK. - Response JSON contains abilities as set in config. - Response is instance of JsonResponse.
Actual Result	HTTP response is 200 OK. JSON contains correct abilities data. Response is JsonResponse.
Status	Pass
Severity	High

Test Case ID	AC-002
Title	AbilitiesController returns empty abilities when config is empty
Objective	Verify that AbilitiesController returns an empty abilities array when no abilities are configured.
Preconditions	- Application running - Database seeded - Laravel config system available - 'abilities.abilities' config key is set to empty array
Test Steps	1. Set config key 'abilities.abilities' to empty array []. 2. Instantiate AbilitiesController. 3. Create a GET request to '/test-abilities'. 4. Call AbilitiesController with the created request and capture JSON response. 5. Wrap response with TestResponse helper. 6. Assert HTTP response is OK (200). 7. Assert response JSON matches: ['abilities' => []]. 8. Assert response is instance of JsonResponse.
Test Data	- Config: 'abilities.abilities' set to [] - Request: GET, path '/test-abilities' - Expected JSON: {'abilities': []}
Expected Result	- HTTP response status is 200 OK. - Response JSON contains empty abilities array. - Response is instance of JsonResponse.

Actual Result	HTTP response is 200 OK. JSON contains empty abilities array. Response is JsonResponse.
Status	Pass
Severity	High

Test Case ID	AC-003
Title	AbilitiesController returns identical abilities for different requests
Objective	Ensure AbilitiesController returns the same abilities regardless of differences in request objects.
Preconditions	- Application running - Database seeded - Laravel config system available - 'abilities.abilities' config key contains defined ability
Test Steps	1. Set config key 'abilities.abilities' to ['test.ability' => 'Test Ability']. 2. Instantiate AbilitiesController. 3. Create GET request to '/api/v1/admin/abilities' with query param 'param=value'. 4. Create POST request to '/another/path' with post data 'data=different' and header 'HTTP_ACCEPT: application/xml'. 5. Call AbilitiesController with each request and capture JSON response. 6. Wrap both responses with TestResponse helper. 7. Assert HTTP response is OK (200) for both requests. 8. Assert response JSON matches: ['abilities' => ['test.ability' => 'Test Ability']] for both. 9. Compare returned abilities from both responses and verify they are identical.
Test Data	- Config: 'abilities.abilities' set to ['test.ability' => 'Test Ability'] - Requests: -- GET '/api/v1/admin/abilities', param 'param=value' -- POST '/another/path', post data 'data=different', header 'HTTP_ACCEPT: application/xml' - Expected JSON: {'abilities': ['test.ability' => 'Test Ability']}
Expected Result	- Both HTTP responses are 200 OK. - Both response JSON contain the same abilities data. - Returned abilities are identical regardless of request method, path, or headers.
Actual Result	HTTP response on both requests is 200 OK. Both contain identical abilities data as set in config.
Status	Pass
Severity	High

File: AbilityCollection-Test.txt

Test Case ID	AC-001
Title	AbilityCollection extends ResourceCollection
Objective	Verify that the AbilityCollection class extends the ResourceCollection class as expected.
Preconditions	- Application running - Required classes available: AbilityCollection, Collection, ResourceCollection - No data in database required
Test Steps	1. Instantiate an empty Collection object. 2. Create an AbilityCollection object using the empty Collection. 3. Verify that the AbilityCollection object is an instance of ResourceCollection.
Test Data	- Input Collection: Empty Collection object - AbilityCollection created from empty Collection
Expected Result	- AbilityCollection object is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	AbilityCollection object was successfully recognized as an instance of ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	AC-002
Title	AbilityCollection toArray returns empty array for empty collection
Objective	Ensure that the toArray method of AbilityCollection returns an empty array when initialized with an empty collection.
Preconditions	- Application running - Required classes available: AbilityCollection, Collection, ResourceCollection - No data in database required
Test Steps	1. Instantiate an empty Collection object. 2. Instantiate a Request object. 3. Create an AbilityCollection object using the empty Collection. 4. Invoke the toArray method on the AbilityCollection object, passing in the Request object. 5. Verify that the result is an empty array.
Test Data	- Input Collection: Empty Collection object - Request: Default Request object
Expected Result	- The result of toArray is an array. - The result array is empty.
Actual Result	The toArray method returned an empty array as expected.
Status	Pass
Severity	Low

File: AcceptEstimateController-Test.txt

Test Case ID	AEC-001
Title	Return 404 when estimate is not found or doesn't belong to authenticated customer
Objective	Verify that the controller returns 404 and appropriate error response when the estimate is either missing or does not belong to the logged-in customer.
Preconditions	- Application server is running - Database seeded, but estimate with ID 99 does not exist or is not linked to customer ID 1 - Customer authentication available (customer ID = 1) - Company and estimate model accessible
Test Steps	1. Authenticate as a customer with ID 1. 2. Call AcceptEstimateController's __invoke method, passing mocked request (status: accepted), a mocked company, and estimate ID 99. 3. Simulate the company’s estimates query such that estimate ID 99 cannot be found for this customer.
Test Data	- customerId: 1 - estimateId: 99 (non-existent or inaccessible) - request payload: { status: "accepted" }
Expected Result	- Response status code is 404. - Response JSON contains: { "error": "estimate_not_found" }
Actual Result	- Response status code is 404. - Response JSON contains: { "error": "estimate_not_found" }
Status	Pass
Severity	High

Test Case ID	AEC-002
Title	Successfully accept an estimate and return updated resource
Objective	Verify that the controller sets the estimate status to "accepted" and returns an EstimateResource when the request is valid.
Preconditions	- Application server is running - Database seeded: Estimate with ID 123 exists, status is "pending", and belongs to customer ID 1 - Customer authentication available (customer ID = 1) - Company and estimate model accessible
Test Steps	1. Authenticate as customer with ID 1. 2. Call AcceptEstimateController's __invoke method, passing mocked request (status: accepted), mocked company, and estimate ID 123. 3. Simulate the company’s estimates query to return the correct estimate. 4. Update the estimate status from "pending" to "accepted".
Test Data	- customerId: 1 - estimateId: 123 - request payload: { status: "accepted" }
Expected Result	- An instance of EstimateResource is returned. - The estimate's status is changed to "accepted".
Actual Result	- An instance of EstimateResource is returned. - The estimate's status is "accepted".
Status	Pass
Severity	High

Test Case ID	AEC-003
Title	Successfully decline an estimate and return updated resource

Objective	Verify that the controller sets the estimate status to "declined" and returns an EstimateResource when the decline action is requested.
Preconditions	- Application server is running - Database seeded: Estimate with ID 124 exists, status is "pending", and belongs to customer ID 1 - Customer authentication available (customer ID = 1) - Company and estimate model accessible
Test Steps	1. Authenticate as customer with ID 1. 2. Call AcceptEstimateController's __invoke method, passing mocked request (status: declined), mocked company, and estimate ID 124. 3. Simulate the company’s estimates query to return the correct estimate. 4. Update the estimate status from "pending" to "declined".
Test Data	- customerId: 1 - estimateId: 124 - request payload: { status: "declined" }
Expected Result	- An instance of EstimateResource is returned. - The estimate's status is changed to "declined".
Actual Result	- An instance of EstimateResource is returned. - The estimate's status is "declined".
Status	Pass
Severity	High

Test Case ID	AEC-004
Title	Handle request with no status field and maintain estimate status
Objective	Verify that if the request does not contain a status field, the estimate status remains unchanged and an EstimateResource is returned.
Preconditions	- Application server is running - Database seeded: Estimate with ID 125 exists, status is "pending", and belongs to customer ID 1 - Customer authentication available (customer ID = 1) - Company and estimate model accessible
Test Steps	1. Authenticate as customer with ID 1. 2. Call AcceptEstimateController's __invoke method, passing mocked request (no status field), mocked company, and estimate ID 125. 3. Simulate the company’s estimates query to return the correct estimate. 4. Try to update the estimate with an empty data array. 5. Verify that the estimate’s status remains "pending".
Test Data	- customerId: 1 - estimateId: 125 - request payload: { } (no status field)
Expected Result	- An instance of EstimateResource is returned. - The estimate's status remains unchanged as "pending".
Actual Result	- An instance of EstimateResource is returned. - The estimate's status is "pending".
Status	Pass
Severity	High

File: ApiController-Test.txt

Test Case ID	AC-001
Title	RespondSuccess returns JSON response with success=true
Objective	Verify that the respondSuccess method returns a JsonResponse object with the "success" property set to true and HTTP status code 200.
Preconditions	- Application running - Database seeded - Mockery library available for mocking - Instance of ResponseFactory bound to mocked object - Controller \Crater\Http\Controllers\V1\Admin\Backup\ApiController is available
Test Steps	1. Create a mock JsonResponse object where getData returns an object with "success" => true, and getStatusCode returns 200. 2. Create a mock ResponseFactory object; set its json() method to expect data with "success" => true and return the mock JsonResponse. 3. Bind the mock ResponseFactory to the Laravel service container. 4. Instantiate the ApiController. 5. Call the respondSuccess() method on the controller and capture the response. 6. Assert that the response is the mock JsonResponse object. 7. Assert that the response data has property "success" set to true. 8. Assert that the response status code is 200.
Test Data	- Input: Call respondSuccess() method (no parameters required) - Expected Output: JsonResponse object with data = {"success": true}, status code = 200
Expected Result	- The returned object is the mocked JsonResponse. - The response data contains "success" property set to true. - The response status code is 200.
Actual Result	- The returned object is the mocked JsonResponse. - The response data contains "success" property set to true. - The response status code is 200.
Status	Pass
Severity	High

File: ApiTokenController-Test.txt

Test Case ID	ATC-001
Title	Authorize and return successful module installer response with valid API token
Objective	Verify that the controller authorizes the request and returns a successful response when a valid API token is provided.
Preconditions	- Application running - Database seeded - Valid API token ("valid-token-123") exists in the system
Test Steps	1. Create a mock request with api_token set to "valid-token-123". 2. Mock ModuleInstaller to expect "valid-token-123" and return a successful response. 3. Mock ApiTokenController to authorize "manage modules". 4. Invoke the controller with the mock request. 5. Check that the response matches the expected successful payload. 6. Confirm the response status code is 200. 7. Verify the returned data matches: ['success' => true, 'message' => 'Token verified']
Test Data	- Input: api_token = "valid-token-123" - Expected values: - Payload: ['success' => true, 'message' => 'Token verified'] - Status Code: 200
Expected Result	- The controller returns a JSON response with payload ['success' => true, 'message' => 'Token verified'] and HTTP status 200. - The authorize method for "manage modules" is called.
Actual Result	The controller returned the expected JSON response with 'success' true and status code 200 as authorized.
Status	Pass
Severity	High

Test Case ID	ATC-002
Title	Authorize and return failure module installer response with invalid API token
Objective	Verify that the controller authorizes the request and returns a failure response when an invalid API token is provided.
Preconditions	- Application running - Database seeded - Invalid API token ("invalid-token-xyz") does not exist in the system
Test Steps	1. Create a mock request with api_token set to "invalid-token-xyz". 2. Mock ModuleInstaller to expect "invalid-token-xyz" and return a failure response. 3. Mock ApiTokenController to authorize "manage modules". 4. Invoke the controller with the mock request. 5. Check that the response matches the expected failure payload. 6. Confirm the response status code is 401. 7. Verify the returned data matches: ['success' => false, 'message' => 'Invalid token']
Test Data	- Input: api_token = "invalid-token-xyz" - Expected values: - Payload: ['success' => false, 'message' => 'Invalid token'] - Status Code: 401
Expected Result	- The controller returns a JSON response with payload ['success' => false, 'message' => 'Invalid token'] and HTTP status 401. - The authorize method for "manage modules" is called.
Actual Result	The controller returned the expected JSON response with 'success' false and status code 401 as authorized.
Status	Pass

Severity	High

Test Case ID	ATC-003
Title	Return error when API token is missing from request
Objective	Verify that the controller returns a 400 error when the API token is not provided in the request.
Preconditions	- Application running - Database seeded
Test Steps	1. Create a mock request without api_token (set to null). 2. Mock ModuleInstaller to expect null token and return an error response. 3. Mock ApiTokenController to authorize "manage modules". 4. Invoke the controller with the mock request. 5. Check that the response matches the expected error payload. 6. Confirm the response status code is 400. 7. Verify the returned data matches: ['success' => false, 'message' => 'API token is required']
Test Data	- Input: api_token = null - Expected values: - Payload: ['success' => false, 'message' => 'API token is required'] - Status Code: 400
Expected Result	- The controller returns a JSON response with payload ['success' => false, 'message' => 'API token is required'] and HTTP status 400. - The authorize method for "manage modules" is called.
Actual Result	The controller returned the expected JSON response with 'success' false and status code 400 for missing API token.
Status	Pass
Severity	High

Test Case ID	ATC-004
Title	Return error when API token is empty string in request
Objective	Verify that the controller returns a 400 error when the API token is an empty string in the request.
Preconditions	- Application running - Database seeded
Test Steps	1. Create a mock request with api_token set to an empty string (""). 2. Mock ModuleInstaller to expect an empty string token and return an error response. 3. Mock ApiTokenController to authorize "manage modules". 4. Invoke the controller with the mock request. 5. Check that the response matches the expected error payload. 6. Confirm the response status code is 400. 7. Verify the returned data matches: ['success' => false, 'message' => 'API token cannot be empty']
Test Data	- Input: api_token = "" - Expected values: - Payload: ['success' => false, 'message' => 'API token cannot be empty'] - Status Code: 400
Expected Result	- The controller returns a JSON response with payload ['success' => false, 'message' => 'API token cannot be empty'] and HTTP status 400. - The authorize method for "manage modules" is called.
Actual Result	The controller returned the expected JSON response with 'success' false and status code 400 for empty API token.
Status	Pass
Severity	High

Test Case ID	ATC-005
Title	Authorize method is invoked with correct capability
Objective	Ensure that the controller's authorize method is called with the correct capability ("manage modules").
Preconditions	- Application running - Database seeded
Test Steps	1. Create a mock request with api_token set to "any-token". 2. Mock ModuleInstaller to expect "any-token" and return a successful response. 3. Mock ApiTokenController to expect authorize called with "manage modules". 4. Invoke the controller with the mock request. 5. Check that the response is an instance of Response.
Test Data	- Input: api_token = "any-token" - Expected values: - The authorize method is called exactly once with capability "manage modules" - Controller returns a Response object
Expected Result	- The authorize method is called once with parameter "manage modules". - The controller returns a Response instance.
Actual Result	The authorize method was called exactly once with the correct capability and the controller returned a Response instance as expected.
Status	Pass
Severity	Medium

File: AppDomainController-Test.txt

Test Case ID	ADC-001
Title	Successful saving of domain environment variables returns success response
Objective	To verify that when environment variables for a domain are saved successfully, the optimize cache is cleared and a success response is returned.
Preconditions	- Application running - Database seeded - Valid DomainEnvironmentRequest provided - Artisan facade available for mocking - EnvironmentManager can be overloaded and mocked
Test Steps	1. Mock the Artisan facade to expect a single call to 'optimize:clear' and return 0 (exit code success). 2. Mock the DomainEnvironmentRequest instance. 3. Overload and mock the EnvironmentManager class. 4. Set EnvironmentManager to expect a constructor call, then its saveDomainVariables method called once with the mocked request, returning ['status' => 'success', 'message' => 'Variables saved successfully']. 5. Instantiate AppDomainController and invoke __invoke() method with the mocked request. 6. Verify the response is an instance of JsonResponse. 7. Check the HTTP status code is within the 2xx range (200 <= code < 300). 8. Assert that the JSON response data equals ['success' => true].
Test Data	- Input: Mocked DomainEnvironmentRequest - saveDomainVariables return: ['status' => 'success', 'message' => 'Variables saved successfully'] - Expected JsonResponse payload: ['success' => true] - HTTP status code: 2xx
Expected Result	- The optimize cache is cleared (Artisan::call('optimize:clear') returns 0). - A JsonResponse object with HTTP status code in the 200-299 range is returned. - The response payload is exactly ['success' => true].
Actual Result	- The optimize cache was cleared. - A JsonResponse was returned with status code 200. - The response payload was ['success' => true].
Status	Pass
Severity	High

Test Case ID	ADC-002
Title	Failure in saving domain environment variables returns error response
Objective	To verify that when saving environment variables fails, the optimize cache is still cleared but an error response is returned.
Preconditions	- Application running - Database seeded - Valid DomainEnvironmentRequest provided - Artisan facade available for mocking - EnvironmentManager can be overloaded and mocked

Test Steps	1. Mock the Artisan facade to expect a single call to 'optimize:clear' and return 0 (exit code success). 2. Mock the DomainEnvironmentRequest instance. 3. Overload and mock the EnvironmentManager class. 4. Set EnvironmentManager to expect a constructor call, then its saveDomainVariables method invoked once with the mocked request, returning ['message' => 'Failed to write to .env file', 'error', 'code' => 500]. 5. Instantiate AppDomainController and call __invoke() method with the mocked request. 6. Verify the response is an instance of JsonResponse. 7. Assert that the HTTP status code is 200. 8. Assert that the JSON response data equals ['message' => 'Failed to write to .env file', 'error', 'code' => 500].
Test Data	- Input: Mocked DomainEnvironmentRequest - saveDomainVariables return: ['message' => 'Failed to write to .env file', 'error', 'code' => 500] - Expected JsonResponse payload: ['message' => 'Failed to write to .env file', 'error', 'code' => 500] - HTTP status code: 200
Expected Result	- The optimize cache is cleared (Artisan::call('optimize:clear') returns 0). - A JsonResponse object with HTTP status code 200 is returned. - The response payload is exactly ['message' => 'Failed to write to .env file', 'error', 'code' => 500].
Actual Result	- The optimize cache was cleared. - A JsonResponse was returned with status code 200. - The response payload was ['message' => 'Failed to write to .env file', 'error', 'code' => 500].
Status	Pass
Severity	High

File: AppServiceProvider-Test.txt

Test Case ID	ASP-001
Title	Verify register method exists and executes without errors
Objective	Ensure the AppServiceProvider::register method exists and can be called without throwing any exceptions.
Preconditions	- Application running - All necessary dependencies are installed - Database is seeded - AppServiceProvider class is available
Test Steps	1. Instantiate the AppServiceProvider with the Laravel application instance. 2. Call the register() method on the provider. 3. Verify that no exceptions are thrown during this process.
Test Data	- Laravel application instance - AppServiceProvider class
Expected Result	- The register() method exists and executes without throwing any exceptions.
Actual Result	The register() method executed successfully without any errors.
Status	Pass
Severity	Medium

Test Case ID	ASP-002
Title	Validate boot method under all storage and schema conditions
Objective	Ensure the AppServiceProvider::boot method runs without errors regardless of database_created file presence and abilities table existence.
Preconditions	- Application running - Database seeded - AppServiceProvider class available - Storage disk 'local' accessible - No conflicting database migrations
Test Steps	1. For each of the following combinations: a. database_created file exists, abilities table exists. b. database_created file exists, abilities table does not exist. c. database_created file does not exist, abilities table exists. d. database_created file does not exist, abilities table does not exist. 2. Set up the storage disk and abilities table as per combination. 3. Instantiate AppServiceProvider. 4. Call the boot() method. 5. Observe if any errors or exceptions are thrown.
Test Data	- hasDatabaseCreated: true/false - hasAbilitiesTable: true/false
Expected Result	- In all combinations, boot() method executes successfully without throwing any exceptions.
Actual Result	The boot() method ran without errors for all tested conditions.
Status	Pass
Severity	High

Test Case ID	ASP-003
Title	Ensure menu configs are properly structured and accessible
Objective	Verify that menu configs (main_menu, setting_menu, customer_menu) are arrays and have the expected count or are empty when set.
Preconditions	- Application running - Config facade accessible - AppServiceProvider loaded

Test Steps	1. Set main_menu config to an array with one menu item. 2. Assert config('crater.main_menu') is an array and has one item. 3. Set setting_menu config to an array with two items. 4. Assert config('crater.setting_menu') is an array and has two items. 5. Set customer_menu config to an empty array. 6. Assert config('crater.customer_menu') is an array and is empty.
Test Data	- Main menu: 1 item (title: 'Test', link: '/test', ...) - Setting menu: 2 items (same structure) - Customer menu: empty array
Expected Result	- crater.main_menu is array and has one item. - crater.setting_menu is array and has two items. - crater.customer_menu is array and is empty.
Actual Result	All menu configs are arrays with the expected structure and counts.
Status	Pass
Severity	Medium

Test Case ID	ASP-004
Title	Check provider dependencies (facades) are available
Objective	Confirm that all required Laravel facades (Storage, Schema, Config, Menu) are available and can be loaded.
Preconditions	- Application running - Laravel framework and all facades installed - AppServiceProvider loaded
Test Steps	1. Verify that the Storage facade class exists. 2. Verify that the Schema facade class exists. 3. Verify that the Config facade class exists. 4. Confirm that the Menu facade would be available (mocked in test).
Test Data	- Classes: \Illuminate\Support\Facades\Storage, Schema, Config, Menu
Expected Result	- All listed facades exist and are available for use.
Actual Result	All provider dependencies (facades) were successfully detected.
Status	Pass
Severity	High

Test Case ID	ASP-005
Title	Validate storage disk "local" functionality
Objective	Ensure the "local" storage disk is available and supports read/write operations.
Preconditions	- Application running - Storage facade accessible - "local" disk configured - Sufficient filesystem permissions
Test Steps	1. Obtain "local" storage disk via Storage::disk('local'). 2. Write the file "test.txt" with content "content" to the disk. 3. Assert that "test.txt" exists on the disk. 4. Read "test.txt" and assert its content is "content".
Test Data	- File name: test.txt - File content: content
Expected Result	- Storage::disk('local') returns an object. - File "test.txt" exists after writing. - File content read matches "content".
Actual Result	Storage disk "local" operated as expected: file was created, detected, and read successfully.
Status	Pass
Severity	High

Test Case ID	ASP-006
Title	Verify Schema facade table existence and operations
Objective	Ensure Schema facade correctly checks for table existence, and allow table creation and removal.
Preconditions	- Application running - Database connection available - Schema facade accessible
Test Steps	1. Check existence of a randomly named table (should not exist). 2. Create a table named "test_table" if not exists. 3. Assert that "test_table" exists. 4. Drop the "test_table".
Test Data	- Table name: "non_existent_table_{random}" - Table name: "test_table"
Expected Result	- The random table does not exist. - "test_table" exists after creation. - "test_table" is dropped at the end of the test.
Actual Result	Schema operations succeeded: new table created, detected, and dropped properly.
Status	Pass
Severity	Medium

Test Case ID	ASP-007
Title	Ensure addMenus method executes when required conditions met
Objective	Confirm that the addMenus method can be invoked and runs without errors when database_created file and abilities table exist.
Preconditions	- Application running - Storage facade accessible - "local" disk with database_created file set - Abilities table exists in database
Test Steps	1. Put "database_created" file with sample content on local storage disk. 2. Create "abilities" table if not exists (with id, name, timestamps). 3. Instantiate AppServiceProvider. 4. Use reflection to access and invoke the addMenus method. 5. Assert no exceptions occur during execution.
Test Data	- Storage file: database_created = content - Database table: abilities
Expected Result	- addMenus method is invoked and completes without errors.
Actual Result	addMenus method executed successfully under required conditions.
Status	Pass
Severity	High

Test Case ID	ASP-008
Title	Validate generateMenu method adds menu items correctly
Objective	Ensure the generateMenu method properly adds menu data to a menu object.
Preconditions	- Application running - AppServiceProvider loaded - Menu object structure available for mocking

Test Steps	1. Create an AppServiceProvider instance. 2. Mock a menu object with add and data methods. 3. Prepare menuData array with valid fields. 4. Use reflection to invoke generateMenu with mock menu and menuData. 5. Assert that a new menu item is added. 6. Assert title and link in the new item match menuData.
Test Data	- menuData: title: "Test Menu", link: "/test", icon: "icon-test", name: "test_menu", owner_only: true, ability: "manage_test", model: "TestModel", group: "test"
Expected Result	- mockMenu->addedItems contains one item. - added item's title is "Test Menu". - added item's link is "/test".
Actual Result	generateMenu method added the menu item exactly as expected.
Status	Pass
Severity	Medium

Test Case ID	ASP-009
Title	Confirm JSON translations are loaded from correct resource path
Objective	Verify the JSON translations path is constructed correctly and would load translations if invoked.
Preconditions	- Application running - Resource directory available - AppServiceProvider loaded
Test Steps	1. Instantiate AppServiceProvider. 2. Retrieve resource path for 'scripts/locales'. 3. Assert resource path is a valid string. 4. (Not directly tested: Would invoke loadJsonTranslationsFrom with this path.)
Test Data	- resource_path: scripts/locales
Expected Result	- resource_path('scripts/locales') returns a valid string. - (By implication) AppServiceProvider would load translations from this path.
Actual Result	JSON translations resource path was retrieved and verified as a valid string.
Status	Pass
Severity	Low

File: AuthController-Test.txt

Test Case ID	AC-001
Title	Login with incorrect credentials throws validation exception when user not found
Objective	Verify that logging in with a non-existent username returns a validation exception indicating invalid data.
Preconditions	- Application running - Database seeded (no user exists with "nonexistent@example.com") - User model and authentication setup configured
Test Steps	1. Submit a login request with username "nonexistent@example.com" and password "password". 2. Invoke the login method of the AuthController with this request.
Test Data	- Username: "nonexistent@example.com" - Password: "password"
Expected Result	- A ValidationException is thrown with message "The given data was invalid."
Actual Result	A ValidationException is correctly thrown with message "The given data was invalid."
Status	Pass
Severity	High

Test Case ID	AC-002
Title	Logout successfully deletes current access token and returns success
Objective	Verify that the logout process deletes the user's current access token and returns a success response.
Preconditions	- Application running - A user is authenticated and has an active access token - Request object properly mocked to return the user
Test Steps	1. Mock a user object with an active access token. 2. Mock the access token to expect a delete operation. 3. Mock a request to return the mocked user. 4. Call the logout method of the AuthController with the mocked request. 5. Check the response status code. 6. Check the response data for success indicator.
Test Data	- Mocked user and access token
Expected Result	- The current access token is deleted exactly once. - The response status code is 200. - The response data is {'success': true}.
Actual Result	Access token deleted. Response status code is 200. Response data is {'success': true}.
Status	Pass
Severity	High

Test Case ID	AC-003
Title	Check returns true when user is authenticated
Objective	Verify that the check method returns true when the user is authenticated.
Preconditions	- Application running - User successfully authenticated in the application - Authentication facade properly mocked to simulate authentication
Test Steps	1. Mock the Auth facade to return true for authentication check. 2. Call the check method of the AuthController. 3. Assert that the result is true.

Test Data	- Authentication state: true
Expected Result	- The returned value is true.
Actual Result	The returned value is true.
Status	Pass
Severity	High

Test Case ID	AC-004
Title	Check returns false when user is not authenticated
Objective	Verify that the check method returns false when the user is not authenticated.
Preconditions	- Application running - No user authenticated in the application - Authentication facade properly mocked to simulate unauthenticated state
Test Steps	1. Mock the Auth facade to return false for authentication check. 2. Call the check method of the AuthController. 3. Assert that the result is false.
Test Data	- Authentication state: false
Expected Result	- The returned value is false.
Actual Result	The returned value is false.
Status	Pass
Severity	High

File: AuthServiceProvider-Test.txt

Test Case ID	ASP-001
Title	Auth Service Provider processes without errors
Objective	Verify that the AuthServiceProvider can be registered and booted without throwing any exceptions.
Preconditions	- Application running - Database seeded - AuthServiceProvider dependency present
Test Steps	1. Instantiate the AuthServiceProvider with the application instance. 2. Execute the register() method of AuthServiceProvider. 3. Execute the boot() method of AuthServiceProvider.
Test Data	- AuthServiceProvider object instantiated with current app instance.
Expected Result	- No exceptions are thrown during registration and booting of AuthServiceProvider. - The test completes successfully.
Actual Result	Test completed with no exceptions raised; registration and boot succeeded.
Status	Pass
Severity	High

Test Case ID	ASP-002
Title	Provider instantiation type validation
Objective	Ensure that the AuthServiceProvider object is correctly instantiated and its type matches both AuthServiceProvider and ServiceProvider.
Preconditions	- Application running - Database seeded - AuthServiceProvider dependency present
Test Steps	1. Instantiate AuthServiceProvider with the current application instance. 2. Validate that the instantiated provider is an instance of AuthServiceProvider. 3. Validate that the provider is also an instance of Illuminate\Support\ServiceProvider.
Test Data	- AuthServiceProvider object instantiated with current app instance.
Expected Result	- The provider object is instance of AuthServiceProvider. - The provider object is also instance of Illuminate\Support\ServiceProvider.
Actual Result	Provider was verified to be instance of both AuthServiceProvider and ServiceProvider.
Status	Pass
Severity	High

Test Case ID	ASP-003
Title	Boot method defines required abilities
Objective	Confirm that calling boot() on AuthServiceProvider correctly defines authorization abilities in the Gate.
Preconditions	- Application running - Database seeded - AuthServiceProvider dependency present
Test Steps	1. Instantiate AuthServiceProvider with the current application instance. 2. Call the boot() method of AuthServiceProvider. 3. Check that the 'view dashboard' ability is defined in Gate. 4. Check that the 'manage settings' ability is defined in Gate. 5. Check that the 'create company' ability is defined in Gate.
Test Data	- Boot method call on AuthServiceProvider.

Expected Result	- Gate::has('view dashboard') returns true. - Gate::has('manage settings') returns true. - Gate::has('create company') returns true.
Actual Result	All specified abilities were present as expected in Gate.
Status	Pass
Severity	High

Test Case ID	ASP-004
Title	Provider instantiation basic type check
Objective	Verify that the AuthServiceProvider can be successfully instantiated.
Preconditions	- Application running - Database seeded - AuthServiceProvider dependency present
Test Steps	1. Instantiate AuthServiceProvider with the current application instance. 2. Verify that the object is an instance of AuthServiceProvider.
Test Data	- AuthServiceProvider object instantiated with current app instance.
Expected Result	- Provider is instance of AuthServiceProvider.
Actual Result	Provider was verified to be instance of AuthServiceProvider.
Status	Pass
Severity	High

File: AvatarRequest-Test.txt

Test Case ID	AR-001
Title	AvatarRequest extends Illuminate FormRequest
Objective	Verify that the AvatarRequest class properly extends the Illuminate\Foundation\Http\FormRequest class.
Preconditions	- Application running - Autoloader properly configured - AvatarRequest class available
Test Steps	1. Instantiate a new object of the AvatarRequest class. 2. Check if the instantiated object is an instance of Illuminate\Foundation\Http\FormRequest.
Test Data	- Instantiation of: AvatarRequest() - Expected value: Instance of Illuminate\Foundation\Http\FormRequest
Expected Result	- The AvatarRequest object is confirmed to be an instance of Illuminate\Foundation\Http\FormRequest.
Actual Result	AvatarRequest was successfully identified as an instance of Illuminate\Foundation\Http\FormRequest.
Status	Pass
Severity	Medium

Test Case ID	AR-002
Title	AvatarRequest authorize method returns true
Objective	Verify that the authorize() method in AvatarRequest always returns true.
Preconditions	- Application running - AvatarRequest class available
Test Steps	1. Instantiate a new object of the AvatarRequest class. 2. Call the authorize() method on the instantiated AvatarRequest object. 3. Verify the value returned by authorize().
Test Data	- Instantiation of: AvatarRequest() - Method call: authorize() - Expected value: true
Expected Result	- The authorize() method of AvatarRequest returns true.
Actual Result	The authorize() method returned true as expected.
Status	Pass
Severity	High

File: BackupDisk-Test.txt

Test Case ID	BD-001
Title	Instantiation of BackupDisk validation rule
Objective	Verify that a BackupDisk object can be created and implements the Rule interface.
Preconditions	- Application running - Required PHP dependencies installed - Database not required for this test
Test Steps	1. Create a new BackupDisk object. 2. Check if the object is an instance of BackupDisk. 3. Check if the object implements the Rule interface.
Test Data	- No specific input data.
Expected Result	- The object is an instance of BackupDisk. - The object implements the Rule interface.
Actual Result	- The object is correctly instantiated as BackupDisk and implements Rule.
Status	Pass
Severity	Medium

Test Case ID	BD-002
Title	Validation error message returned by BackupDisk
Objective	Ensure that the BackupDisk rule returns the correct validation error message.
Preconditions	- Application running - Required PHP dependencies installed
Test Steps	1. Create a new BackupDisk object. 2. Retrieve the validation error message via the message() method.
Test Data	- No specific input data.
Expected Result	- The returned message is "This disk is not configured as a backup disk."
Actual Result	- The returned message is "This disk is not configured as a backup disk."
Status	Pass
Severity	Medium

Test Case ID	BD-003
Title	Passing validation when disk is configured
Objective	Verify the BackupDisk rule passes when the disk value is among configured backup disks.
Preconditions	- Application running - Configuration set: backup.backup.destination.disks = ['local', 's3', 'ftp']
Test Steps	1. Set configuration backup.backup.destination.disks to ['local', 's3', 'ftp']. 2. Create a new BackupDisk object. 3. Validate 'local' as disk value. 4. Validate 's3' as disk value. 5. Validate 'ftp' as disk value.
Test Data	- Configured disks: ['local', 's3', 'ftp'] - Input disk values: 'local', 's3', 'ftp'
Expected Result	- Validation passes for 'local'. - Validation passes for 's3'. - Validation passes for 'ftp'.
Actual Result	- Validation passed for all tested values.
Status	Pass

Severity	High

Test Case ID	BD-004
Title	Failing validation when disk is not configured
Objective	Verify the BackupDisk rule fails when the disk value is not among configured backup disks.
Preconditions	- Application running - Configuration set: backup.backup.destination.disks = ['local', 's3']
Test Steps	1. Set configuration backup.backup.destination.disks to ['local', 's3']. 2. Create a new BackupDisk object. 3. Validate 'ftp' as disk value. 4. Validate 'dropbox' as disk value. 5. Validate 'unknown_disk' as disk value.
Test Data	- Configured disks: ['local', 's3'] - Input disk values: 'ftp', 'dropbox', 'unknown_disk'
Expected Result	- Validation fails for 'ftp'. - Validation fails for 'dropbox'. - Validation fails for 'unknown_disk'.
Actual Result	- Validation failed for all tested values.
Status	Pass
Severity	High

Test Case ID	BD-005
Title	Failing validation when no backup disks are configured
Objective	Ensure the BackupDisk rule fails for any disk value when no backup disks are configured.
Preconditions	- Application running - Configuration set: backup.backup.destination.disks = []
Test Steps	1. Set configuration backup.backup.destination.disks to an empty array. 2. Create a new BackupDisk object. 3. Validate 'local' as disk value. 4. Validate 's3' as disk value. 5. Validate 'any_disk' as disk value.
Test Data	- Configured disks: [] - Input disk values: 'local', 's3', 'any_disk'
Expected Result	- Validation fails for 'local'. - Validation fails for 's3'. - Validation fails for 'any_disk'.
Actual Result	- Validation failed for all tested values.
Status	Pass
Severity	High

Test Case ID	BD-006
Title	Case sensitivity handling in backup disk validation
Objective	Verify that BackupDisk rule handles disk value case sensitivity according to PHP in_array default behavior.
Preconditions	- Application running - Configuration set: backup.backup.destination.disks = ['local', 's3', 'FTP']

Test Steps	1. Set configuration backup.backup.destination.disks to ['local', 's3', 'FTP']. 2. Create a new BackupDisk object. 3. Validate 's3' as disk value (lowercase). 4. Validate 'S3' as disk value (uppercase). 5. Validate 'FTP' as disk value (uppercase). 6. Validate 'ftp' as disk value (lowercase).
Test Data	- Configured disks: ['local', 's3', 'FTP'] - Input disk values: 's3', 'S3', 'FTP', 'ftp'
Expected Result	- Validation passes for 's3'. - Validation fails for 'S3'. - Validation passes for 'FTP'. - Validation fails for 'ftp'.
Actual Result	- Validation passed for 's3' and 'FTP'; failed for 'S3' and 'ftp'.
Status	Pass
Severity	High

Test Case ID	BD-007
Title	Validation of null and empty string disk values
Objective	Verify how BackupDisk rule validates null and empty string disk values in different configurations.
Preconditions	- Application running - Configuration set: backup.backup.destination.disks = ['local', 's3', ''] - Later updated to: ['local', 's3']
Test Steps	1. Set configuration backup.backup.destination.disks to ['local', 's3', '']. 2. Create a new BackupDisk object. 3. Validate '' (empty string) as disk value. 4. Validate null as disk value. 5. Set configuration backup.backup.destination.disks to ['local', 's3'] (without empty string). 6. Validate '' (empty string) as disk value. 7. Validate null as disk value.
Test Data	- Configured disks: ['local', 's3', ''] and ['local', 's3'] - Input disk values: '', null
Expected Result	- Validation passes for '' and null when '' is configured. - Validation fails for '' and null when '' is NOT configured.
Actual Result	- Validation passed for '' and null when '' present; failed when '' not present.
Status	Pass
Severity	High

Test Case ID	BD-008
Title	BackupDisk validation with type conversions in configured disks
Objective	Verify BackupDisk rule properly validates disks with type differences and loose comparisons.
Preconditions	- Application running - Configuration set: backup.backup.destination.disks = ['disk1', 123, 'disk2']
Test Steps	1. Set configuration backup.backup.destination.disks to ['disk1', 123, 'disk2']. 2. Create a new BackupDisk object. 3. Validate 'disk1' as disk value. 4. Validate 'disk2' as disk value. 5. Validate integer 123 as disk value. 6. Validate string '123' as disk value. 7. Validate 'disk3' as disk value.
Test Data	- Configured disks: ['disk1', 123, 'disk2'] - Input disk values: 'disk1', 'disk2', 123, '123', 'disk3'

Expected Result	- Validation passes for 'disk1'. - Validation passes for 'disk2'. - Validation passes for 123. - Validation passes for '123' (string matches integer via loose comparison). - Validation fails for 'disk3'.
Actual Result	- Validation passed for 'disk1', 'disk2', 123, and '123'; failed for 'disk3'.
Status	Pass
Severity	High

File: BackupsController-Test.txt

Test Case ID	BC-001
Title	Verify 'index' Method Existence and JsonResponse Return
Objective	Ensure the 'index' method of BackupsController exists and returns a JsonResponse when called with a GET request.
Preconditions	- Application running - Backup configuration available - Required config keys: 'backup.backup.destination.disks', 'filesystems.default', 'backup.backup.name' seeded - Appropriate routes configured - Database seeded if necessary
Test Steps	1. Mock the configuration to return specific values for backup disks, default filesystem, and backup name. 2. Create a GET request to '/admin/backups'. 3. Call the 'index' method of BackupsController with the request object. 4. Check if the response is an instance of JsonResponse.
Test Data	- Config 'backup.backup.destination.disks': ['local'] - Config 'filesystems.default': 'local' - Config 'backup.backup.name': 'test-app' - Request: Method 'GET' to '/admin/backups'
Expected Result	- The 'index' method returns an object of type Illuminate\Http\JsonResponse.
Actual Result	- The 'index' method returns an object of type Illuminate\Http\JsonResponse.
Status	Pass
Severity	Medium

Test Case ID	BC-002
Title	Verify 'destroy' Method Existence and JsonResponse Return
Objective	Ensure the 'destroy' method of BackupsController exists and returns a JsonResponse when called with a valid backup file path.
Preconditions	- Application running - Backup functionality enabled - Required 'path' exists (e.g., 'test.zip' on the 'local' disk) - Appropriate routes configured - Database seeded if necessary
Test Steps	1. Mock the backup deletion request object. 2. Mock the 'validate' method to return ['path' => 'test.zip']. 3. Call the 'destroy' method of BackupsController with disk 'local' and the mocked request. 4. Check if the response is an instance of JsonResponse.
Test Data	- Request validation result: ['path' => 'test.zip'] - Disk: 'local'
Expected Result	- The 'destroy' method returns an object of type Illuminate\Http\JsonResponse.
Actual Result	- The 'destroy' method returns an object of type Illuminate\Http\JsonResponse.
Status	Pass
Severity	High

Test Case ID	BC-003
Title	BackupsController Instantiation
Objective	Confirm that an instance of BackupsController can be created successfully.
Preconditions	- Application running - BackupsController class available and autoloaded

Test Steps	1. Instantiate BackupsController. 2. Verify the instantiated object is of class BackupsController.
Test Data	- None (direct instantiation)
Expected Result	- Instance created is of class Crater\Http\Controllers\V1\Admin\Backup\BackupsController.
Actual Result	- Instance created is of class Crater\Http\Controllers\V1\Admin\Backup\BackupsController.
Status	Pass
Severity	Low

File: Base64Mime-Test.txt

Test Case ID	BM-001
Title	Constructor initializes extensions property correctly
Objective	Verify that the Base64Mime constructor sets the 'extensions' property as provided.
Preconditions	- Application is running - Database is seeded - PHPUnit/Pest test framework set up - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with an array of extensions: ['jpg', 'jpeg', 'png']. 2. Retrieve the 'extensions' property using reflection. 3. Check that the property matches the input array.
Test Data	- Input: ['jpg', 'jpeg', 'png'] - Expected: 'extensions' property equals ['jpg', 'jpeg', 'png']
Expected Result	- The 'extensions' property of the object is set to ['jpg', 'jpeg', 'png'].
Actual Result	The 'extensions' property is set to ['jpg', 'jpeg', 'png'] as expected.
Status	Pass
Severity	Medium

Test Case ID	BM-002
Title	Validation message returns correct error with attribute and extensions
Objective	Ensure that message() outputs the correct validation error message including the attribute and allowed extensions.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpg', 'jpeg']. 2. Use reflection to set the 'attribute' property to 'document_file'. 3. Call the message() method. 4. Verify the returned string matches the expected format.
Test Data	- Extensions: ['jpg', 'jpeg'] - Attribute: 'document_file' - Expected Message: "The document_file must be a json with file of type: jpg, jpeg encoded in base64."
Expected Result	- The message() method returns: "The document_file must be a json with file of type: jpg, jpeg encoded in base64."
Actual Result	The message() method returns the expected string.
Status	Pass
Severity	Medium

Test Case ID	BM-003
Title	Validation message returns correct error when no extensions are provided
Objective	Ensure that message() outputs proper validation message even when extensions array is empty.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with an empty extensions array. 2. Set 'attribute' property to 'avatar' using reflection. 3. Call the message() method. 4. Confirm the message format reflects no extensions.

Test Data	- Extensions: [] - Attribute: 'avatar' - Expected Message: "The avatar must be a json with file of type: encoded in base64."
Expected Result	- The message() returns: "The avatar must be a json with file of type: encoded in base64."
Actual Result	The message is returned as expected.
Status	Pass
Severity	Medium

Test Case ID	BM-004
Title	Fails validation with non-JSON string input
Objective	Validate that passes() returns false when input is not valid JSON.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpg']. 2. Call passes() with key 'file' and value 'this is not a json string but plain text'. 3. Assert the result is false.
Test Data	- Input value: 'this is not a json string but plain text' - Expected: false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-005
Title	Fails validation for valid JSON without "data" key
Objective	Ensure passes() returns false for JSON input missing the "data" key.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpg']. 2. Call passes() with key 'file' and value '{"name": "test.jpg", "size": 100}'. 3. Assert the result is false.
Test Data	- Input value: '{"name": "test.jpg", "size": 100}' - Expected: false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-006
Title	Fails validation for data URI with missing "data:" prefix
Objective	Confirm passes() returns false when "data" string lacks the 'data:' prefix.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpg']. 2. JSON encode: ['data' => 'image/jpeg;base64,R0lGODlhAQABAIAAAP///wAAACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw==']. 3. Call passes() with key 'file' and this value. 4. Assert the result is false.

Test Data	- Input value: '{"data":"image/jpegbase64,R0lGODlhAQABAIAAAP///wAAACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw=="}'- Expected: false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-007
Title	Fails validation for base64 string with invalid characters
Objective	Confirm passes() returns false for base64 strings containing invalid characters.
Preconditions	- Application is running- Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpg'].2. JSON encode: ['data' => '$'].3. Call passes() with key 'file' and this value.4. Assert the result is false.
Test Data	- Input: '{"data":"$"}'- Expected: false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-008
Title	Fails validation for missing comma separator in data string
Objective	Ensure passes() fails when base64 data URI lacks comma separator between metadata and data.
Preconditions	- Application is running- Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpg'].2. JSON encode: ['data' => 'data:image/jpeg;base64R0lGODlhAQABAIAAAP//wAAACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw=='].3. Call passes() with key 'file' and this value.4. Assert the result is false.
Test Data	- Input: '{"data":"data:image/jpeg;base64R0lGODlhAQABAIAAAP///wAAACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw=="}'- Expected: false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-009
Title	Fails validation for empty base64 encoded data after comma
Objective	Ensure passes() returns false if base64 encoded part of data string is empty.
Preconditions	- Application is running- Base64Mime class is loaded

Test Steps	1. Instantiate Base64Mime with extensions: ['jpg']. 2. JSON encode: ['data' => '']. 3. Call passes() with key 'file' and this value. 4. Assert the result is false.
Test Data	- Input: '{"data":""}' - Expected: false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-010
Title	Fails validation for unrecognized or unallowed binary data type
Objective	Ensure passes() returns false if data cannot be recognized as allowed file type or detected as unallowed type.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Create 32 bytes of arbitrary binary data. 2. Base64 encode the binary. 3. JSON encode: ['data' => 'data:application/octet-stream;base64,']. 4. Instantiate Base64Mime with extensions: ['jpg']. 5. Call passes() with key 'file' and this value. 6. Assert the result is false.
Test Data	- Input: '{"data":"data:application/octet-stream;base64,..." }' (base64 of binary garbage) - Allowed extension: ['jpg'] - Expected: false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-011
Title	Fails validation if detected file extension not in allowed list (single-part)
Objective	Ensure passes() returns false when MIME-detected extension does not match allowed extensions with single allowed value.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['pdf']. 2. Prepare a valid base64-encoded JPG data string in data URL format. 3. JSON encode the object: ['data' => '']. 4. Call passes() with key 'image' and this value. 5. Assert the result is false.
Test Data	- Allowed extension: ['pdf'] - Input: base64 data for 1x1 black JPG - Expected: passes() returns false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-012

Title	Fails validation if detected file extension not in allowed list (multi-part)
Objective	Ensure passes() returns false when MIME-detected extension does not match allowed extensions with multiple allowed extensions.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpg']. 2. Prepare a valid base64-encoded PNG data string as a data URL. 3. JSON encode object: ['data' => '']. 4. Call passes() with key 'image' and the value. 5. Assert the result is false.
Test Data	- Allowed extension: ['jpg'] - Input: base64 data for 1x1 black PNG - Expected: passes() returns false
Expected Result	- passes() returns false.
Actual Result	passes() returns false as expected.
Status	Pass
Severity	High

Test Case ID	BM-013
Title	Passes validation for allowed single-part extension
Objective	Ensure passes() returns true when file extension matches one of the allowed extensions (single part).
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpeg', 'png']. 2. Prepare a valid base64-encoded JPG. 3. JSON encode object: ['data' => '']. 4. Call passes() with key 'image' and this value. 5. Assert the result is true.
Test Data	- Allowed extensions: ['jpeg', 'png'] - Input: base64 data for 1x1 black JPG - Expected: passes() returns true
Expected Result	- passes() returns true.
Actual Result	passes() returns true as expected.
Status	Pass
Severity	High

Test Case ID	BM-014
Title	Passes validation for allowed multi-part extension
Objective	Ensure passes() returns true when file extension matches one of the allowed extensions (multi-part).
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpg', 'png']. 2. Prepare a valid base64-encoded PNG. 3. JSON encode object: ['data' => '']. 4. Call passes() with key 'image' and this value. 5. Assert the result is true.
Test Data	- Allowed extensions: ['jpg', 'png'] - Input: base64 data for 1x1 black PNG - Expected: passes() returns true
Expected Result	- passes() returns true.

Actual Result	passes() returns true as expected.
Status	Pass
Severity	High

Test Case ID	BM-015
Title	Passes validation when detected extension alias matches allowed extension
Objective	Ensure passes() returns true when detected extension (alias) matches an allowed extension.
Preconditions	- Application is running - Base64Mime class is loaded
Test Steps	1. Instantiate Base64Mime with extensions: ['jpeg']. 2. Prepare a valid base64-encoded JPG with MIME type 'image/pjpeg'. 3. JSON encode object: ['data' => '']. 4. Call passes() with key 'image' and this value. 5. Assert the result is true.
Test Data	- Allowed extension: ['jpeg'] - Input: base64 data for 1x1 black JPG, MIME 'image/pjpeg' - Expected: passes() returns true
Expected Result	- passes() returns true.
Actual Result	passes() returns true as expected.
Status	Pass
Severity	High

File: BroadcastServiceProvider-Test.txt

Test Case ID	BSP-001
Title	Verify Existence and Structure of BroadcastServiceProvider
Objective	To verify that the BroadcastServiceProvider class exists, adheres to required implementation patterns, and its key methods function without error.
Preconditions	- Application running - Database seeded - Crater\Providers\BroadcastServiceProvider class available - Illuminate\Support\ServiceProvider available - Application container accessible
Test Steps	1. Check if the BroadcastServiceProvider class exists in the codebase. 2. Create an instance of BroadcastServiceProvider using the application container. 3. Verify the instance is of type BroadcastServiceProvider. 4. Verify the instance is also a subclass of Illuminate\Support\ServiceProvider. 5. Check if the 'register' method exists in the provider. 6. Check if the 'boot' method exists in the provider. 7. Invoke the 'register' method and ensure no Exception is thrown.
Test Data	- Class: Crater\Providers\BroadcastServiceProvider - Application container injected for instantiation
Expected Result	- The BroadcastServiceProvider class exists. - An instance can be created via new BroadcastServiceProvider(app()). - The instance is of type BroadcastServiceProvider. - The instance extends Illuminate\Support\ServiceProvider. - The methods 'register' and 'boot' exist on the instance. - Invoking 'register' does not throw any Exception.
Actual Result	All conditions and assertions passed. The BroadcastServiceProvider class exists, is constructed correctly, has required methods, and its register method executes without error.
Status	Pass
Severity	High

Test Case ID	BSP-002
Title	Verify Existence of routes Method in Broadcast Facade Root
Objective	To confirm that the Broadcast facade root possesses the 'routes' method necessary for route registration.
Preconditions	- Application running - Database seeded - Illuminate\Support\Facades\Broadcast available and loaded
Test Steps	1. Retrieve the root object from the Broadcast facade using getFacadeRoot(). 2. Check if the 'routes' method exists on the Broadcast facade root object.
Test Data	- Facade: Illuminate\Support\Facades\Broadcast - Method checked: 'routes'
Expected Result	- The Broadcast facade root object has a 'routes' method defined.
Actual Result	The Broadcast facade root object possesses the 'routes' method as expected.
Status	Pass
Severity	High

File: BulkExchangeRateController-Test.txt

Test Case ID	BERC-001
Title	items method updates all child items and calls taxes
Objective	Verify that the 'items' method correctly updates all child items' financial fields (including base values and exchange rate) and also updates all associated taxes for these items.
Preconditions	- Application running - BulkExchangeRateController instantiated - Test items and taxes created - Test invoice created with items and taxes - Database seeded
Test Steps	1. Create two item objects with specific discount, price, tax, and total values. 2. Assign each item a tax object with specific amounts. 3. Create an invoice object with exchange_rate set to 1.5, and assign the items to it. 4. Invoke the 'items' method via reflection on the controller, passing the invoice model. 5. Check that each item's base_discount_val, base_price, base_tax, base_total, and exchange_rate are updated correctly. 6. Check that the update method has been called on each item. 7. Verify that each tax's exchange_rate and base_amount are correctly updated and the update method called.
Test Data	- item1: discount_val=10, price=50, tax=5, total=55, tax1 amount=5 - item2: discount_val=20, price=100, tax=10, total=110, tax2 amount=10 - Invoice: exchange_rate=1.5, items=[item1, item2] - Expected item1: exchange_rate=1.5, base_discount_val=15.0, base_price=75.0, base_tax=7.5, base_total=82.5, update called - Expected item2: exchange_rate=1.5, base_discount_val=30.0, base_price=150.0, base_tax=15.0, base_total=165.0 - Expected tax1: exchange_rate=1.5, base_amount=7.5, update called - Expected tax2: exchange_rate=1.5, base_amount=15.0
Expected Result	- Both items' financial fields (base_* values and exchange_rate) are correctly recalculated. - Update method is called on each item and tax. - Items and taxes reflect the correct values post-multiplication by exchange rate.
Actual Result	- All fields updated as expected and update methods were called.
Status	Pass
Severity	High

Test Case ID	BERC-002
Title	items method handles model with no child items
Objective	Verify that the 'items' method safely handles an invoice model with an empty items collection without error, and still calls taxes processing.
Preconditions	- Application running - BulkExchangeRateController instantiated - Invoice object created with no items - Database seeded
Test Steps	1. Create an invoice object with exchange_rate set to 1.5 and an empty items collection. 2. Invoke the 'items' method via reflection on the controller, passing the invoice model. 3. Validate that the taxes method is called on the model after item processing.
Test Data	- Invoice: exchange_rate=1.5, items=[] - Expected: model.calls should contain 'taxes'
Expected Result	- Taxes method called even if there are no child items.

Actual Result	- taxes method was called as expected.
Status	Pass
Severity	Medium

Test Case ID	BERC-003
Title	taxes method updates taxes when they exist
Objective	Verify that the 'taxes' method updates all tax objects with the correct exchange_rate and base_amount when taxes exist.
Preconditions	- Application running - BulkExchangeRateController instantiated - Model object created with taxes present - Database seeded
Test Steps	1. Create two tax objects with specific amounts. 2. Create a model object with an exchange_rate of 1.5 and assign the taxes to it. 3. Ensure the model's taxes() method returns exists=true. 4. Invoke the 'taxes' method via reflection on the controller. 5. Check each tax for updated exchange_rate and base_amount values. 6. Verify that the update method was called for each tax.
Test Data	- tax1: amount=25, expected base_amount=37.5, exchange_rate=1.5 - tax2: amount=30, expected base_amount=45.0, exchange_rate=1.5 - Model: taxes=[tax1, tax2], exchange_rate=1.5
Expected Result	- Both taxes updated: exchange_rate=1.5, base_amounts recalculated correctly, update method called.
Actual Result	- Both taxes’ fields updated and methods called as expected.
Status	Pass
Severity	High

Test Case ID	BERC-004
Title	taxes method does nothing when no taxes exist
Objective	Ensure the 'taxes' method does not attempt to process or update any tax objects when none exist (exists returns false).
Preconditions	- Application running - BulkExchangeRateController instantiated - Model object created with taxes() returning exists=false - Database seeded
Test Steps	1. Create a model object with an exchange_rate of 1.5 and no taxes. 2. Implement taxes() on the model to return exists=false. 3. Invoke the 'taxes' method via reflection on the controller. 4. Validate that no operations or updates are attempted on taxes.
Test Data	- Model: exchange_rate=1.5, taxes() returns exists=false
Expected Result	- No changes or operations performed; function exits cleanly.
Actual Result	- No operations performed as expected.
Status	Pass
Severity	Low

Test Case ID	BERC-005
Title	taxes method handles empty taxes collection even when exists returns true
Objective	Verify the 'taxes' method safely handles a model with an empty taxes collection, even when exists returns true (no exceptions/errors and no updates).

Preconditions	- Application running - BulkExchangeRateController instantiated - Model object created with an empty taxes collection and taxes() returns exists=true - Database seeded
Test Steps	1. Create a model with exchange_rate=1.5 and an empty taxes collection. 2. Implement taxes() to return exists=true. 3. Invoke the 'taxes' method via reflection on the controller. 4. Validate no changes or attempts to update non-existent tax objects.
Test Data	- Model: exchange_rate=1.5, taxes=[], taxes() returns exists=true
Expected Result	- No tax updates performed as there are no taxes present.
Actual Result	- No updates performed; function executed successfully.
Status	Pass
Severity	Low

Test Case ID	BERC-006
Title	items method calls taxes on model after processing items
Objective	Verify that the 'items' method processes an empty items collection and then properly calls taxes update logic for the model’s own taxes.
Preconditions	- Application running - BulkExchangeRateController instantiated - Invoice object created with exchange_rate and taxes but empty items - Database seeded
Test Steps	1. Create an invoice object with exchange_rate=1.5, empty items collection, and a tax object (amount=50) in its taxes. 2. Invoke the 'items' method via reflection on the controller, passing the invoice model. 3. Verify that the tax’s exchange_rate and base_amount are correctly updated. 4. Check that the tax object’s update method was called.
Test Data	- modelTax: amount=50, expected base_amount=75.0, exchange_rate=1.5 - Invoice: exchange_rate=1.5, items=[], taxes=[modelTax]
Expected Result	- modelTax’s exchange_rate set to 1.5, base_amount recalculated to 75.0, update called.
Actual Result	- Fields updated and update method called as expected.
Status	Pass
Severity	Medium

File: BulkExchangeRateRequest-Test.txt

Test Case ID	BERR-001
Title	Authorize method always returns true
Objective	Verify that the BulkExchangeRateRequest::authorize() method always returns true, allowing all users to proceed with the request.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available
Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Call the authorize() method. 3. Check the returned value.
Test Data	- BulkExchangeRateRequest instance
Expected Result	- The authorize() method returns true.
Actual Result	The authorize() method returned true as expected.
Status	Pass
Severity	Medium

Test Case ID	BERR-002
Title	Validation rules are correctly returned by rules() method
Objective	Validate that the rules() method of BulkExchangeRateRequest returns the correct array of validation rules for the currencies input.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available
Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Call the rules() method. 3. Compare the returned array to the expected validation rules.
Test Data	- BulkExchangeRateRequest instance - Expected rules: - 'currencies' => ['required'] - 'currencies.*.id' => ['required', 'numeric'] - 'currencies.*.exchange_rate' => ['required']
Expected Result	- The rules() method returns the exact expected array of validation rules.
Actual Result	rules() method returned the expected rules array.
Status	Pass
Severity	High

Test Case ID	BERR-003
Title	Validation passes with valid currencies data
Objective	Verify that validation passes when currencies data includes all required fields and valid values.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available - Validation rules available

Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Retrieve validation rules with rules() method. 3. Prepare valid input data: - 'currencies' => [['id' => 1, 'exchange_rate' => 1.23], ['id' => 2, 'exchange_rate' => 0.98],] 4. Create a validator instance with the data and rules. 5. Execute passes() check.
Test Data	- Input data: {'currencies': [{'id': 1, 'exchange_rate': 1.23}, {'id': 2, 'exchange_rate': 0.98}]} - Validation rules from rules() method
Expected Result	- Validator passes() returns true.
Actual Result	Validator passed with valid currencies data.
Status	Pass
Severity	High

Test Case ID	BERR-004
Title	Validation fails when currencies array is missing
Objective	Confirm that the validation fails and errors are returned when the currencies array is omitted from the data.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available - Validation rules available
Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Retrieve validation rules with rules() method. 3. Prepare input data with no currencies key: {} 4. Create a validator instance with data and rules. 5. Execute fails() check. 6. Check validator errors for 'currencies'.
Test Data	- Input data: {} - Validation rules from rules() method
Expected Result	- Validator fails() returns true. - Validator errors contain 'currencies'.
Actual Result	Validation failed as expected; 'currencies' error present.
Status	Pass
Severity	High

Test Case ID	BERR-005
Title	Validation fails when currencies array is empty
Objective	Validate the system rejects empty currencies array, triggering errors.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available - Validation rules available
Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Retrieve validation rules with rules() method. 3. Prepare input data: {'currencies': []} 4. Create a validator instance with data and rules. 5. Execute fails() check. 6. Check validator errors for 'currencies'.
Test Data	- Input data: {'currencies': []} - Validation rules from rules() method

Expected Result	- Validator fails() returns true. - Validator errors contain 'currencies'.
Actual Result	Validation failed as expected; 'currencies' error present.
Status	Pass
Severity	High

Test Case ID	BERR-006
Title	Validation fails when currency id is missing
Objective	Ensure that the absence of the 'id' field in currencies items results in a validation error.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available - Validation rules available
Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Retrieve validation rules with rules() method. 3. Prepare input data: {'currencies': [{'exchange_rate': 1.23}]} 4. Create a validator instance with data and rules. 5. Execute fails() check. 6. Check validator errors for 'currencies.0.id'.
Test Data	- Input data: {'currencies': [{'exchange_rate': 1.23}]} - Validation rules from rules() method
Expected Result	- Validator fails() returns true. - Validator errors contain 'currencies.0.id'.
Actual Result	Validation failed as expected; 'currencies.0.id' error present.
Status	Pass
Severity	High

Test Case ID	BERR-007
Title	Validation fails when currency id is not numeric
Objective	Verify that a non-numeric currency 'id' in the data triggers a validation error.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available - Validation rules available
Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Retrieve validation rules with rules() method. 3. Prepare input data: {'currencies': [{'id': 'abc', 'exchange_rate': 1.23}]} 4. Create a validator instance with data and rules. 5. Execute fails() check. 6. Check validator errors for 'currencies.0.id'.
Test Data	- Input data: {'currencies': [{'id': 'abc', 'exchange_rate': 1.23}]} - Validation rules from rules() method
Expected Result	- Validator fails() returns true. - Validator errors contain 'currencies.0.id'.
Actual Result	Validation failed as expected; 'currencies.0.id' error present.
Status	Pass
Severity	High

Test Case ID	BERR-008
Title	Validation fails when currency exchange_rate is missing

Objective	Confirm that missing 'exchange_rate' field in currencies items causes a validation error.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available - Validation rules available
Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Retrieve validation rules with rules() method. 3. Prepare input data: {'currencies': [{'id': 1}]} 4. Create a validator instance with data and rules. 5. Execute fails() check. 6. Check validator errors for 'currencies.0.exchange_rate'.
Test Data	- Input data: {'currencies': [{'id': 1}]} - Validation rules from rules() method
Expected Result	- Validator fails() returns true. - Validator errors contain 'currencies.0.exchange_rate'.
Actual Result	Validation failed as expected; 'currencies.0.exchange_rate' error present.
Status	Pass
Severity	High

Test Case ID	BERR-009
Title	Validation fails with multiple issues in currency items
Objective	Validate that when multiple currency items have missing or incorrect fields, multiple errors are triggered for each specific issue.
Preconditions	- Application running - Container and validator instantiated - BulkExchangeRateRequest class available - Validation rules available
Test Steps	1. Instantiate BulkExchangeRateRequest. 2. Retrieve validation rules with rules() method. 3. Prepare input data: - 'currencies' => [['id' => 'abc', 'exchange_rate' => 1.23], // Invalid ID ['id' => 2], // Missing exchange_rate ['exchange_rate' => 4.56], // Missing ID] 4. Create a validator instance with data and rules. 5. Execute fails() check. 6. Check validator errors for 'currencies.0.id', 'currencies.1.exchange_rate', and 'currencies.2.id'.
Test Data	- Input data: {'currencies': [{'id': 'abc', 'exchange_rate': 1.23}, {'id': 2}, {'exchange_rate': 4.56}]} - Validation rules from rules() method
Expected Result	- Validator fails() returns true. - Validator errors contain 'currencies.0.id'. - Validator errors contain 'currencies.1.exchange_rate'. - Validator errors contain 'currencies.2.id'.
Actual Result	Validation failed as expected; errors for 'currencies.0.id', 'currencies.1.exchange_rate', and 'currencies.2.id' all present.
Status	Pass
Severity	High

File: CheckInvoiceStatus-Test.txt

Test Case ID	CIS-001
Title	Verify existence of CheckInvoiceStatus command class
Objective	To confirm the CheckInvoiceStatus command class is defined and available in the application.
Preconditions	- Application is running - Source code contains Crater\Console\Commands\CheckInvoiceStatus - Database seeded (required for application, not directly for this test)
Test Steps	1. Attempt to load the Crater\Console\Commands\CheckInvoiceStatus class. 2. Check if the class exists in the codebase.
Test Data	- Crater\Console\Commands\CheckInvoiceStatus
Expected Result	- The CheckInvoiceStatus class should exist and be detected by class_exists().
Actual Result	CheckInvoiceStatus class exists and is detected successfully.
Status	Pass
Severity	High

Test Case ID	CIS-002
Title	Verify CheckInvoiceStatus command properties and inheritance
Objective	To validate the CheckInvoiceStatus command has the correct name, description, and extends the base Command class.
Preconditions	- Application running - CheckInvoiceStatus class defined - Database seeded
Test Steps	1. Instantiate the CheckInvoiceStatus command class. 2. Retrieve the command name using getName(). 3. Retrieve the command description using getDescription(). 4. Verify the instance inherits from Illuminate\Console\Command.
Test Data	- Command name: 'check:invoices:status' - Command description: 'Check invoices status.'
Expected Result	- getName() returns 'check:invoices:status'. - getDescription() returns 'Check invoices status.' - The instance is of Illuminate\Console\Command.
Actual Result	Command name, description, and inheritance are all correct.
Status	Pass
Severity	High

Test Case ID	CIS-003
Title	Verify presence of essential methods in CheckInvoiceStatus command
Objective	Ensure the CheckInvoiceStatus command has handle and __construct methods implemented.
Preconditions	- Application running - CheckInvoiceStatus class defined - Database seeded
Test Steps	1. Instantiate the CheckInvoiceStatus command class. 2. Check if the class has a method named 'handle'. 3. Check if the class has a method named '__construct'.
Test Data	- CheckInvoiceStatus command class
Expected Result	- method_exists($command, 'handle') returns true. - method_exists($command, '__construct') returns true.

Actual Result	Both 'handle' and '__construct' methods exist in the class.
Status	Pass
Severity	High

Test Case ID	CIS-004
Title	Validate command signature property in CheckInvoiceStatus command
Objective	Confirm the 'signature' property of CheckInvoiceStatus is set correctly.
Preconditions	- Application running - CheckInvoiceStatus class exists - Database seeded
Test Steps	1. Use reflection to access the 'signature' property of CheckInvoiceStatus. 2. Retrieve the value of the 'signature' property. 3. Verify it matches the expected signature.
Test Data	- Expected signature: 'check:invoices:status'
Expected Result	- The 'signature' property should be 'check:invoices:status'.
Actual Result	The 'signature' property is correctly set to 'check:invoices:status'.
Status	Pass
Severity	High

Test Case ID	CIS-005
Title	Validate command description property in CheckInvoiceStatus command
Objective	Confirm the 'description' property in CheckInvoiceStatus is set to the correct value.
Preconditions	- Application running - CheckInvoiceStatus class exists - Database seeded
Test Steps	1. Use reflection to access the 'description' property of CheckInvoiceStatus. 2. Retrieve the value of the 'description' property. 3. Verify it matches the expected description.
Test Data	- Expected description: 'Check invoices status.'
Expected Result	- The 'description' property should be 'Check invoices status.'
Actual Result	The 'description' property is correctly set to 'Check invoices status.'
Status	Pass
Severity	High

Test Case ID	CIS-006
Title	Validate structure and required properties/methods for CheckInvoiceStatus command
Objective	To ensure the CheckInvoiceStatus command can be instantiated and contains essential properties.
Preconditions	- Application running - CheckInvoiceStatus class exists - Database seeded
Test Steps	1. Instantiate CheckInvoiceStatus command. 2. Verify the object is created successfully. 3. Using reflection, check that the object has a 'signature' property. 4. Using reflection, check that the object has a 'description' property. 5. Using reflection, check that the object has a 'handle' method.
Test Data	- CheckInvoiceStatus command class

Expected Result	- Object is instantiated successfully. - 'signature' property exists. - 'description' property exists. - 'handle' method exists.
Actual Result	Object structure and required properties/methods are all valid.
Status	Pass
Severity	High

File: CloneInvoiceController-Test.txt

Test Case ID	CIC-001
Title	CloneInvoiceController Instantiation
Objective	Verify that the CloneInvoiceController can be instantiated correctly and is an instance of the expected controller classes.
Preconditions	- Application running - All required classes autoloaded - Dependencies for CloneInvoiceController available
Test Steps	1. Instantiate the CloneInvoiceController. 2. Verify the instance is of type CloneInvoiceController. 3. Verify the instance is of type Illuminate\Routing\Controller.
Test Data	- Controller class: CloneInvoiceController - Expected classes: CloneInvoiceController, Illuminate\Routing\Controller
Expected Result	- The instantiated object is an instance of CloneInvoiceController. - The instantiated object is also an instance of Illuminate\Routing\Controller.
Actual Result	Both instance checks passed; the object is of the correct types.
Status	Pass
Severity	High

Test Case ID	CIC-002
Title	CloneInvoiceController __invoke Method Existence and Signature
Objective	Verify that the CloneInvoiceController has a public __invoke method with correct parameter types.
Preconditions	- Application running - CloneInvoiceController class loaded
Test Steps	1. Check if the __invoke method exists in CloneInvoiceController. 2. Reflect on the method to check if it is public. 3. Confirm that the method has exactly two parameters. 4. Check parameter types: first is Illuminate\Http\Request, second is Crater\Models\Invoice.
Test Data	- None (method signature inspection)
Expected Result	- __invoke method exists. - __invoke is public. - Method has two parameters: Illuminate\Http\Request and Crater\Models\Invoice.
Actual Result	All requirements matched; method exists with correct signature.
Status	Pass
Severity	High

Test Case ID	CIC-003
Title	Controller Authorization Logic
Objective	Verify that the controller uses appropriate authorization when performing actions.
Preconditions	- Application running - Controller class available
Test Steps	1. Check if the authorize method exists in CloneInvoiceController. 2. Inspect the controller source code for a call to authorize with 'create' and Invoice::class.
Test Data	- Authorization method: 'authorize' - Expected string in source: "authorize('create', Invoice::class)"

Expected Result	- authorize method exists in the controller. - Controller source contains authorization check for invoice creation.
Actual Result	Authorization method and call found in source code as expected.
Status	Pass
Severity	High

Test Case ID	CIC-004
Title	Carbon Date Handling in Controller
Objective	Ensure the controller uses Carbon for date management and specific Carbon methods.
Preconditions	- Controller class available - Source code is accessible
Test Steps	1. Scan the controller source for usage of Carbon\Carbon. 2. Look for call to Carbon::now() in the code.
Test Data	- Required usage: 'Carbon\Carbon', 'Carbon::now()'
Expected Result	- Controller uses Carbon\Carbon. - Controller invokes Carbon::now() for date handling.
Actual Result	Both usages found in source code.
Status	Pass
Severity	Medium

Test Case ID	CIC-005
Title	Use of Required Models and Services in Controller
Objective	Confirm that the controller uses all required classes.
Preconditions	- Application running - Source code accessible
Test Steps	1. Scan controller source for these classes: - Crater\Models\Invoice - Crater\Models\CompanySetting - Crater\Services\SerialNumberFormatter - Vinkla\Hashids\Facades\Hashids - Crater\Http\Resources\InvoiceResource
Test Data	- Required class names
Expected Result	- All required classes are present in the controller source.
Actual Result	All listed classes found in controller source.
Status	Pass
Severity	High

Test Case ID	CIC-006
Title	Existence of Company Setting Constants
Objective	Verify that required company setting keys exist and are strings, and getSetting method exists.
Preconditions	- Application running - CompanySetting class available
Test Steps	1. Check if CompanySetting::getSetting method exists. 2. Confirm settings keys 'invoice_set_due_date_automatically' and 'invoice_due_date_days' are strings.
Test Data	- Setting keys: 'invoice_set_due_date_automatically', 'invoice_due_date_days'
Expected Result	- getSetting method is present. - Setting keys are of string type.

Actual Result	Method present; both keys confirmed as strings.
Status	Pass
Severity	Medium

Test Case ID	CIC-007
Title	Invoice Status Constants Existence and Type
Objective	Validate that STATUS_DRAFT and STATUS_UNPAID exist in Invoice model and are strings.
Preconditions	- Application running - Invoice model loaded
Test Steps	1. Check if constants STATUS_DRAFT and STATUS_UNPAID are defined. 2. Verify that each constant is of string type.
Test Data	- Constants: STATUS_DRAFT, STATUS_UNPAID
Expected Result	- Both constants are defined in Invoice. - Both constants are strings.
Actual Result	Constants found and confirmed as strings.
Status	Pass
Severity	Medium

Test Case ID	CIC-008
Title	Due Date Logic Handling in Controller
Objective	Verify due date setting logic based on company configuration.
Preconditions	- Application running - Carbon library loaded
Test Steps	1. For each data set, determine if due date should be set: a. dueDateSetting = 'YES', dueDateDays = 30, shouldSetDueDate = true b. dueDateSetting = 'NO', dueDateDays = 30, shouldSetDueDate = false c. dueDateSetting = 'YES', dueDateDays = null, shouldSetDueDate = false d. dueDateSetting = 'NO', dueDateDays = null, shouldSetDueDate = false 2. If due date should be set, calculate expected due date using Carbon. 3. Check that resulting due date is a string of length 10 (YYYY-MM-DD).
Test Data	- Various combinations as outlined above
Expected Result	- Due date logic matches shouldSetDueDate for each configuration. - Due date (if set) is in correct format.
Actual Result	Logic matches for all cases; expected due date calculated and formatted correctly.
Status	Pass
Severity	High

Test Case ID	CIC-009
Title	Exchange Rate Calculation
Objective	Ensure controller correctly calculates amounts based on exchange rate.
Preconditions	- Application running
Test Steps	1. For each data set, multiply amount by exchange rate and compare with expected base amount: a. exchangeRate = 1.0, amount = 100.0, expectedBaseAmount = 100.0 b. exchangeRate = 1.5, amount = 100.0, expectedBaseAmount = 150.0 c. exchangeRate = 0.8, amount = 100.0, expectedBaseAmount = 80.0 d. exchangeRate = 2.0, amount = 50.0, expectedBaseAmount = 100.0
Test Data	- as above

Expected Result	- Calculated amount matches expected base amount for each test data set.
Actual Result	All calculated amounts matched expectations.
Status	Pass
Severity	High

Test Case ID	CIC-010
Title	Invoice Item Cloning Logic
Objective	Confirm that invoice items are cloned and recalculated correctly, with proper tax filtering.
Preconditions	- Application running
Test Steps	1. Set up test item with fields: name, price, discount_val, tax, total, taxes. 2. Set exchange rate = 1.5, recalculate base_price, base_discount_val, base_tax, base_total. 3. Verify recalculated values match expectations. 4. Apply tax filtering to only include taxes with amount > 0. 5. Verify only valid taxes are included and check the name of the valid tax.
Test Data	- Test item data: as described
Expected Result	- base_price = 150.0 - base_discount_val = 15.0 - base_tax = 30.0 - base_total = 165.0 - Only taxes with amount > 0 included; 'Tax 1' is the only valid tax post-filter.
Actual Result	All recalculated values and filtering logic correct; only 'Tax 1' included as valid tax.
Status	Pass
Severity	High

Test Case ID	CIC-011
Title	Invoice Resource Type Returned by Controller
Objective	Check that the controller returns the correct API resource type.
Preconditions	- Application running - InvoiceResource class available
Test Steps	1. Check controller source for 'return new InvoiceResource'. 2. Verify InvoiceResource class existence. 3. Confirm that InvoiceResource extends Illuminate\Http\Resources\Json\JsonResource.
Test Data	- Class name: InvoiceResource
Expected Result	- Controller returns InvoiceResource. - InvoiceResource class exists and extends JsonResource.
Actual Result	Controller returns correct resource type; InvoiceResource confirmed and inheritance verified.
Status	Pass
Severity	Medium

Test Case ID	CIC-012
Title	SerialNumberFormatter Class Existence and Methods
Objective	Validate that SerialNumberFormatter class exists and has required API methods.
Preconditions	- Application running

Test Steps	1. Confirm existence of Crater\Services\SerialNumberFormatter class. 2. Create an instance of SerialNumberFormatter. 3. For each method in ['setModel', 'setCompany', 'setCustomer', 'setNextNumbers', 'getNextNumber']: - Check method exists in SerialNumberFormatter.
Test Data	- Method names as above
Expected Result	- SerialNumberFormatter exists and has all required methods.
Actual Result	Class and all methods found as expected.
Status	Pass
Severity	Medium

Test Case ID	CIC-013
Title	Data Transformation Logic in Controller
Objective	Confirm that controller performs all necessary data transformations for invoice cloning.
Preconditions	- Application running
Test Steps	1. Set up test data for original invoice with sub_total, discount, discount_type, discount_val, total, tax, exchange_rate. 2. Calculate base_total, base_discount_val, base_sub_total, base_tax, base_due_amount using exchange rate. 3. Verify the calculated values are numeric and match the expected calculations.
Test Data	- original_invoice: sub_total = 1000, discount = 100, discount_type = 'percentage', discount_val = 100, total = 900, tax = 180, exchange_rate = 1.5 - expected_calculations: base_total = 1350, base_discount_val = 150, base_sub_total = 1500, base_tax = 270, base_due_amount = 1350
Expected Result	- All calculated base fields are numeric and match expected results.
Actual Result	Calculated values are numeric and correct for all fields.
Status	Pass
Severity	High

File: CompaniesController-Test.txt

Test Case ID	CC-001
Title	Instantiate CompaniesController
Objective	Verify that the CompaniesController can be instantiated and is an instance of both CompaniesController and Illuminate\Routing\Controller.
Preconditions	- Application running - Required codebase loaded - CompaniesController class available
Test Steps	1. Instantiate the CompaniesController. 2. Assert that the instance is of type CompaniesController. 3. Assert that the instance is also of type Illuminate\Routing\Controller.
Test Data	- CompaniesController object instance
Expected Result	- The instance is of type CompaniesController. - The instance is of type Illuminate\Routing\Controller.
Actual Result	The instance was properly created and matches both expected types.
Status	Pass
Severity	High

Test Case ID	CC-002
Title	Verify Controller Has All Required Methods
Objective	Ensure that CompaniesController has the methods: store, destroy, transferOwnership, and getUserCompanies.
Preconditions	- Application running - CompaniesController class available
Test Steps	1. Define the required methods: store, destroy, transferOwnership, getUserCompanies. 2. For each method, assert that it exists in the controller.
Test Data	- Methods list: ['store', 'destroy', 'transferOwnership', 'getUserCompanies']
Expected Result	- All specified methods exist in CompaniesController.
Actual Result	All required methods are present in the controller.
Status	Pass
Severity	High

Test Case ID	CC-003
Title	Store Method Flow Verification
Objective	Verify store method includes authorization, company creation, attach logic, and returns correct resource.
Preconditions	- Application running - CompaniesController class available
Test Steps	1. Obtain store method source code from CompaniesController. 2. Check for the following steps: a. Authorization for creating company b. Company::create invocation c. Hashids usage for unique hash d. Company save process e. Default data setup f. Attachment of company to user g. Super admin role assignment h. Address creation (if provided) i. Returning CompanyResource
Test Data	- Store method source code

Expected Result	- Each expected steps is present in the method.
Actual Result	All steps found in the store method implementation.
Status	Pass
Severity	High

Test Case ID	CC-004
Title	Destroy Method Authorization Verification
Objective	Ensure the destroy method requires delete company authorization.
Preconditions	- Application running - CompaniesController class available
Test Steps	1. Obtain destroy method source code from CompaniesController. 2. Check for authorization call with 'delete company'.
Test Data	- Destroy method source code
Expected Result	- Destroy method contains authorization for 'delete company'.
Actual Result	Authorization step found as expected.
Status	Pass
Severity	High

Test Case ID	CC-005
Title	Destroy Method Validation Flow
Objective	Confirm that destroy method handles all validation branches properly.
Preconditions	- Application running - CompaniesController class available
Test Steps	1. Extract destroy method's implementation. 2. Search for handling of the following validation branches: a. Error if company name mismatches (company_name_must_match_with_given_name). b. Error if user attempts to delete their last company (You_cannot_delete_all_companies). c. Success flow for deletion (deleteCompany).
Test Data	- Branch keywords: "company_name_must_match_with_given_name", "You_cannot_delete_all_companies", "deleteCompany"
Expected Result	- All validation branches are covered in destroy method.
Actual Result	All branches found in the destroy method logic.
Status	Pass
Severity	High

Test Case ID	CC-006
Title	TransferOwnership Method Authorization Verification
Objective	Ensure transferOwnership method enforces transfer company ownership authorization.
Preconditions	- Application running - CompaniesController class available
Test Steps	1. Obtain transferOwnership method source code. 2. Check for authorization call with 'transfer company ownership'.
Test Data	- TransferOwnership method source code
Expected Result	- Method includes authorization for 'transfer company ownership'.
Actual Result	Authorization found in the method as required.
Status	Pass

Severity	High

Test Case ID	CC-007
Title	TransferOwnership Method Success and Failure Handling
Objective	Ensure transferOwnership method correctly handles both success and failure scenarios.
Preconditions	- Application running - CompaniesController class available
Test Steps	1. Obtain transferOwnership method implementation. 2. Confirm correct logic for: a. Handling when 'success' is false. b. Handling when 'success' is true. c. Logic involving hasCompany(). d. Role syncing using BouncerFacade::sync.
Test Data	- Method implementation containing 'success' states and BouncerFacade::sync
Expected Result	- Both success and failure cases are handled in the method. - hasCompany and BouncerFacade::sync are properly used.
Actual Result	All success and failure cases with correct data handling were confirmed.
Status	Pass
Severity	High

Test Case ID	CC-008
Title	Verify Controller References Correct Models and Facades
Objective	Confirm CompaniesController imports and uses the required classes/models.
Preconditions	- Application running - CompaniesController class available
Test Steps	1. Obtain CompaniesController source code. 2. Verify inclusion of the following: a. Crater\Models\Company b. Crater\Models\User c. Crater\Http\Requests\CompaniesRequest d. Silber\Bouncer\BouncerFacade e. Vinkla\Hashids\Facades\Hashids f. Crater\Http\Resources\CompanyResource
Test Data	- List of required classes
Expected Result	- All specified models and facades are imported or referenced in the controller.
Actual Result	All required models and facades found in controller source.
Status	Pass
Severity	High

Test Case ID	CC-009
Title	Verify User Model Methods Required by Controller
Objective	Ensure the User model has all methods required by CompaniesController.
Preconditions	- Application running - User model available
Test Steps	1. Instantiate User model object. 2. Assert presence of methods: a. companies b. loadCount c. hasCompany d. assign 3. Confirm companies() relationship returns a Collection.

Test Data	- User object - Methods: companies, loadCount, hasCompany, assign
Expected Result	- All methods exist on User model. - companies() relationship yields a Collection.
Actual Result	All required methods and correct relationship found.
Status	Pass
Severity	High

Test Case ID	CC-010
Title	CompanyResource Existence and Inheritance
Objective	Confirm that CompanyResource exists and extends JsonResource appropriately.
Preconditions	- Application running - CompanyResource class available
Test Steps	1. Assert CompanyResource class existence. 2. Assert parent class of CompanyResource is Illuminate\Http\Resources\Json\JsonResource.
Test Data	- CompanyResource class
Expected Result	- CompanyResource exists and inherits from required parent.
Actual Result	CompanyResource exists and extends JsonResource as expected.
Status	Pass
Severity	Medium

Test Case ID	CC-011
Title	Destroy Method Validation Logic Coverage
Objective	Validate destroy method logic for all input scenarios: name mismatch, single company, and success.
Preconditions	- Application running - Destroy method logic available
Test Steps	1. For each scenario, provide input data: a. Name matches, multiple companies b. Name mismatches, multiple companies c. Name matches, single company d. Name mismatches, single company 2. Compare shouldPassValidation against expected succeed state. 3. Verify returned error type matches expectations.
Test Data	- ['Company A', 'Company A', 2, true, 'success'] - ['Company A', 'Company B', 2, false, 'company_name_must_match_with_given_name'] - ['Company A', 'Company A', 1, false, 'You_cannot_delete_all_companies'] - ['Wrong', 'Company A', 1, false, 'company_name_must_match_with_given_name']
Expected Result	- shouldPassValidation and error type match for each scenario.
Actual Result	All input scenarios validated with correct result and error type.
Status	Pass
Severity	High

Test Case ID	CC-012
Title	Transfer Ownership Validation Logic Coverage
Objective	Verify validation logic for transfer ownership based on user-company relationship.

Preconditions	- Application running - TransferOwnership method logic available
Test Steps	1. For each test case: a. Set userHasCompany boolean b. Set shouldSucceed expected result 2. Validate: - If userHasCompany = true, success = false, message is 'User does not belongs to this company.' - If userHasCompany = false, success = true, no error message.
Test Data	- [true, false] - [false, true]
Expected Result	- Logic matches expected behavior for both user ownership scenarios.
Actual Result	Both scenarios validated, logic matches expected outcomes.
Status	Pass
Severity	High

Test Case ID	CC-013
Title	Store Method Data Flow Coverage
Objective	Verify all steps taken in store method, including conditional address creation.
Preconditions	- Application running - Store method logic available
Test Steps	1. For each input scenario: a. Set hasAddress boolean b. Set shouldCreateAddress expectation 2. Verify all steps except address creation are always true. 3. Confirm address creation step only happens if hasAddress is true.
Test Data	- [true, true] - [false, false]
Expected Result	- All mandatory steps always occur. - Address creation step matches hasAddress condition.
Actual Result	Data flow matches expected sequence for both address scenarios.
Status	Pass
Severity	High

Test Case ID	CC-014
Title	getUserCompanies Returns Correct Data Structure
Objective	Ensure getUserCompanies method returns a collection of CompanyResource and uses correct user companies relationship.
Preconditions	- Application running - getUserCompanies method available
Test Steps	1. Obtain getUserCompanies method source code. 2. Check for presence of: a. CompanyResource::collection in the output. b. Usage of $request->user()->companies.
Test Data	- getUserCompanies method source
Expected Result	- Collection of CompanyResource returned, using user companies relationship.
Actual Result	Both collection and relationship found as expected.
Status	Pass
Severity	High

Test Case ID	CC-015
Title	Controller Uses Proper Request Validation
Objective	Confirm CompaniesController uses CompaniesRequest type hint and getCompanyPayload method for validation.
Preconditions	- Application running - Controller source available
Test Steps	1. Check for CompaniesRequest $request type hint in controller. 2. Verify presence of getCompanyPayload() method.
Test Data	- Controller source code
Expected Result	- CompaniesRequest type hint used. - getCompanyPayload() method found.
Actual Result	Both request type hint and method present as required.
Status	Pass
Severity	High

Test Case ID	CC-016
Title	Address Creation Follows Conditional Logic
Objective	Ensure address creation only occurs if an address exists in the request within the store method.
Preconditions	- Application running - Store method available
Test Steps	1. Obtain store method source code. 2. Check for conditional code: 'if ($request->address)' to decide address creation.
Test Data	- Source code with conditional check
Expected Result	- Address is only created if $request->address is set.
Actual Result	Conditional logic for address creation confirmed.
Status	Pass
Severity	Medium

Test Case ID	CC-017
Title	Role Assignment Uses Correct String
Objective	Ensure store method assigns 'super admin' role string to user.
Preconditions	- Application running - Store method available
Test Steps	1. Obtain store method source code. 2. Verify presence of: assign('super admin')
Test Data	- Assignment string in code
Expected Result	- User assigned 'super admin' role in store method.
Actual Result	Role assignment string confirmed.
Status	Pass
Severity	High

Test Case ID	CC-018
Title	Bouncer Role Sync Uses Correct Array
Objective	Confirm transferOwnership method syncs user roles using correct array format.

Preconditions	- Application running - transferOwnership method available
Test Steps	1. Obtain transferOwnership method implementation. 2. Check for: sync($user)->roles(['super admin'])
Test Data	- Method implementation code
Expected Result	- BouncerFacade syncs user roles with proper array.
Actual Result	Bouncer role sync confirmed as expected.
Status	Pass
Severity	High

File: CompanyCollection-Test.txt

Test Case ID	CC-001
Title	Returns an empty array when the company collection is empty
Objective	Verify that the CompanyCollection resource returns an empty array when initialized with an empty collection.
Preconditions	- Application running - No company records in the current collection - CompanyCollection resource is available
Test Steps	1. Initialize an empty Illuminate\Support\Collection. 2. Create a new CompanyCollection instance with the empty collection. 3. Convert the CompanyCollection to an array using the HTTP Request object.
Test Data	- Input Collection: [] - Expected Output: []
Expected Result	- The result is an array. - The result array is empty.
Actual Result	- The result is an array. - The result array is empty.
Status	Pass
Severity	Medium

Test Case ID	CC-002
Title	Transforms a collection of company objects correctly
Objective	Verify that a CompanyCollection properly transforms a collection of company objects to an array and maintains the correct count.
Preconditions	- Application running - Two valid company model objects exist - CompanyCollection resource is available
Test Steps	1. Create two TestCompanyModel objects with the following data: - Item 1: id=1, name="Company A", email="a@example.com", logo=null - Item 2: id=2, name="Company B", email="b@example.com", logo=null 2. Initialize a CompanyCollection with these two objects. 3. Convert the CompanyCollection to an array using the HTTP Request object.
Test Data	- Input Objects: - [{id: 1, name: "Company A", email: "a@example.com", logo: null}, {id: 2, name: "Company B", email: "b@example.com", logo: null}] - Expected Count: 2
Expected Result	- The result is an array. - The result array has a count of 2.
Actual Result	- The result is an array. - The result array has a count of 2.
Status	Pass
Severity	Medium

Test Case ID	CC-003
Title	Transforms collection items that are already CompanyResource instances
Objective	Verify that CompanyCollection accurately transforms a collection containing CompanyResource objects and maintains the expected count.
Preconditions	- Application running - Two CompanyResource instances wrapping company model objects exist - CompanyCollection resource is available

Test Steps	1. Create two TestCompanyModel objects with: - Company 1: id=101, name="Mock Company A", email="mocka@example.com" - Company 2: id=102, name="Mock Company B", email="mockb@example.com" 2. Wrap each TestCompanyModel in a CompanyResource. 3. Create a Collection of the two CompanyResource objects. 4. Initialize a CompanyCollection with the Collection. 5. Convert the CompanyCollection to an array using the HTTP Request object.
Test Data	- Input: [CompanyResource(Mock Company A), CompanyResource(Mock Company B)] - Expected Count: 2
Expected Result	- The result is an array. - The result array has a count of 2.
Actual Result	- The result is an array. - The result array has a count of 2.
Status	Pass
Severity	Medium

Test Case ID	CC-004
Title	Instantiates CompanyCollection with various collection types
Objective	Ensure CompanyCollection can be constructed with different collection types (array, Collection, paginator) and returns an array for empty collections.
Preconditions	- Application running - CompanyCollection resource is available - Various collection types available
Test Steps	For each collection type (array, Illuminate\Support\Collection, LengthAwarePaginator): 1. Initialize CompanyCollection with the given collection. 2. Assert that the instance is of type CompanyCollection. 3. If the collection is empty, convert to array using HTTP Request and validate result.
Test Data	- Input Collections: - [] (empty array) - new Collection([]) - new LengthAwarePaginator([], 0, 10, 1) - [] (as 'empty') - Expected Output for empty: []
Expected Result	- The CompanyCollection instance is created successfully for each type. - For empty collection inputs, toArray() returns an array.
Actual Result	- The CompanyCollection instance is created for all provided types. - For empty collections, toArray() returns an array.
Status	Pass
Severity	Medium

Test Case ID	CC-005
Title	Transforms an array of company objects wrapped in CompanyResource
Objective	Verify that when given an array of CompanyResource objects, the CompanyCollection transforms and returns an array of the correct size.
Preconditions	- Application running - Two valid TestCompanyModel objects exist - CompanyResource and CompanyCollection resources are available

Test Steps	1. Create two TestCompanyModel objects: - Object 1: id=1, name="Test 1", email="test1@example.com" - Object 2: id=2, name="Test 2", email="test2@example.com" 2. Wrap each in a CompanyResource. 3. Create an array of the two CompanyResource objects. 4. Initialize CompanyCollection with the array. 5. Convert the CompanyCollection to an array using the HTTP Request object.
Test Data	- Input: [CompanyResource(Test 1), CompanyResource(Test 2)] - Expected Count: 2
Expected Result	- The result is an array. - The result array has a count of 2.
Actual Result	- The result is an array. - The result array has a count of 2.
Status	Pass
Severity	Medium

File: CompanyCurrencyCheckTransactionsController-Test.txt

Test Case ID	CCCTC-001
Title	Instantiating CompanyCurrencyCheckTransactionsController
Objective	Verify that the controller can be instantiated and is an instance of both CompanyCurrencyCheckTransactionsController and Illuminate\Routing\Controller.
Preconditions	- Application running - PHP environment set up - Required classes available
Test Steps	1. Instantiate the CompanyCurrencyCheckTransactionsController. 2. Verify the instance is of type CompanyCurrencyCheckTransactionsController. 3. Verify the instance is of type Illuminate\Routing\Controller.
Test Data	- Controller class: Crater\Http\Controllers\V1\Admin\Settings\CompanyCurrencyCheckTransactionsController
Expected Result	- Controller is successfully instantiated. - Controller is instance of CompanyCurrencyCheckTransactionsController. - Controller is instance of Illuminate\Routing\Controller.
Actual Result	Controller instantiated and type checks passed.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-002
Title	__invoke Method Parameters Validation
Objective	Ensure that the __invoke method on the controller only has one parameter of type Illuminate\Http\Request.
Preconditions	- Controller class is available - Application running
Test Steps	1. Use ReflectionClass to access the __invoke method. 2. Count the parameters of the method. 3. Retrieve the parameter type.
Test Data	- Method: __invoke - Expected parameter type: Illuminate\Http\Request
Expected Result	- __invoke method has exactly one parameter of type Illuminate\Http\Request.
Actual Result	__invoke method has one parameter of type Illuminate\Http\Request.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-003
Title	Controller Authorization Requirement
Objective	Verify that the controller requires 'manage company' authorization within the __invoke method.
Preconditions	- Controller class available - Source code accessible
Test Steps	1. Read the source code for the __invoke method. 2. Search for "authorize('manage company'" in the method's source.
Test Data	- Authorization string: "authorize('manage company'"
Expected Result	- The __invoke method contains the 'manage company' authorization check.
Actual Result	Authorization string found in method source.

Status	Pass
Severity	High

Test Case ID	CCCTC-004
Title	Execution Flow Verification
Objective	Ensure the controller follows the proper execution steps: finding company, authorization, transaction check, and response.
Preconditions	- Controller class available - Source code accessible
Test Steps	1. Read source code for __invoke method. 2. Search for the following steps/patterns: - Company::find - Authorization check for 'manage company' - hasTransactions check - response()->json - has_transactions key in response
Test Data	- Patterns: Company::find, authorize.*manage company, hasTransactions, response.*json, has_transactions
Expected Result	- All the expected execution steps are present in the method.
Actual Result	All required steps found in method source.
Status	Pass
Severity	High

Test Case ID	CCCTC-005
Title	Company Model hasTransactions Method Existence
Objective	Verify that the Company model includes the hasTransactions method.
Preconditions	- Company model available
Test Steps	1. Instantiate a Company model. 2. Verify that hasTransactions method exists.
Test Data	- Company model: Crater\Models\Company
Expected Result	- hasTransactions method exists on Company model.
Actual Result	hasTransactions method exists on Company model.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-006
Title	Company Header Used from Request
Objective	Ensure the controller reads the 'company' header from incoming requests.
Preconditions	- Controller class available - Source code accessible
Test Steps	1. Read source for the __invoke method. 2. Search for usage of header('company').
Test Data	- Header: 'company'
Expected Result	- The method accesses 'company' via request header.
Actual Result	header('company') found in method source.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-007
Title	Correct Response Structure
Objective	Verify that the controller returns the expected response structure with 'has_transactions' key.
Preconditions	- Controller class available - Source code accessible
Test Steps	1. Read source for __invoke method. 2. Ensure response contains '"has_transactions' =>". 3. Verify usage of $company->hasTransactions(). 4. Check for response()->json usage.
Test Data	- Response structure: ['has_transactions' => bool]
Expected Result	- Response contains key 'has_transactions' - Value of 'has_transactions' from hasTransactions() - Response returned as JSON
Actual Result	Response structure and JSON return confirmed in method.
Status	Pass
Severity	High

Test Case ID	CCCTC-008
Title	Authorization Method Exists on Controller
Objective	Ensure the controller contains an 'authorize' method.
Preconditions	- Controller class available
Test Steps	1. Check if 'authorize' method exists on controller.
Test Data	- Method: authorize
Expected Result	- 'authorize' method exists on controller.
Actual Result	authorize method exists on controller.
Status	Pass
Severity	High

Test Case ID	CCCTC-009
Title	Controller Handles hasTransactions Outcomes
Objective	Verify the controller returns the actual value of hasTransactions in response.
Preconditions	- Application running
Test Steps	1. Provide boolean value for hasTransactions (true/false). 2. Simulate controller response using hasTransactions value. 3. Confirm response accurately reflects input. 4. Ensure response contains 'has_transactions' key.
Test Data	- Inputs: hasTransactions = true, expectedHasTransactions = true - Inputs: hasTransactions = false, expectedHasTransactions = false
Expected Result	- response['has_transactions'] matches input value - response always contains 'has_transactions' key
Actual Result	Response matches input value and contains required key.
Status	Pass
Severity	High

Test Case ID	CCCTC-010
Title	Response has_transactions is Boolean
Objective	Ensure the 'has_transactions' key in response always holds a boolean value.

Preconditions	- Source code accessible
Test Steps	1. Inspect __invoke method for usage of hasTransactions(). 2. Verify return type from hasTransactions is boolean.
Test Data	- Method: hasTransactions() - Expected type: boolean
Expected Result	- has_transactions value is always boolean.
Actual Result	has_transactions value is boolean as expected.
Status	Pass
Severity	High

Test Case ID	CCCTC-011
Title	No Conditional Branches Except Authorization
Objective	Ensure the controller method contains no conditional logic aside from possible authorization check.
Preconditions	- Controller class available - Source code accessible
Test Steps	1. Count if/else statements in __invoke method. 2. Count number of return statements. 3. Confirm only authorization logic has conditionals.
Test Data	- Source code: __invoke method
Expected Result	- Maximum one conditional branch (authorization) - Only one return statement - No else branches
Actual Result	Only authorization branch found, no else branches, single return.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-012
Title	Controller Name Matches Purpose
Objective	Confirm the controller name accurately reflects its purpose and follows naming conventions.
Preconditions	- Controller class available
Test Steps	1. Retrieve controller's class name. 2. Check name for 'CompanyCurrencyCheck' and 'Transactions' substrings.
Test Data	- ClassName: CompanyCurrencyCheckTransactionsController
Expected Result	- Class name contains purpose-relevant keywords.
Actual Result	Class name reflects purpose and contains all keywords.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-013
Title	Controller Namespace Matches Laravel Convention
Objective	Ensure the controller's namespace adheres to Laravel's best practices for organization.
Preconditions	- Controller class available
Test Steps	1. Retrieve controller's namespace using ReflectionClass. 2. Confirm namespace starts with 'Crater\Http\Controllers'. 3. Check for 'V1\Admin\Settings' in namespace.
Test Data	- Namespace: Crater\Http\Controllers\V1\Admin\Settings

Expected Result	- Namespace meets Laravel conventions.
Actual Result	Namespace structure matches Laravel standards.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-014
Title	Response JSON Structure Validation
Objective	Confirm response always contains 'has_transactions' key of boolean type.
Preconditions	- Application running
Test Steps	1. Create response object with various has_transactions values. 2. Check key existence. 3. Confirm value is boolean. 4. Confirm type is 'boolean'.
Test Data	- has_transactions: true, expected_type: boolean - has_transactions: false, expected_type: boolean
Expected Result	- Response always contains 'has_transactions' as boolean.
Actual Result	Response contains correct keys and boolean types.
Status	Pass
Severity	High

Test Case ID	CCCTC-015
Title	Controller Method Length is Short and Focused
Objective	Ensure the __invoke method is concise and focused (≤ 15 lines).
Preconditions	- Source code accessible
Test Steps	1. Use ReflectionClass to get start and end lines of __invoke method. 2. Calculate line count.
Test Data	- Method: __invoke
Expected Result	- Method is 15 lines or fewer.
Actual Result	Method length confirmed at ≤ 15 lines.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-016
Title	Dependency Injection Usage Validation
Objective	Verify controller uses dependency injection correctly, only injecting Request.
Preconditions	- Controller class available
Test Steps	1. Check __invoke method parameters: should only be Request. 2. Confirm controller constructor injects no additional dependencies.
Test Data	- Method parameter: Illuminate\Http\Request - Constructor parameters: none
Expected Result	- __invoke only uses Request parameter - No constructor dependencies
Actual Result	Only Request dependency injected.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-017

Title	No Side Effects Except Response
Objective	Ensure the controller method only prepares the response and has no side effects like database writes.
Preconditions	- Controller class available - Source code accessible
Test Steps	1. Count assignment operations (excluding necessary $company = ...). 2. Search for database write operations: create, update, delete, save, insert.
Test Data	- Assignment count ≤ 3
Expected Result	- No write operations found - Only permissible assignments present
Actual Result	No side effects; only response created.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-018
Title	Response Always Contains has_transactions Key
Objective	Ensure controller's response always includes 'has_transactions' key.
Preconditions	- Application running
Test Steps	1. Prepare response as controller would. 2. Check for existence of 'has_transactions' key. 3. Verify response has only one key.
Test Data	- Response: ['has_transactions' => true/false]
Expected Result	- Response contains 'has_transactions' key and no extra keys.
Actual Result	Response contains only 'has_transactions' key.
Status	Pass
Severity	Medium

Test Case ID	CCCTC-019
Title	Authorization Failure Scenario Handling
Objective	Validate controller correctly checks authorization before performing other logic.
Preconditions	- Controller class available - Source code accessible
Test Steps	1. Read the __invoke method source lines. 2. Locate line index where Company::find occurs. 3. Locate line index for authorization check. 4. Compare their order.
Test Data	- Source code of __invoke method
Expected Result	- Authorization is checked after finding company, before other logic.
Actual Result	Authorization checked at correct point in method.
Status	Pass
Severity	High

File: CompanyLogoRequest-Test.txt

Test Case ID	CLR-001
Title	CompanyLogoRequest authorize method returns true
Objective	Verify that the authorize() method of CompanyLogoRequest returns true, indicating authorization for the request is granted.
Preconditions	- Application running - Required dependencies loaded - CompanyLogoRequest class available
Test Steps	1. Instantiate a CompanyLogoRequest object. 2. Initialize the form request with the application container. 3. Call the authorize() method on the request instance. 4. Assert that the returned value is true.
Test Data	- Request instance: new CompanyLogoRequest() - Expected value: true
Expected Result	authorize() method returns true, granting authorization for the request.
Actual Result	authorize() method returned true as expected.
Status	Pass
Severity	Medium

Test Case ID	CLR-002
Title	CompanyLogoRequest rules method returns correct validation rules
Objective	Validate that the rules() method of CompanyLogoRequest returns the correct array of validation rules for the company_logo field.
Preconditions	- Application running - CompanyLogoRequest class available - Base64Mime rule class available
Test Steps	1. Instantiate a CompanyLogoRequest object. 2. Call the rules() method to retrieve validation rules. 3. Assert that the returned value is an array and contains the 'company_logo' key. 4. Retrieve validation rules for 'company_logo'. 5. Assert that company_logo rules are an array containing 'nullable'. 6. Assert that company_logo rules contain an instance of Base64Mime.
Test Data	- Request instance: new CompanyLogoRequest() - Expected keys: ['company_logo'] - Expected rules: ['nullable', instance of Base64Mime]
Expected Result	- rules() returns an array containing the 'company_logo' key. - The 'company_logo' rules contain 'nullable' and an instance of Base64Mime.
Actual Result	All assertions passed; rules contained required keys and instances.
Status	Pass
Severity	High

Test Case ID	CLR-003
Title	Validation passes with valid base64 image
Objective	Confirm that validation passes when a valid base64-encoded PNG image in JSON format is provided to company_logo.
Preconditions	- Application running - CompanyLogoRequest and Base64Mime classes available - Validator facade available

Test Steps	1. Retrieve validation rules from a CompanyLogoRequest instance. 2. Create valid 1x1 transparent PNG base64 string. 3. JSON encode as ['data' => valid base64 PNG]. 4. Pass JSON string to validator for company_logo using the rules. 5. Assert that validator passes.
Test Data	- Input: {'company_logo' => '{"data":"..."}'} - Expected value: passes() -> true
Expected Result	Validation passes for valid base64 PNG image in company_logo.
Actual Result	Validation passed successfully.
Status	Pass
Severity	High

Test Case ID	CLR-004
Title	Validation passes with null value for company_logo
Objective	Confirm that providing a null value for company_logo successfully passes validation.
Preconditions	- Application running - CompanyLogoRequest and Base64Mime classes available - Validator facade available
Test Steps	1. Retrieve validation rules from a CompanyLogoRequest instance. 2. Pass null as the value for company_logo to the validator. 3. Assert that validation passes.
Test Data	- Input: {'company_logo' => null} - Expected value: passes() -> true
Expected Result	Validation passes when company_logo is null.
Actual Result	Validation passed; null value accepted.
Status	Pass
Severity	Medium

Test Case ID	CLR-005
Title	Validation passes with empty string for company_logo
Objective	Confirm that providing an empty string for company_logo passes validation without error.
Preconditions	- Application running - CompanyLogoRequest and Base64Mime classes available - Validator facade available
Test Steps	1. Retrieve validation rules from a CompanyLogoRequest instance. 2. Pass empty string as the value for company_logo to the validator. 3. Assert that validation passes.
Test Data	- Input: {'company_logo' => ''} - Expected value: passes() -> true
Expected Result	Validation passes with empty string as input for company_logo.
Actual Result	Validation passed; empty string accepted.
Status	Pass
Severity	Medium

Test Case ID	CLR-006
Title	Validation fails with invalid JSON string for company_logo
Objective	Verify that validation fails when company_logo is provided an invalid JSON string.

Preconditions	- Application running - CompanyLogoRequest and Base64Mime classes available - Validator facade available
Test Steps	1. Retrieve validation rules from a CompanyLogoRequest instance. 2. Prepare invalid JSON string for company_logo. 3. Pass invalid JSON string to the validator. 4. Assert that validation fails.
Test Data	- Input: {'company_logo' => '{invalid json'} - Expected value: fails() -> true
Expected Result	Validation fails due to invalid JSON format in company_logo.
Actual Result	Validation failed as expected for invalid JSON input.
Status	Pass
Severity	High

Test Case ID	CLR-007
Title	Validation fails with JSON missing data property
Objective	Check that validation fails when the JSON input for company_logo does not contain the required 'data' property.
Preconditions	- Application running - CompanyLogoRequest and Base64Mime classes available - Validator facade available
Test Steps	1. Retrieve validation rules from a CompanyLogoRequest instance. 2. Prepare valid JSON without 'data' property for company_logo. 3. Pass JSON string to the validator. 4. Assert that validation fails.
Test Data	- Input: {'company_logo' => '{"something":"else"}'} - Expected value: fails() -> true
Expected Result	Validation fails when 'data' property is missing from company_logo JSON input.
Actual Result	Validation failed as expected when 'data' property missing.
Status	Pass
Severity	High

Test Case ID	CLR-008
Title	Validation fails with invalid file type for company_logo
Objective	Ensure that validation fails when a PDF file type is provided in the base64 data for company_logo.
Preconditions	- Application running - CompanyLogoRequest and Base64Mime classes available - Validator facade available
Test Steps	1. Retrieve validation rules from a CompanyLogoRequest instance. 2. Prepare base64 PDF data string and encode as JSON with 'data' property. 3. Pass JSON string with PDF data to the validator. 4. Assert that validation fails.
Test Data	- Input: {'company_logo' => '{"data":"data:application/pdf;base64,..."}'} - Expected value: fails() -> true
Expected Result	Validation fails when file type is PDF and not an allowed image type.
Actual Result	Validation failed as expected due to invalid file type.
Status	Pass
Severity	High

Test Case ID	CLR-009
Title	Base64Mime rule validates allowed types directly
Objective	Verify Base64Mime rule allows only specified file types, passing for valid types and failing for invalid types.
Preconditions	- Application running - Base64Mime rule class available
Test Steps	1. Instantiate Base64Mime rule with allowed types ['gif', 'jpg', 'png']. 2. Prepare valid base64 PNG image as JSON with 'data' property. 3. Call passes() method of rule with valid PNG JSON; assert it returns true. 4. Prepare invalid PDF base64 string in JSON with 'data' property. 5. Call passes() with invalid PDF JSON and assert it returns false.
Test Data	- Allowed types: ['gif', 'jpg', 'png'] - Input 1: '{"data":"..."}' - Input 2: '{"data":"data:application/pdf;base64,..."}' - Expected 1: passes() -> true - Expected 2: passes() -> false
Expected Result	- passes() returns true for allowed PNG type. - passes() returns false for disallowed PDF type.
Actual Result	- True for PNG; false for PDF as expected.
Status	Pass
Severity	High

Test Case ID	CLR-010
Title	Request rules are functionally consistent across HTTP methods
Objective	Ensure that the validation rules in CompanyLogoRequest are consistent for different HTTP methods (POST and PUT).
Preconditions	- Application running - CompanyLogoRequest class available - Base64Mime rule class available
Test Steps	1. Instantiate two CompanyLogoRequest objects (simulating POST and PUT). 2. Retrieve rules from both objects. 3. Assert that both rule sets have the same keys. 4. Assert that 'company_logo' rules have the same number of rules for both. 5. Assert that both contain 'nullable'. 6. Assert that both contain an instance of Base64Mime.
Test Data	- Request methods: POST and PUT (simulated using instances) - Expected keys: ['company_logo'] - Expected rules: same number, both containing 'nullable' and Base64Mime
Expected Result	Rules sets for both methods are consistent in structure and content.
Actual Result	Rules were consistent as expected for both request methods.
Status	Pass
Severity	High

Test Case ID	CLR-011
Title	Base64Mime rule accepts allowed extensions parameter
Objective	Confirm that Base64Mime rule can be instantiated with an array of allowed file extensions and that it correctly validates an allowed file type.
Preconditions	- Application running - Base64Mime rule class available
Test Steps	1. Instantiate Base64Mime rule with allowed types ['gif', 'jpg', 'png']. 2. Assert that the rule is an instance of Base64Mime. 3. Prepare valid base64 PNG image as JSON with 'data' property. 4. Call passes() method of rule with valid PNG JSON and assert it returns true.

Test Data	- Allowed types: ['gif', 'jpg', 'png'] - Input: '{"data":"..."}' - Expected: passes() -> true
Expected Result	- Rule is an instance of Base64Mime. - passes() returns true for valid PNG type.
Actual Result	Base64Mime instantiated correctly and PNG validation passed.
Status	Pass
Severity	Medium

File: CompanyPolicy-Test.txt

Test Case ID	CP-001
Title	Company Creation Allowed for Owner User
Objective	Verify that the create method of CompanyPolicy allows company creation when the user is an owner.
Preconditions	- Application running - Database seeded - The user object exists and is set to owner
Test Steps	1. Mock a User object and set its isOwner method to return true. 2. Instantiate a CompanyPolicy object. 3. Call the create method on CompanyPolicy with the mocked user. 4. Observe the result.
Test Data	- User: isOwner = true
Expected Result	- result = true (Company creation is allowed)
Actual Result	result = true (Company creation is allowed as expected)
Status	Pass
Severity	High

Test Case ID	CP-002
Title	Company Creation Denied for Non-owner User
Objective	Verify that the create method of CompanyPolicy denies company creation when the user is not an owner.
Preconditions	- Application running - Database seeded - The user object exists and is set not to owner
Test Steps	1. Mock a User object and set its isOwner method to return false. 2. Instantiate a CompanyPolicy object. 3. Call the create method on CompanyPolicy with the mocked user. 4. Observe the result.
Test Data	- User: isOwner = false
Expected Result	- result = false (Company creation is denied)
Actual Result	result = false (Company creation is denied as expected)
Status	Pass
Severity	High

Test Case ID	CP-003
Title	Company Deletion Allowed for Owner User
Objective	Verify that the delete method of CompanyPolicy allows company deletion when the user is the owner of the company.
Preconditions	- Application running - Database seeded - User and Company objects exist - User is owner of the company (user.id = company.owner_id)
Test Steps	1. Create a User object with id = 1. 2. Create a Company object with owner_id = 1. 3. Instantiate a CompanyPolicy object. 4. Call the delete method on CompanyPolicy with the user and company. 5. Observe the result.
Test Data	- User: id = 1 - Company: owner_id = 1

Expected Result	- result = true (Company deletion is allowed)
Actual Result	result = true (Company deletion is allowed as expected)
Status	Pass
Severity	High

Test Case ID	CP-004
Title	Company Deletion Denied for Non-owner User
Objective	Verify that the delete method of CompanyPolicy denies company deletion when the user is not the owner of the company.
Preconditions	- Application running - Database seeded - User and Company objects exist - User is not owner of the company (user.id ≠ company.owner_id)
Test Steps	1. Create a User object with id = 1. 2. Create a Company object with owner_id = 2. 3. Instantiate a CompanyPolicy object. 4. Call the delete method on CompanyPolicy with the user and company. 5. Observe the result.
Test Data	- User: id = 1 - Company: owner_id = 2
Expected Result	- result = false (Company deletion is denied)
Actual Result	result = false (Company deletion is denied as expected)
Status	Pass
Severity	High

Test Case ID	CP-005
Title	Ownership Transfer Allowed for Owner User
Objective	Verify that the transferOwnership method of CompanyPolicy allows ownership transfer when the user is the owner of the company.
Preconditions	- Application running - Database seeded - User and Company objects exist - User is owner of the company (user.id = company.owner_id)
Test Steps	1. Create a User object with id = 1. 2. Create a Company object with owner_id = 1. 3. Instantiate a CompanyPolicy object. 4. Call the transferOwnership method on CompanyPolicy with the user and company. 5. Observe the result.
Test Data	- User: id = 1 - Company: owner_id = 1
Expected Result	- result = true (Ownership transfer is allowed)
Actual Result	result = true (Ownership transfer is allowed as expected)
Status	Pass
Severity	High

Test Case ID	CP-006
Title	Ownership Transfer Denied for Non-owner User
Objective	Verify that the transferOwnership method of CompanyPolicy denies ownership transfer when the user is not the owner of the company.

Preconditions	- Application running - Database seeded - User and Company objects exist - User is not owner of the company (user.id ≠ company.owner_id)
Test Steps	1. Create a User object with id = 1. 2. Create a Company object with owner_id = 2. 3. Instantiate a CompanyPolicy object. 4. Call the transferOwnership method on CompanyPolicy with the user and company. 5. Observe the result.
Test Data	- User: id = 1 - Company: owner_id = 2
Expected Result	- result = false (Ownership transfer is denied)
Actual Result	result = false (Ownership transfer is denied as expected)
Status	Pass
Severity	High

File: CompanySettingRequest-Test.txt

Test Case ID	CSR-001
Title	Authorization Method Returns True for CompanySettingRequest
Objective	Verify that the authorize() method of CompanySettingRequest always returns true.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\CompanySettingRequest class available
Test Steps	1. Instantiate a new CompanySettingRequest object. 2. Call the authorize() method of the object. 3. Observe the returned value.
Test Data	- CompanySettingRequest instantiated with no parameters.
Expected Result	- The authorize() method returns true.
Actual Result	The authorize() method returned true.
Status	Pass
Severity	High

Test Case ID	CSR-002
Title	CompanySettingRequest Returns Correct Validation Rules
Objective	Verify that CompanySettingRequest returns the expected validation rules for company settings.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\CompanySettingRequest class available
Test Steps	1. Instantiate a new CompanySettingRequest object. 2. Call the rules() method of the object. 3. Compare the returned rules array to the expected validation rules.
Test Data	- Expected validation rules: 'currency' => ['required'], 'time_zone' => ['required'], 'language' => ['required'], 'fiscal_year' => ['required'], 'moment_date_format' => ['required'], 'carbon_date_format' => ['required']
Expected Result	- The rules() method returns an array matching the expected validation rules for all specified fields.
Actual Result	The rules() method returned the expected array with all required rules matching.
Status	Pass
Severity	High

File: CompanySettingTest.txt

Test Case ID	CST-001
Title	CompanySetting model exposes correct fillable attributes
Objective	Verify that the CompanySetting model's fillable attributes are correctly set to allow mass assignment of required fields.
Preconditions	- Application running - Database seeded - Crater\Models\CompanySetting class available
Test Steps	1. Instantiate a CompanySetting model object. 2. Retrieve the fillable attributes using getFillable() method.
Test Data	- Model: CompanySetting - Expected fillable attributes: ['company_id', 'option', 'value']
Expected Result	- The getFillable() method returns ['company_id', 'option', 'value'].
Actual Result	getFillable() returned ['company_id', 'option', 'value'] as expected.
Status	Pass
Severity	High

Test Case ID	CST-002
Title	CompanySetting model belongsTo company relationship
Objective	Verify that the CompanySetting model defines a belongsTo relationship with the Company model.
Preconditions	- Application running - Database seeded with at least one company - Crater\Models\CompanySetting and Company classes available
Test Steps	1. Instantiate a CompanySetting model object. 2. Call the company() relationship method.
Test Data	- Model: CompanySetting - Expected relationship type: BelongsTo
Expected Result	- The company() method returns an instance of BelongsTo relationship.
Actual Result	company() returned an instance of BelongsTo as expected.
Status	Pass
Severity	High

Test Case ID	CST-003
Title	getSettings mapping returns correct option-value pairs
Objective	Verify that the getSettings method correctly maps option-value pairs into a keyed collection.
Preconditions	- Application running - Database seeded - Collection helper available
Test Steps	1. Prepare a collection with option-value pairs: [{'option': 'theme', 'value': 'dark'}, {'option': 'language', 'value': 'en'}]. 2. Map the collection keys using mapWithKeys function. 3. Convert the result to an array.
Test Data	- Input: [{'option': 'theme', 'value': 'dark'}, {'option': 'language', 'value': 'en'}] - Expected keys: 'theme' => 'dark', 'language' => 'en'
Expected Result	- The result is an instance of Collection. - The mapped collection array is ['theme' => 'dark', 'language' => 'en'].
Actual Result	Collection mapping produced ['theme' => 'dark', 'language' => 'en'].

Status	Pass
Severity	Medium

Test Case ID	CST-004
Title	CompanySetting model allows setting and retrieving attributes
Objective	Ensure that attributes of the CompanySetting model can be set and retrieved correctly.
Preconditions	- Application running - Database seeded - Crater\Models\CompanySetting class available
Test Steps	1. Instantiate CompanySetting model. 2. Set company_id to 777. 3. Set option to 'test_option'. 4. Set value to 'test_value'. 5. Retrieve each attribute and check the value.
Test Data	- company_id: 777 - option: 'test_option' - value: 'test_value'
Expected Result	- company_id equals 777. - option equals 'test_option'. - value equals 'test_value'.
Actual Result	Attributes returned expected values: 777, 'test_option', 'test_value'.
Status	Pass
Severity	High

Test Case ID	CST-005
Title	CompanySetting model handles multiple value types
Objective	Verify that the CompanySetting model's value attribute can store and retrieve different data types.
Preconditions	- Application running - Database seeded - Crater\Models\CompanySetting class available
Test Steps	1. Instantiate CompanySetting model. 2. Set value to 'Hello World' and verify. 3. Set value to 123 (integer) and verify. 4. Set value to 123.45 (float) and verify. 5. Set value to true (boolean) and verify. 6. Set value to null and verify. 7. Set value to ['a','b','c'] (array) and verify. 8. Set value to object with key 'value' and verify.
Test Data	- string: 'Hello World' - integer: 123 - float: 123.45 - boolean: true - null: null - array: ['a','b','c'] - object: (object) ['key' => 'value']
Expected Result	- In each iteration, model->value equals the test value set.
Actual Result	All value types stored and retrieved as expected.
Status	Pass
Severity	Medium

Test Case ID	CST-006
Title	CompanySetting model supports mass assignment for fillable attributes

Objective	Verify that fillable attributes can be assigned in a single fill() operation and reject non-fillable ones.
Preconditions	- Application running - Database seeded - Crater\Models\CompanySetting class available
Test Steps	1. Instantiate CompanySetting model. 2. Use fill() to assign company_id = 999, option = 'mass_option', value = 'mass_value'. 3. Retrieve company_id, option, and value.
Test Data	- Input for fill: ['company_id' => 999, 'option' => 'mass_option', 'value' => 'mass_value'] - Only fillable attributes should be set.
Expected Result	- company_id equals 999. - option equals 'mass_option'. - value equals 'mass_value'.
Actual Result	Mass assignment set company_id = 999, option = 'mass_option', value = 'mass_value'.
Status	Pass
Severity	High

Test Case ID	CST-007
Title	CompanySetting model has correct table name
Objective	Ensure the model's getTable() method returns 'company_settings' as the table name.
Preconditions	- Application running - Database seeded - Crater\Models\CompanySetting class available
Test Steps	1. Instantiate CompanySetting model. 2. Call getTable() method.
Test Data	- Expected table name: 'company_settings'
Expected Result	- getTable() returns 'company_settings'.
Actual Result	getTable() returned 'company_settings'.
Status	Pass
Severity	Medium

Test Case ID	CST-008
Title	CompanySetting model utilizes timestamps
Objective	Verify that the CompanySetting model's usesTimestamps() method returns true.
Preconditions	- Application running - Database seeded - Crater\Models\CompanySetting class available
Test Steps	1. Instantiate CompanySetting model. 2. Call usesTimestamps().
Test Data	- Expected value: true
Expected Result	- usesTimestamps() method returns true.
Actual Result	usesTimestamps() was true.
Status	Pass
Severity	Medium

Test Case ID	CST-009

Title	mapWithKeys helper correctly transforms collection to key-value mapping
Objective	Verify the mapWithKeys logic accurately maps collection of option-value pairs to associative array.
Preconditions	- Application running - Database seeded - Collection helper available
Test Steps	1. Prepare a collection: [{'option': 'key1', 'value': 'value1'}, {'option': 'key2', 'value': 'value2'}, {'option': 'key3', 'value': 'value3'}]. 2. Use mapWithKeys to transform collection. 3. Convert result to array and verify keys and values.
Test Data	- Input: [{'option': 'key1', 'value': 'value1'}, {'option': 'key2', 'value': 'value2'}, {'option': 'key3', 'value': 'value3'}] - Expected output: ['key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3']
Expected Result	- Result is a Collection instance. - Result array equals ['key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3'].
Actual Result	Collection transformed as expected: ['key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3'].
Status	Pass
Severity	Medium

Test Case ID	CST-010
Title	CompanySetting model handles special option names correctly
Objective	Verify that the model can store and retrieve options with special characters, spaces, and unicode in their names.
Preconditions	- Application running - Database seeded - Crater\Models\CompanySetting class available
Test Steps	1. Instantiate CompanySetting model. 2. For each special option name in: - 'dot.name' => 'dot_value' - 'dash-name' => 'dash_value' - 'underscore_name' => 'underscore_value' - 'name with spaces' => 'space_value' - 'name@special#chars' => 'special_value' - 'unicode_name_■' => 'unicode_value' a. Set model->option to the special name. b. Set model->value to the corresponding value. c. Retrieve and verify both option and value.
Test Data	- Special option names and values as above.
Expected Result	- For each input, model->option returns special name, model->value returns corresponding value.
Actual Result	Model stored and retrieved all special option names and their values correctly.
Status	Pass
Severity	Medium

File: CompleteModuleInstallationController-Test.txt

Test Case ID	CMIC-001
Title	Authorizes module management successfully
Objective	Verify that module management authorization passes and the controller returns successful installation response.
Preconditions	- Application running - Database seeded with module data - User has permission to manage modules - ModuleInstaller is available
Test Steps	1. Create a Request object with module='test_module' and version='1.0.0'. 2. Mock the Gate to authorize 'manage modules'. 3. Mock ModuleInstaller::complete to succeed for 'test_module', '1.0.0'. 4. Call CompleteModuleInstallationController with the request. 5. Check if the response is an instance of JsonResponse. 6. Verify the response data is ['success' => true]. 7. Verify the response HTTP status code is 200.
Test Data	- Input: module = 'test_module', version = '1.0.0' - Expected values: JsonResponse, data ['success' => true], status code 200
Expected Result	- Response is a JsonResponse instance. - Response data equals ['success' => true]. - Response status code is 200.
Actual Result	- Response is a JsonResponse instance. - Response data equals ['success' => true]. - Response status code is 200.
Status	Pass
Severity	High

Test Case ID	CMIC-002
Title	Installs a module and returns success on completion
Objective	Verify that ModuleInstaller::complete is called for valid module and version, and returns a successful install response.
Preconditions	- Application running - Database seeded with module data - User has permission to manage modules - ModuleInstaller is available
Test Steps	1. Create a Request object with module='my-awesome-module' and version='2.5.1'. 2. Mock the Gate to authorize 'manage modules'. 3. Mock ModuleInstaller::complete to succeed for 'my-awesome-module', '2.5.1'. 4. Call CompleteModuleInstallationController with the request. 5. Verify response is an instance of JsonResponse. 6. Check the response data is ['success' => true]. 7. Check the HTTP status is 200.
Test Data	- Input: module = 'my-awesome-module', version = '2.5.1' - Expected values: JsonResponse, data ['success' => true], status code 200
Expected Result	- Response is a JsonResponse instance. - Response data equals ['success' => true]. - Response status code is 200.
Actual Result	- Response is a JsonResponse instance. - Response data equals ['success' => true]. - Response status code is 200.
Status	Pass
Severity	High

Test Case ID	CMIC-003
Title	Handles module installation failure and returns success false
Objective	Verify that when ModuleInstaller::complete fails, the controller returns a failure response.
Preconditions	- Application running - Database seeded with module data - User has permission to manage modules - ModuleInstaller is available
Test Steps	1. Create a Request object with module='failing-module' and version='1.0.0'. 2. Mock the Gate to authorize 'manage modules'. 3. Mock ModuleInstaller::complete to return false for 'failing-module', '1.0.0'. 4. Call CompleteModuleInstallationController with the request. 5. Verify response is an instance of JsonResponse. 6. Check the response data is ['success' => false]. 7. Check the HTTP status is 200.
Test Data	- Input: module = 'failing-module', version = '1.0.0' - Expected values: JsonResponse, data ['success' => false], status code 200
Expected Result	- Response is a JsonResponse instance. - Response data equals ['success' => false]. - Response status code is 200.
Actual Result	- Response is a JsonResponse instance. - Response data equals ['success' => false]. - Response status code is 200.
Status	Pass
Severity	High

Test Case ID	CMIC-004
Title	Handles null or empty module/version gracefully
Objective	Verify that controller processes requests with null or empty module/version values without error.
Preconditions	- Application running - User has permission to manage modules - ModuleInstaller is available
Test Steps	1. Create a Request object with module=null and version='' (empty string). 2. Mock the Gate to authorize 'manage modules'. 3. Mock ModuleInstaller::complete to return true for (null, ''). 4. Call CompleteModuleInstallationController with the request. 5. Verify response is an instance of JsonResponse. 6. Check the response data is ['success' => true]. 7. Check the HTTP status is 200.
Test Data	- Input: module = null, version = '' - Expected values: JsonResponse, data ['success' => true], status code 200
Expected Result	- Response is a JsonResponse instance. - Response data equals ['success' => true]. - Response status code is 200.
Actual Result	- Response is a JsonResponse instance. - Response data equals ['success' => true]. - Response status code is 200.
Status	Pass
Severity	Medium

Test Case ID	CMIC-005
Title	Throws AuthorizationException if gate authorization fails

Objective	Verify that the controller throws an AuthorizationException when the user is not authorized to manage modules.
Preconditions	- Application running - ModuleInstaller is available
Test Steps	1. Create a Request object with module='any-module' and version='any-version'. 2. Mock the Gate to throw AuthorizationException for 'manage modules'. 3. Ensure ModuleInstaller::complete is not called. 4. Call CompleteModuleInstallationController with the request. 5. Verify that an AuthorizationException is thrown with message 'User not authorized.'
Test Data	- Input: module = 'any-module', version = 'any-version' - Gate should throw AuthorizationException
Expected Result	- AuthorizationException is thrown with message 'User not authorized.' - ModuleInstaller::complete is not called.
Actual Result	- AuthorizationException is thrown with message 'User not authorized.' - ModuleInstaller::complete is not called.
Status	Pass
Severity	High

File: ConfigController-Test.txt

Test Case ID	CC-001
Title	Returns specific config value when the key exists
Objective	To verify that the ConfigController returns the correct value for an existing key in the "crater" config.
Preconditions	- Application running - Database seeded - "crater" config available and modifiable
Test Steps	1. Set the "crater.app_name" config value to "Crater App Test". 2. Create a GET request to the ConfigController with "key" parameter set to "app_name". 3. Invoke the controller with the prepared request. 4. Assert that the response status is HTTP 200 OK. 5. Assert that the returned JSON includes "app_name" => "Crater App Test". 6. Unset the "crater.app_name" config value to clean up.
Test Data	- Input: key = "app_name" - Config set: "crater.app_name" => "Crater App Test" - Expected JSON: {"app_name": "Crater App Test"} - Expected status: HTTP 200
Expected Result	- The response status is HTTP 200. - The returned JSON is {"app_name": "Crater App Test"}.
Actual Result	- The response status was HTTP 200. - The returned JSON was {"app_name": "Crater App Test"}.
Status	Pass
Severity	High

Test Case ID	CC-002
Title	Returns null when the config key does not exist
Objective	To verify that the ConfigController returns null for a non-existent config key.
Preconditions	- Application running - Database seeded - "crater" config available and modifiable
Test Steps	1. Ensure "crater.non_existent_key" is unset in the config. 2. Create a GET request to the ConfigController with "key" parameter set to "non_existent_key". 3. Invoke the controller with the prepared request. 4. Assert that the response status is HTTP 200 OK. 5. Assert that the returned JSON includes "non_existent_key" => null.
Test Data	- Input: key = "non_existent_key" - Config unset: "crater.non_existent_key" - Expected JSON: {"non_existent_key": null} - Expected status: HTTP 200
Expected Result	- The response status is HTTP 200. - The returned JSON is {"non_existent_key": null}.
Actual Result	- The response status was HTTP 200. - The returned JSON was {"non_existent_key": null}.
Status	Pass
Severity	High

Test Case ID	CC-003
Title	Returns null when request key is missing
Objective	To verify that the ConfigController returns null for an empty string key when the "key" request parameter is missing.

Preconditions	- Application running - Database seeded - "crater" config available and modifiable
Test Steps	1. Save the original "crater" config. 2. Set "crater" config to {"email": "test@example.com", "currency": "USD"}. 3. Create a GET request to the ConfigController without the "key" parameter. 4. Invoke the controller with the prepared request. 5. Assert that the response status is HTTP 200 OK. 6. Assert that the returned JSON includes "" => null. 7. Restore the original "crater" config.
Test Data	- Input: key is missing - Config set: {"email": "test@example.com", "currency": "USD"} - Expected JSON: {"": null} - Expected status: HTTP 200
Expected Result	- The response status is HTTP 200. - The returned JSON is {"": null}.
Actual Result	- The response status was HTTP 200. - The returned JSON was {"": null}.
Status	Pass
Severity	High

Test Case ID	CC-004
Title	Returns null when request key is an empty string
Objective	To verify that the ConfigController returns null for an empty string key when the "key" request parameter is present but empty.
Preconditions	- Application running - Database seeded - "crater" config available and modifiable
Test Steps	1. Save the original "crater" config. 2. Set "crater" config to {"language": "en", "timezone": "UTC"}. 3. Create a GET request to the ConfigController with "key" parameter set to empty string (""). 4. Invoke the controller with the prepared request. 5. Assert that the response status is HTTP 200 OK. 6. Assert that the returned JSON includes "" => null. 7. Restore the original "crater" config.
Test Data	- Input: key = "" - Config set: {"language": "en", "timezone": "UTC"} - Expected JSON: {"": null} - Expected status: HTTP 200
Expected Result	- The response status is HTTP 200. - The returned JSON is {"": null}.
Actual Result	- The response status was HTTP 200. - The returned JSON was {"": null}.
Status	Pass
Severity	High

Test Case ID	CC-005
Title	Handles nested config keys correctly
Objective	To verify that the ConfigController returns correct value for a nested config key in the "crater" config.
Preconditions	- Application running - Database seeded - "crater" config available and modifiable

Test Steps	1. Set the "crater.settings.general.app_title" config value to "My Nested App Title". 2. Create a GET request to the ConfigController with "key" parameter set to "settings.general.app_title". 3. Invoke the controller with the prepared request. 4. Assert that the response status is HTTP 200 OK. 5. Assert that the returned JSON includes "settings.general.app_title" => "My Nested App Title". 6. Unset the "crater.settings.general.app_title" config value to clean up.
Test Data	- Input: key = "settings.general.app_title" - Config set: "crater.settings.general.app_title" => "My Nested App Title" - Expected JSON: {"settings.general.app_title": "My Nested App Title"} - Expected status: HTTP 200
Expected Result	- The response status is HTTP 200. - The returned JSON is {"settings.general.app_title": "My Nested App Title"}.
Actual Result	- The response status was HTTP 200. - The returned JSON was {"settings.general.app_title": "My Nested App Title"}.
Status	Pass
Severity	High

Test Case ID	CC-006
Title	Returns complex data structure config value
Objective	To verify that the ConfigController correctly returns a complex array structure as a config value.
Preconditions	- Application running - Database seeded - "crater" config available and modifiable
Test Steps	1. Set the "crater.features" config value to {"darkMode": true, "notifications": {"email": true, "sms": false}}. 2. Create a GET request to the ConfigController with "key" parameter set to "features". 3. Invoke the controller with the prepared request. 4. Assert that the response status is HTTP 200 OK. 5. Assert that the returned JSON includes the correct array structure for "features". 6. Unset the "crater.features" config value to clean up.
Test Data	- Input: key = "features" - Config set: {"darkMode": true, "notifications": {"email": true, "sms": false}} - Expected JSON: {"features": {"darkMode": true, "notifications": {"email": true, "sms": false}}} - Expected status: HTTP 200
Expected Result	- The response status is HTTP 200. - The returned JSON is {"features": {"darkMode": true, "notifications": {"email": true, "sms": false}}}.
Actual Result	- The response status was HTTP 200. - The returned JSON was {"features": {"darkMode": true, "notifications": {"email": true, "sms": false}}}.
Status	Pass
Severity	High

File: ConfirmPasswordController-Test.txt

Test Case ID	CPC-001
Title	ConfirmPasswordController instantiation
Objective	Ensure that the ConfirmPasswordController class can be instantiated successfully.
Preconditions	- Application running - All dependencies autoloaded - Namespace Crater\Http\Controllers\V1\Admin\Auth\ConfirmPasswordController accessible
Test Steps	1. Attempt to instantiate ConfirmPasswordController with no arguments.
Test Data	- Class: Crater\Http\Controllers\V1\Admin\Auth\ConfirmPasswordController
Expected Result	- Instance of ConfirmPasswordController is created without error. - The created object is of type ConfirmPasswordController.
Actual Result	ConfirmPasswordController instantiated successfully; object type matches expected.
Status	Pass
Severity	High

Test Case ID	CPC-002
Title	ConfirmPasswordController trait usage verification
Objective	Verify that ConfirmPasswordController class uses the ConfirmsPasswords trait.
Preconditions	- Application running - All dependencies autoloaded - Namespace Crater\Http\Controllers\V1\Admin\Auth\ConfirmPasswordController accessible
Test Steps	1. Retrieve all traits used by ConfirmPasswordController class using class_uses(). 2. Check for presence of ConfirmsPasswords trait.
Test Data	- Class: Crater\Http\Controllers\V1\Admin\Auth\ConfirmPasswordController - Trait: Illuminate\Foundation\Auth\ConfirmsPasswords
Expected Result	- ConfirmsPasswords trait is present in the list of used traits for ConfirmPasswordController.
Actual Result	ConfirmsPasswords trait detected in ConfirmPasswordController trait list.
Status	Pass
Severity	High

Test Case ID	CPC-003
Title	ConfirmPasswordController redirectTo property value verification
Objective	Validate that the protected $redirectTo property is set to RouteServiceProvider::HOME.
Preconditions	- Application running - All dependencies autoloaded - Namespace Crater\Http\Controllers\V1\Admin\Auth\ConfirmPasswordController accessible - RouteServiceProvider::HOME is defined
Test Steps	1. Instantiate ConfirmPasswordController. 2. Use reflection to access protected $redirectTo property. 3. Retrieve the value of $redirectTo.
Test Data	- Property: protected $redirectTo in ConfirmPasswordController - Expected Value: RouteServiceProvider::HOME

Expected Result	- The value of $redirectTo matches RouteServiceProvider::HOME.
Actual Result	$redirectTo property correctly set to RouteServiceProvider::HOME.
Status	Pass
Severity	High

Test Case ID	CPC-004
Title	ConfirmPasswordController applies auth middleware in constructor
Objective	Ensure that the controller applies the 'auth' middleware in its constructor.
Preconditions	- Application running - All dependencies autoloaded - Namespace Crater\Http\Controllers\V1\Admin\Auth\ConfirmPasswordController accessible
Test Steps	1. Create an anonymous class extending ConfirmPasswordController. 2. Override the middleware method to track calls and arguments. 3. Instantiate the anonymous class to trigger the constructor. 4. Check the number of times middleware() is called. 5. Verify the first and only call uses 'auth' as middleware and empty options array.
Test Data	- Middleware expected: 'auth' - Options expected: []
Expected Result	- middleware() is called exactly once during construction. - Called with 'auth' as middleware. - Called with empty options array.
Actual Result	middleware() called once with 'auth' middleware and no options.
Status	Pass
Severity	High

File: Controller-Test.txt

Test Case ID	C-001
Title	Controller Instantiation
Objective	Verify that the Crater\Http\Controllers\Controller can be instantiated.
Preconditions	- Application running - Crater\Http\Controllers\Controller class available - Necessary dependencies are autoloaded
Test Steps	1. Create an instance of Crater\Http\Controllers\Controller. 2. Verify the created object is an instance of Crater\Http\Controllers\Controller.
Test Data	- Class: Crater\Http\Controllers\Controller
Expected Result	- The object instantiated is of type Crater\Http\Controllers\Controller.
Actual Result	The object was successfully instantiated and is of the expected type.
Status	Pass
Severity	Medium

Test Case ID	C-002
Title	Controller Uses AuthorizesRequests Trait
Objective	Verify that Crater\Http\Controllers\Controller uses the AuthorizesRequests trait.
Preconditions	- Application running - Crater\Http\Controllers\Controller and AuthorizesRequests available
Test Steps	1. Use ReflectionClass on Crater\Http\Controllers\Controller. 2. Retrieve trait names used by the class. 3. Check that AuthorizesRequests trait is listed among used traits.
Test Data	- Class: Crater\Http\Controllers\Controller - Trait: Illuminate\Foundation\Auth\Access\AuthorizesRequests
Expected Result	- AuthorizesRequests trait is present in the list of traits used by Controller.
Actual Result	AuthorizesRequests trait is present in the class trait list.
Status	Pass
Severity	High

Test Case ID	C-003
Title	Controller Uses DispatchesJobs Trait
Objective	Verify that Crater\Http\Controllers\Controller uses the DispatchesJobs trait.
Preconditions	- Application running - Crater\Http\Controllers\Controller and DispatchesJobs available
Test Steps	1. Use ReflectionClass on Crater\Http\Controllers\Controller. 2. Retrieve trait names used by the class. 3. Check that DispatchesJobs trait is listed among used traits.
Test Data	- Class: Crater\Http\Controllers\Controller - Trait: Illuminate\Foundation\Bus\DispatchesJobs
Expected Result	- DispatchesJobs trait is present in the list of traits used by Controller.
Actual Result	DispatchesJobs trait is present in the class trait list.
Status	Pass
Severity	High

Test Case ID	C-004

Title	Controller Uses ValidatesRequests Trait
Objective	Verify that Crater\Http\Controllers\Controller uses the ValidatesRequests trait.
Preconditions	- Application running - Crater\Http\Controllers\Controller and ValidatesRequests available
Test Steps	1. Use ReflectionClass on Crater\Http\Controllers\Controller. 2. Retrieve trait names used by the class. 3. Check that ValidatesRequests trait is listed among used traits.
Test Data	- Class: Crater\Http\Controllers\Controller - Trait: Illuminate\Foundation\Validation\ValidatesRequests
Expected Result	- ValidatesRequests trait is present in the list of traits used by Controller.
Actual Result	ValidatesRequests trait is present in the class trait list.
Status	Pass
Severity	High

Test Case ID	C-005
Title	Controller Extends Illuminate's BaseController
Objective	Verify that Crater\Http\Controllers\Controller extends Illuminate\Routing\Controller.
Preconditions	- Application running - Crater\Http\Controllers\Controller and Illuminate\Routing\Controller available
Test Steps	1. Use ReflectionClass on Crater\Http\Controllers\Controller. 2. Retrieve the parent class of Controller. 3. Confirm that the parent class is Illuminate\Routing\Controller.
Test Data	- Class: Crater\Http\Controllers\Controller - Parent Class: Illuminate\Routing\Controller
Expected Result	- Controller's parent class is Illuminate\Routing\Controller.
Actual Result	Parent class is Illuminate\Routing\Controller as expected.
Status	Pass
Severity	Medium

File: CopyFilesController-Test.txt

Test Case ID	CFC-001
Title	Unauthorized access returns 401 when no user is authenticated
Objective	Verify that the CopyFilesController returns a 401 Unauthorized response when the request does not have an authenticated user.
Preconditions	- Application running - Database seeded (if authentication relies on user records) - No user is authenticated in the system
Test Steps	1. Create a mock request object. 2. Configure the request object's user() method to return null (simulating no authenticated user). 3. Instantiate the CopyFilesController. 4. Invoke the controller with the request. 5. Observe the response object returned.
Test Data	- Input: No authenticated user - Expected values: - Response type: JsonResponse - Status code: 401 - success: false - message: 'You are not allowed to update this app.'
Expected Result	- The response is an instance of JsonResponse. - The response has a status code of 401. - The 'success' field in response data is false. - The 'message' field in response data is 'You are not allowed to update this app.'
Actual Result	- The response matched all expectations: returned JsonResponse with 401, success=false, and appropriate message.
Status	Pass
Severity	High

Test Case ID	CFC-002
Title	Unauthorized access returns 401 when authenticated user is not an owner
Objective	Verify that CopyFilesController returns a 401 Unauthorized response for authenticated users who are not owners.
Preconditions	- Application running - Database seeded with users - Authenticated user exists but is not designated as owner
Test Steps	1. Create a mock user object. 2. Configure the user's isOwner() method to return false. 3. Create a mock request object. 4. Configure the request object's user() method to return the mock user. 5. Instantiate the CopyFilesController. 6. Invoke the controller with the request. 7. Observe the response object returned.
Test Data	- Input: Authenticated user (isOwner returns false) - Expected values: - Response type: JsonResponse - Status code: 401 - success: false - message: 'You are not allowed to update this app.'
Expected Result	- The response is an instance of JsonResponse. - The response has a status code of 401. - The 'success' field in response data is false. - The 'message' field in response data is 'You are not allowed to update this app.'

Actual Result	- The response matched all expectations: returned JsonResponse with 401, success=false, and appropriate message.
Status	Pass
Severity	High

Test Case ID	CFC-003
Title	Validation exception is thrown if required path parameter is missing
Objective	Confirm that a ValidationException is thrown if the required 'path' parameter is absent in the request.
Preconditions	- Application running - Database seeded with owner users - Authenticated user identified as owner - Request missing 'path' parameter
Test Steps	1. Create a mock user object. 2. Configure the user's isOwner() method to return true. 3. Create a mock request object. 4. Configure the request object's user() method to return the mock user. 5. Configure the request object's validate() method to expect ['path' => 'required'] and throw ValidationException. 6. Instantiate the CopyFilesController. 7. Invoke the controller with the request, expecting an exception. 8. Observe exception handling.
Test Data	- Input: request missing 'path', validate(['path' => 'required']) throws ValidationException - Expected values: - Exception: ValidationException
Expected Result	- A ValidationException is thrown when the path parameter is missing.
Actual Result	- ValidationException was thrown as expected when path was not present in the request.
Status	Pass
Severity	High

Test Case ID	CFC-004
Title	Successful file copy operation returns destination path in response
Objective	Verify that CopyFilesController copies files when given a valid path and owner user, returning destination path and success message.
Preconditions	- Application running - Database seeded with owner users - Authenticated user identified as owner - Source and destination paths exist and are accessible
Test Steps	1. Create a mock user object. 2. Configure the user's isOwner() method to return true. 3. Create a mock request object. 4. Configure the request object's user() method to return the mock user. 5. Set dummyPath = '/path/to/source', copiedPath = '/path/to/destination'. 6. Configure the request object's validate() method to expect ['path' => 'required'] and return ['path' => dummyPath]. 7. Attach path property to request object with value dummyPath. 8. Mock Crater\Space\Updater's copyFiles(dummyPath) method to return copiedPath once. 9. Instantiate the CopyFilesController. 10. Invoke the controller with the request. 11. Observe the response object returned.

Test Data	- Input: - User is owner - Valid path: '/path/to/source' - Updater returns: '/path/to/destination' - Expected values: - Response type: JsonResponse - Status code: 200 - success: true - path: '/path/to/destination'
Expected Result	- The response is an instance of JsonResponse. - The response has a status code of 200. - The 'success' field in response data is true. - The 'path' field in response data is '/path/to/destination'.
Actual Result	- The response matched all expectations: returned JsonResponse with 200, success=true, and expected destination path.
Status	Pass
Severity	High

File: CopyModuleController-Test.txt

Test Case ID	CMC-001
Title	Invoke controller with successful module copy
Objective	Verify that the CopyModuleController invokes successfully, authorizes, calls copy files, and returns correct JSON response on successful copy.
Preconditions	- Application running - Database seeded - CopyModuleController and dependencies loaded - Gate allows 'manage modules'
Test Steps	1. Create a TestRequest with module='test-module-name' and path='/var/www/html/crater/modules'. 2. Instantiate TestCopyModuleController. 3. Mock Gate to allow authorization for 'manage modules'. 4. Invoke controller with test request. 5. Assert authorize is called, ability is 'manage modules', copy files is called, parameters match, response is JsonResponse, status code is 200, and JSON response is {'success': true}.
Test Data	- Request: { module: 'test-module-name', path: '/var/www/html/crater/modules' } - Expected ability: 'manage modules' - Expected response: { success: true }
Expected Result	- Authorize is called - Ability is 'manage modules' - copyFiles is called with correct params - Response is a 200 JsonResponse with { success: true }
Actual Result	- All conditions met; response JSON is { success: true }
Status	Pass
Severity	High

Test Case ID	CMC-002
Title	Controller handles failed copy operation
Objective	Verify controller returns JSON response { success: false } when copy fails.
Preconditions	- Application running - Database seeded - CopyModuleController and dependencies loaded - Gate allows 'manage modules'
Test Steps	1. Create a TestRequest with module='another-module' and path='/app/data/modules'. 2. Instantiate TestCopyModuleController and set copyFilesResult=false. 3. Mock Gate to allow authorization. 4. Invoke controller with test request. 5. Assert authorize and copyFiles are called, parameters match, response is JsonResponse, status is 200, JSON is { success: false }.
Test Data	- Request: { module: 'another-module', path: '/app/data/modules' } - copyFilesResult: false
Expected Result	- Authorize is called - copyFiles is called with provided params - Response is a 200 JsonResponse with { success: false }
Actual Result	- All conditions met; response JSON is { success: false }
Status	Pass
Severity	High

Test Case ID	CMC-003

Title	Controller throws AuthorizationException when unauthorized
Objective	Verify that the controller throws an AuthorizationException if Gate denies access.
Preconditions	- Application running - Database seeded - CopyModuleController and dependencies loaded - Gate denies 'manage modules'
Test Steps	1. Create a TestRequest with module='unauthorized-module' and path='/tmp/modules'. 2. Instantiate TestCopyModuleController. 3. Mock Gate to deny authorization for 'manage modules'. 4. Invoke controller and catch exception. 5. Assert AuthorizationException is thrown with correct message and copyFiles is not called.
Test Data	- Request: { module: 'unauthorized-module', path: '/tmp/modules' }
Expected Result	- AuthorizationException thrown with message 'Unauthorized to manage modules.' - copyFiles not called
Actual Result	- AuthorizationException thrown; copyFiles not called
Status	Pass
Severity	High

Test Case ID	CMC-004
Title	Controller handles null parameters gracefully
Objective	Ensure controller can process requests with null module and path without errors.
Preconditions	- Application running - Database seeded - CopyModuleController and dependencies loaded - Gate allows 'manage modules'
Test Steps	1. Create a TestRequest with module=null and path=null. 2. Instantiate TestCopyModuleController. 3. Mock Gate to allow authorization. 4. Invoke controller with request. 5. Check copyFilesParams are [null, null]. 6. Assert response is JSON with { success: true }.
Test Data	- Request: { module: null, path: null }
Expected Result	- copyFilesParams are [null, null] - Response JSON is { success: true }
Actual Result	- Params are [null, null]; JSON response is { success: true }
Status	Pass
Severity	Medium

Test Case ID	CMC-005
Title	Controller handles empty string parameters
Objective	Ensure controller processes requests with empty string values for module and path.
Preconditions	- Application running - Database seeded - CopyModuleController and dependencies loaded - Gate allows 'manage modules'

Test Steps	1. Create a TestRequest with module='' and path=''. 2. Instantiate TestCopyModuleController and set copyFilesResult=false. 3. Mock Gate to allow authorization. 4. Invoke controller with request. 5. Check copyFilesParams are ['', '']. 6. Assert response JSON is { success: false }.
Test Data	- Request: { module: '', path: '' } - copyFilesResult: false
Expected Result	- copyFilesParams are ['', ''] - Response JSON is { success: false }
Actual Result	- Params are ['', '']; JSON response is { success: false }
Status	Pass
Severity	Medium

Test Case ID	CMC-006
Title	Controller processes different path formats
Objective	Verify controller correctly handles various path formats (absolute, relative, parent, current).
Preconditions	- Application running - Database seeded - CopyModuleController loaded - Gate allows 'manage modules'
Test Steps	For each test case: 1. Create a TestRequest with selected module and path format. 2. Instantiate TestCopyModuleController and reset state. 3. Invoke controller with request. 4. Assert copyFilesParams match input. 5. Assert response JSON is { success: true }.
Test Data	- Cases: - { module: 'module1', path: '/absolute/path' } - { module: 'module2', path: 'relative/path' } - { module: 'module3', path: '../parent/path' } - { module: 'module4', path: './current/path' }
Expected Result	- copyFilesParams match request - Response JSON is { success: true } for each case
Actual Result	- Params and responses correct in each iteration
Status	Pass
Severity	Medium

Test Case ID	CMC-007
Title	Controller response JSON structure verification
Objective	Ensure that controller's response is a proper JSON object with only the 'success' key.
Preconditions	- Application running - CopyModuleController loaded - Gate allows 'manage modules'
Test Steps	1. Create TestRequest with module='test-module' and path='/test/path'. 2. Instantiate TestCopyModuleController. 3. Mock Gate to allow. 4. Invoke controller. 5. Extract response data and check it is an array, has only 'success' key, value is boolean.
Test Data	- Request: { module: 'test-module', path: '/test/path' }

Expected Result	- Response data is array - Contains only 'success' key (boolean) - No extra keys present
Actual Result	- Response array with only 'success' key, value is boolean
Status	Pass
Severity	Medium

Test Case ID	CMC-008
Title	Controller inherits from base Controller
Objective	Ensure CopyModuleController inherits from Crater\Http\Controllers\Controller.
Preconditions	- Application running - Class definitions loaded
Test Steps	1. Instantiate CopyModuleController. 2. Assert instance is of Crater\Http\Controllers\Controller.
Test Data	- None
Expected Result	- CopyModuleController is instance of Crater\Http\Controllers\Controller
Actual Result	- Inheritance confirmed
Status	Pass
Severity	Low

Test Case ID	CMC-009
Title	Controller uses __invoke single action pattern
Objective	Verify controller uses Laravel's single action controller pattern (has __invoke and is callable).
Preconditions	- Application running - CopyModuleController loaded
Test Steps	1. Instantiate CopyModuleController. 2. Assert __invoke method exists and instance is callable.
Test Data	- None
Expected Result	- __invoke method exists - Controller is callable
Actual Result	- __invoke exists; controller is callable
Status	Pass
Severity	Low

Test Case ID	CMC-010
Title	Controller request validation with diverse data types
Objective	Ensure controller accepts and processes various data types in module and path request parameters.
Preconditions	- Application running - CopyModuleController loaded - Gate allows 'manage modules'
Test Steps	For each test case: 1. Create TestRequest with different types (numeric, boolean, array, object). 2. Reset controller state. 3. Invoke controller. 4. Assert copyFiles is called. 5. Assert response JSON is { success: true }.

Test Data	- Cases: - { module: 123, path: 456 } - { module: true, path: false } - { module: ['array'], path: ['another'] } - { module: (object), path: (object) }
Expected Result	- Controller handles and responds with { success: true } for all types
Actual Result	- Controller accepts data and responds with { success: true }
Status	Pass
Severity	Medium

Test Case ID	CMC-011
Title	Controller handles very long module name
Objective	Verify controller can process requests with extremely long module names.
Preconditions	- Application running - CopyModuleController loaded - Gate allows 'manage modules'
Test Steps	1. Generate a string with 1000 'a' characters for module. 2. Create TestRequest with long module name and path='/path'. 3. Instantiate controller. 4. Invoke controller. 5. Assert copyFilesParams[0] length is 1000 and matches input. 6. Assert response JSON is { success: true }.
Test Data	- Request: { module: 'a...a' (1000 chars), path: '/path' }
Expected Result	- Module name in params is 1000 chars - Response JSON is { success: true }
Actual Result	- Params and response correct
Status	Pass
Severity	Medium

Test Case ID	CMC-012
Title	Controller handles special characters in parameters
Objective	Ensure controller processes module and path parameters containing special characters.
Preconditions	- Application running - CopyModuleController loaded - Gate allows 'manage modules'
Test Steps	1. Create TestRequest with special characters in module/path. 2. Instantiate controller. 3. Invoke controller. 4. Assert copyFilesParams match input strings. 5. Assert response JSON is { success: true }.
Test Data	- Request: { module: 'module-with-special-@#$%^&*()-chars', path: '/path/with/special/@#$%^&*()/chars' }
Expected Result	- Params match input strings - Response JSON is { success: true }
Actual Result	- Special chars preserved and response is { success: true }
Status	Pass
Severity	Medium

Test Case ID	CMC-013
Title	Controller JSON response encoding validation

Objective	Ensure controller responses use JSON encoding with correct Content-Type header and decodable body.
Preconditions	- Application running - CopyModuleController loaded - Gate allows 'manage modules'
Test Steps	1. Create TestRequest with module='json-test' and path='/json/path'. 2. Instantiate controller. 3. Invoke controller. 4. Assert Content-Type header contains 'application/json'. 5. Assert JSON-encoded content matches { success: true }.
Test Data	- Request: { module: 'json-test', path: '/json/path' }
Expected Result	- Response Content-Type is 'application/json' - Response decodes to { success: true }
Actual Result	- Content-Type and JSON encoding correct
Status	Pass
Severity	Medium

Test Case ID	CMC-014
Title	Actual controller structure matches expectations
Objective	Inspect controller for correct public __invoke method with Request parameter.
Preconditions	- Application running - CopyModuleController definition available
Test Steps	1. Use reflection to inspect CopyModuleController class. 2. Check for public __invoke method. 3. Assert method has exactly one parameter of Illuminate\Http\Request type.
Test Data	- None
Expected Result	- __invoke method exists, is public, accepts Request parameter
Actual Result	- Method signature is correct and public
Status	Pass
Severity	Low

Test Case ID	CMC-015
Title	Controller follows Laravel conventions
Objective	Ensure controller is in expected namespace and named correctly.
Preconditions	- Application running - CopyModuleController definition available
Test Steps	1. Instantiate CopyModuleController. 2. Assert class name matches expectation. 3. Assert namespace begins with 'Crater\Http\Controllers\V1\Admin\Modules'.
Test Data	- None
Expected Result	- Controller has correct class name and namespace
Actual Result	- Class name and namespace as expected
Status	Pass
Severity	Low

File: CountriesController-Test.txt

Test Case ID	CC-001
Title	Returns a collection of countries when countries exist
Objective	Verify that the CountriesController returns a collection of country resources when countries exist in the database.
Preconditions	- Application running - Database seeded with country records (USA and Canada) - Required classes are properly autoloaded - Mockery is available for mocking methods
Test Steps	1. Seed the database or mock country records with: - USA (id: 1, name: 'USA', code: 'US') - Canada (id: 2, name: 'Canada', code: 'CA') 2. Mock the Country::all() static method to return the above collection. 3. Mock CountryResource::collection() to ensure it receives the correct collection and returns a mock resource collection. 4. Instantiate CountriesController. 5. Create a new HTTP request. 6. Call the CountriesController with the request. 7. Assert that the result is an instance of AnonymousResourceCollection. 8. Assert that the result equals the mocked resource collection.
Test Data	- Input: Collection containing: - (id: 1, name: 'USA', code: 'US') - (id: 2, name: 'Canada', code: 'CA') - Mocked output: Instance of AnonymousResourceCollection as returned by CountryResource::collection()
Expected Result	- The result is an instance of AnonymousResourceCollection. - The returned collection is exactly the mocked collection provided by CountryResource::collection().
Actual Result	- The result was an instance of AnonymousResourceCollection and matched the mocked resource collection.
Status	Pass
Severity	Medium

Test Case ID	CC-002
Title	Returns an empty collection when no countries exist
Objective	Verify that the CountriesController returns an empty collection of country resources when no countries exist in the database.
Preconditions	- Application running - Database seeded with no country records (empty) - Required classes are properly autoloaded - Mockery is available for mocking methods
Test Steps	1. Ensure the database contains no country records, or mock Country::all() to return an empty collection. 2. Mock CountryResource::collection() to ensure it receives an empty collection and returns a mock empty resource collection. 3. Instantiate CountriesController. 4. Create a new HTTP request. 5. Call the CountriesController with the request. 6. Assert that the result is an instance of AnonymousResourceCollection. 7. Assert that the result equals the mocked empty resource collection.
Test Data	- Input: Empty collection - Mocked output: Instance of AnonymousResourceCollection as returned by CountryResource::collection() for empty input
Expected Result	- The result is an instance of AnonymousResourceCollection. - The returned collection is exactly the mocked empty resource collection provided by CountryResource::collection().

Actual Result	- The result was an instance of AnonymousResourceCollection and matched the mocked empty resource collection.
Status	Pass
Severity	Medium

File: CreateBackupJob-Test.txt

Test Case ID	CBJ-001
Title	CreateBackupJob constructor stores arbitrary array data
Objective	Verify that the CreateBackupJob constructor correctly stores data when provided with an array.
Preconditions	- Application running - Required classes (CreateBackupJob, ReflectionClass) available - No database precondition
Test Steps	1. Create an array $data = ['test' => 'data']. 2. Instantiate CreateBackupJob with $data. 3. Use ReflectionClass to access the private 'data' property from the job instance. 4. Assert that the 'data' property equals the input $data.
Test Data	- Input data: ['test' => 'data'] - Expected value: 'data' property equals ['test' => 'data']
Expected Result	- The 'data' property in CreateBackupJob instance is set to ['test' => 'data'].
Actual Result	The 'data' property was correctly set to ['test' => 'data'].
Status	Pass
Severity	Medium

Test Case ID	CBJ-002
Title	CreateBackupJob constructor handles empty string input
Objective	Verify that the CreateBackupJob constructor correctly handles and stores an empty string.
Preconditions	- Application running - Required classes available
Test Steps	1. Instantiate CreateBackupJob with input ''. 2. Use ReflectionClass to access the private 'data' property. 3. Assert that the 'data' property is ''.
Test Data	- Input data: '' - Expected value: 'data' property equals ''
Expected Result	- The 'data' property in CreateBackupJob instance is set to an empty string.
Actual Result	The 'data' property was correctly set to ''.
Status	Pass
Severity	Medium

Test Case ID	CBJ-003
Title	CreateBackupJob constructor handles null input
Objective	Verify that the constructor stores an empty string when called without arguments.
Preconditions	- Application running - Required classes available
Test Steps	1. Instantiate CreateBackupJob with no arguments. 2. Use ReflectionClass to access 'data'. 3. Assert that 'data' is ''.
Test Data	- Input data: null (no argument) - Expected value: 'data' property equals ''
Expected Result	- The 'data' property is set to '' when no argument is provided.
Actual Result	The 'data' property was correctly set to ''.

Status	Pass
Severity	Medium

Test Case ID	CBJ-004
Title	CreateBackupJob constructor stores array data with multiple keys
Objective	Ensure the constructor correctly stores complex array data.
Preconditions	- Application running
Test Steps	1. Create $data = ['key1' => 'value1', 'key2' => 'value2']. 2. Instantiate CreateBackupJob with $data. 3. Use reflection to read 'data'. 4. Assert that 'data' equals $data.
Test Data	- Input data: ['key1' => 'value1', 'key2' => 'value2'] - Expected value: 'data' property matches input array
Expected Result	- 'data' property is equal to ['key1' => 'value1', 'key2' => 'value2']
Actual Result	The 'data' property was correctly set.
Status	Pass
Severity	Medium

Test Case ID	CBJ-005
Title	Handle method works with default option
Objective	Verify the handle() method correctly executes backup logic with default option.
Preconditions	- Application running - TestHelper and TestCreateBackupJob available - FileDisk instance with id=1 and driver='local' mocked
Test Steps	1. Reset TestHelper state. 2. Mock FileDisk instance with id=1, driver='local'. 3. Prepare $data = ['file_disk_id' => 1, 'option' => '']. 4. Instantiate TestCreateBackupJob with $data. 5. Call handle() on the job. 6. Retrieve the FileDisk instance. 7. Assert handle was called, setConfig called, and configSet includes correct disk.
Test Data	- Input: ['file_disk_id' => 1, 'option' => ''] - Expected disk: ['temp_local']
Expected Result	- handleCalled property is true - setConfigCalled property of FileDisk is true - configSet contains ['backup.backup.destination.disks' => ['temp_local']]
Actual Result	All expected properties were true and config was set as expected.
Status	Pass
Severity	High

Test Case ID	CBJ-006
Title	Handle method works with 'only-db' option
Objective	Ensure only database backup is performed when 'only-db' option is given.
Preconditions	- Application running - TestHelper reset - FileDisk with id=2, driver='s3' mocked

Test Steps	1. Reset TestHelper. 2. Mock FileDisk with id=2, driver='s3'. 3. Prepare $data = ['file_disk_id' => 2, 'option' => 'only-db']. 4. Instantiate TestCreateBackupJob. 5. Call handle(). 6. Retrieve backupJob from TestHelper. 7. Assert dontBackupFilesystemCalled is true. 8. Assert filenameSet matches regex /^only-db-[timestamp].zip$/ 9. Assert runCalled is true.
Test Data	- Input: ['file_disk_id' => 2, 'option' => 'only-db']
Expected Result	- dontBackupFilesystemCalled property is true - filenameSet matches pattern - runCalled is true
Actual Result	All properties were set as expected; filename matched regex.
Status	Pass
Severity	High

Test Case ID	CBJ-007
Title	Handle method works with 'only-files' option
Objective	Ensure only file backup is performed when 'only-files' option is given.
Preconditions	- Application running - FileDisk mocked with id=3, driver='dropbox'
Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=3, driver='dropbox'. 3. Prepare $data with 'option' => 'only-files'. 4. Instantiate and handle job. 5. Get backupJob. 6. Assert dontBackupDatabasesCalled is true. 7. Assert filenameSet matches regex /^only-files-[timestamp].zip$/ 8. Assert runCalled is true.
Test Data	- Input: ['file_disk_id' => 3, 'option' => 'only-files']
Expected Result	- dontBackupDatabasesCalled is true - filename pattern matches - runCalled is true
Actual Result	Properties and filename set as expected.
Status	Pass
Severity	High

Test Case ID	CBJ-008
Title	Handle method works with custom option
Objective	Verify custom option name sets proper filename and does not skip backup types.
Preconditions	- Application running - FileDisk mocked with id=4, driver='ftp'
Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=4, driver='ftp'. 3. Prepare $data with 'option' => 'custom_backup'. 4. Instantiate and handle job. 5. Get backupJob. 6. Assert filenameSet matches /^custom-backup-[timestamp].zip$/ 7. Assert runCalled is true. 8. Assert dontBackupFilesystemCalled is false. 9. Assert dontBackupDatabasesCalled is false.
Test Data	- Input: ['file_disk_id' => 4, 'option' => 'custom_backup']

Expected Result	- filenameSet matches pattern - runCalled is true - dontBackupFilesystemCalled is false - dontBackupDatabasesCalled is false
Actual Result	All expected properties and filename confirmed.
Status	Pass
Severity	High

Test Case ID	CBJ-009
Title	Handle method with option containing underscores
Objective	Ensure option containing underscores is correctly converted in filename.
Preconditions	- Application running - FileDisk mocked with id=5, driver='local'
Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=5, driver='local'. 3. Prepare $data: ['file_disk_id' => 5, 'option' => 'test_backup_name']. 4. Instantiate and handle job. 5. Retrieve backupJob. 6. Assert filenameSet matches /^test-backup-name-[timestamp].zip$/
Test Data	- Input: ['file_disk_id' => 5, 'option' => 'test_backup_name']
Expected Result	- filenameSet matches /^test-backup-name-[timestamp].zip$/
Actual Result	Filename matched expected pattern.
Status	Pass
Severity	High

Test Case ID	CBJ-010
Title	Handle method when option is missing
Objective	Ensure handle processes backup when 'option' is missing in the input data.
Preconditions	- Application running - FileDisk mocked with id=6, driver='s3'
Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=6, driver='s3'. 3. Prepare $data without 'option' key. 4. Instantiate and handle job. 5. Retrieve backupJob. 6. Assert runCalled is true. 7. Assert filenameSet is null.
Test Data	- Input: ['file_disk_id' => 6]
Expected Result	- runCalled is true - filenameSet is null
Actual Result	runCalled was true; filenameSet remained null.
Status	Pass
Severity	High

Test Case ID	CBJ-011
Title	Handle method uses default disk prefix when env not set
Objective	Ensure default prefix ('temp_') is used when DYNAMIC_DISK_PREFIX environment variable is not set.
Preconditions	- Application running - TestHelper::$envValue is null

Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=7, driver='local'. 3. Ensure TestHelper::$envValue is null. 4. Prepare $data with 'option' => ''. 5. Instantiate and handle job. 6. Assert that configSet contains ['backup.backup.destination.disks' => ['temp_local']].
Test Data	- Input: ['file_disk_id' => 7, 'option' => ''] - Expected disk: ['temp_local']
Expected Result	- configSet includes ['backup.backup.destination.disks' => ['temp_local']]
Actual Result	Configuration set to ['temp_local'] when environment variable is null.
Status	Pass
Severity	Medium

Test Case ID	CBJ-012
Title	Handle method uses custom disk prefix from environment variable
Objective	Ensure custom disk prefix is used from DYNAMIC_DISK_PREFIX environment variable.
Preconditions	- Application running - TestHelper::$envValue set to 'custom_prefix_'
Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=8, driver='s3'. 3. Set TestHelper::$envValue to 'custom_prefix_'. 4. Prepare $data with 'option' => ''. 5. Instantiate and handle job. 6. Assert configSet contains ['custom_prefix_s3'].
Test Data	- Input: ['file_disk_id' => 8, 'option' => ''] - Expected disk: ['custom_prefix_s3']
Expected Result	- configSet contains ['custom_prefix_s3']
Actual Result	Configuration reflected custom prefix.
Status	Pass
Severity	Medium

Test Case ID	CBJ-013
Title	Handle method does not set filename when option is empty
Objective	Ensure that an empty 'option' does not result in a filename being set for the backup job.
Preconditions	- Application running - FileDisk id=10, driver='local' mocked
Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=10, driver='local'. 3. Prepare $data with 'option' => ''. 4. Instantiate and handle job. 5. Get backupJob. 6. Assert filenameSet is null.
Test Data	- Input: ['file_disk_id' => 10, 'option' => '']
Expected Result	- filenameSet remains null
Actual Result	filenameSet confirmed null.
Status	Pass
Severity	Medium

Test Case ID	CBJ-014

Title	Handle method does not set filename when option is not set
Objective	Ensure filename is not set when 'option' key is absent in the data.
Preconditions	- Application running - FileDisk id=11, driver='s3' mocked
Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=11, driver='s3'. 3. Prepare $data without 'option'. 4. Instantiate and handle job. 5. Get backupJob. 6. Assert filenameSet is null.
Test Data	- Input: ['file_disk_id' => 11]
Expected Result	- filenameSet remains null
Actual Result	filenameSet confirmed null.
Status	Pass
Severity	Medium

Test Case ID	CBJ-015
Title	CreateBackupJob implements ShouldQueue interface
Objective	Ensure CreateBackupJob implements the Laravel ShouldQueue contract.
Preconditions	- Application running
Test Steps	1. Instantiate CreateBackupJob. 2. Assert job is instance of \Illuminate\Contracts\Queue\ShouldQueue.
Test Data	- Input: none - Expected type: ShouldQueue
Expected Result	- CreateBackupJob is an instance of ShouldQueue
Actual Result	Instance confirmed as ShouldQueue.
Status	Pass
Severity	High

Test Case ID	CBJ-016
Title	CreateBackupJob uses Laravel queue traits
Objective	Ensure the class utilizes standard Laravel job traits.
Preconditions	- Application running
Test Steps	1. Instantiate CreateBackupJob. 2. Get list of traits used by object. 3. Assert traits contain: Dispatchable, InteractsWithQueue, Queueable, SerializesModels.
Test Data	- Input: none
Expected Result	- All required traits are present in the class
Actual Result	All traits confirmed present.
Status	Pass
Severity	High

Test Case ID	CBJ-017
Title	Handle method with very long option name
Objective	Verify that handle correctly sets a long option as a prefix in the filename.
Preconditions	- Application running - FileDisk id=12, driver='local' mocked

Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=12, driver='local'. 3. Generate $longOption as str_repeat('a', 50) . '_' . str_repeat('b', 50). 4. Prepare $data: ['file_disk_id' => 12, 'option' => $longOption]. 5. Instantiate and handle job. 6. Check backupJob.filenameSet starts with $expectedPrefix and matches regex.
Test Data	- Input: $longOption - Expected filename prefix: 50 'a's, dash, 50 'b's, dash
Expected Result	- filenameSet matches /^aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb-\d{4}-\d{2}-\d{2}-\d{2}-\d{2}-\d{2}\.zip$/
Actual Result	Filename successfully matched expected pattern.
Status	Pass
Severity	Medium

Test Case ID	CBJ-018
Title	Handle method with special characters in option
Objective	Confirm special characters are preserved in backup filename.
Preconditions	- Application running - FileDisk id=13, driver='s3' mocked
Test Steps	1. Reset TestHelper. 2. Mock FileDisk id=13, driver='s3'. 3. Prepare $data with 'option' => 'test@backup#name'. 4. Instantiate and handle job. 5. Retrieve backupJob. 6. Assert filenameSet matches /^test@backup#name-[timestamp].zip$/
Test Data	- Input: 'test@backup#name'
Expected Result	- filenameSet retains special characters, matches regex /^test@backup#name-[timestamp].zip$/
Actual Result	Filename confirmed contains special characters and matches pattern.
Status	Pass
Severity	Medium

File: CronJobController-Test.txt

Test Case ID	CJC-001
Title	CronJobController invokes Artisan schedule:run and returns success response
Objective	Verify that the CronJobController calls the Artisan schedule:run command exactly once and returns a valid JSON HTTP 200 response indicating success.
Preconditions	- Application running - Database seeded - CronJobController available - Artisan facade mock configured to intercept static calls
Test Steps	1. Replace the default Artisan facade instance with a mock object. 2. Configure the mock to expect the 'call' method with 'schedule:run' argument to be called once and return 0. 3. Instantiate the CronJobController. 4. Create a dummy Request object. 5. Invoke the controller's __invoke method with the dummy Request. 6. Verify that the mock's 'call' method was invoked as expected. 7. Assert the HTTP response status code is 200. 8. Assert that the response content is valid JSON. 9. Decode the response JSON and check if it equals ['success' => true].
Test Data	- Request: Empty HTTP Request object - Artisan mock expectation: 'call' method with argument 'schedule:run'; returns 0 - Expected JSON: {'success': true}
Expected Result	- The Artisan mock's 'call' method is called exactly once with 'schedule:run'. - The HTTP response status code is 200. - The response content is valid JSON. - The decoded JSON response equals {'success': true}.
Actual Result	- The Artisan mock's 'call' method was called once with 'schedule:run'. - HTTP response status code was 200. - Response content was valid JSON. - Decoded JSON response equaled {'success': true}.
Status	Pass
Severity	High

File: CronJobMiddleware-Test.txt

Test Case ID	CJM-001
Title	Allow access when correct authorization token is present in header
Objective	To verify that CronJobMiddleware allows access when the request contains the correct authorization token matching the configured token.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'super_secret_cron_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'super_secret_cron_token'. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'super_secret_cron_token' in the request. 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = 'super_secret_cron_token'
Expected Result	- Response should be 'allowed_response_from_controller'
Actual Result	Response is 'allowed_response_from_controller'
Status	Pass
Severity	High

Test Case ID	CJM-002
Title	Deny access when authorization token header is missing
Objective	To verify that CronJobMiddleware denies access if the request does not contain the authorization token header.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'super_secret_cron_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'super_secret_cron_token'. 2. Create a GET request to '/' without the 'x-authorization-token' header. 3. Pass request through CronJobMiddleware. 4. Observe response.
Test Data	- Header: x-authorization-token not set
Expected Result	- Response is instance of JsonResponse - Response status code is 401 - Response data is ['unauthorized']
Actual Result	Response is JsonResponse with status 401 and data ['unauthorized']
Status	Pass
Severity	High

Test Case ID	CJM-003
Title	Deny access when authorization token is incorrect
Objective	To verify that CronJobMiddleware denies access when the request token does not match the configured token.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'correct_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'correct_token'. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'incorrect_token' in the request. 4. Pass request through CronJobMiddleware. 5. Observe response.

Test Data	- Header: x-authorization-token = 'incorrect_token'
Expected Result	- Response is instance of JsonResponse - Response status code is 401 - Response data is ['unauthorized']
Actual Result	Response is JsonResponse with status 401 and data ['unauthorized']
Status	Pass
Severity	High

Test Case ID	CJM-004
Title	Deny access when header token is an empty string
Objective	To verify that CronJobMiddleware denies access if the token header exists but is an empty string.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'super_secret_cron_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'super_secret_cron_token'. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to '' (empty string). 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = ''
Expected Result	- Response is instance of JsonResponse - Response status code is 401 - Response data is ['unauthorized']
Actual Result	Response is JsonResponse with status 401 and data ['unauthorized']
Status	Pass
Severity	High

Test Case ID	CJM-005
Title	Deny access when config token is empty string and header token is present
Objective	To verify that CronJobMiddleware denies access if the configured token is set to empty but a non-empty header token is provided.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to ''
Test Steps	1. Set config 'services.cron_job.auth_token' to '' (empty string). 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'some_token_from_request'. 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = 'some_token_from_request'
Expected Result	- Response is instance of JsonResponse - Response status code is 401 - Response data is ['unauthorized']
Actual Result	Response is JsonResponse with status 401 and data ['unauthorized']
Status	Pass
Severity	High

Test Case ID	CJM-006
Title	Deny access when config token is null and header token is present

Objective	To verify that CronJobMiddleware denies access if the configured token is null and request contains a token in the header.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to null
Test Steps	1. Set config 'services.cron_job.auth_token' to null. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'some_token_from_request'. 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = 'some_token_from_request'
Expected Result	- Response is instance of JsonResponse - Response status code is 401 - Response data is ['unauthorized']
Actual Result	Response is JsonResponse with status 401 and data ['unauthorized']
Status	Pass
Severity	High

Test Case ID	CJM-007
Title	Deny access when both config token and header token are empty strings
Objective	Ensure CronJobMiddleware denies access if both the configuration token and header token are set to empty strings.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to ''
Test Steps	1. Set config 'services.cron_job.auth_token' to '' (empty string). 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to '' (empty string). 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = ''
Expected Result	- Response is instance of JsonResponse - Response status code is 401 - Response data is ['unauthorized']
Actual Result	Response is JsonResponse with status 401 and data ['unauthorized']
Status	Pass
Severity	High

Test Case ID	CJM-008
Title	Deny access when config key is not set at all
Objective	To verify that CronJobMiddleware denies access when the configuration key is missing.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' is NOT set
Test Steps	1. Remove config key 'services.cron_job.auth_token'. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'some_valid_looking_token'. 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = 'some_valid_looking_token'

Expected Result	- Response is instance of JsonResponse - Response status code is 401 - Response data is ['unauthorized']
Actual Result	Response is JsonResponse with status 401 and data ['unauthorized']
Status	Pass
Severity	High

Test Case ID	CJM-009
Title	Pass request to next middleware when authorized
Objective	To verify that CronJobMiddleware invokes the next middleware when the correct token is present.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'test_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'test_token'. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'test_token'. 4. Prepare a closure to simulate next middleware. 5. Pass request through CronJobMiddleware. 6. Observe if next middleware is called and request is passed.
Test Data	- Header: x-authorization-token = 'test_token'
Expected Result	- Next middleware closure is called (nextCalled = true) - Passed request equals the original request object - Response is 'next_response'
Actual Result	Next middleware was called, request was passed unchanged, response is 'next_response'
Status	Pass
Severity	High

Test Case ID	CJM-010
Title	Handle case-insensitive header names for authorization token
Objective	To verify that CronJobMiddleware reads the authorization token header regardless of its case.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'test_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'test_token'. 2. For each of these headers: - 'X-Authorization-Token' - 'X-AUTHORIZATION-TOKEN' - 'x-authorization-token' - 'X-Authorization-Token' (duplicate for consistency) 3. Create a GET request to '/' with each header and value 'test_token'. 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Headers: various case variations, each with value 'test_token'
Expected Result	- Each request returns 'allowed'
Actual Result	All variations return 'allowed'
Status	Pass
Severity	High

Test Case ID	CJM-011
Title	Return JSON response with 401 status on unauthorized request

Objective	To ensure unauthorized requests return a proper JSON response with 401 status and correct headers.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'test_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'test_token'. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'wrong_token'. 4. Pass request through CronJobMiddleware. 5. Observe response, status, headers, and payload.
Test Data	- Header: x-authorization-token = 'wrong_token'
Expected Result	- Response is instance of JsonResponse - Response status code is 401 - Content-Type header contains 'application/json' - Response data is ['unauthorized']
Actual Result	Response is a JsonResponse with status 401, application/json header, data ['unauthorized']
Status	Pass
Severity	High

Test Case ID	CJM-012
Title	Handle special characters in authorization tokens
Objective	To verify CronJobMiddleware successfully matches tokens containing special characters.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to a special character token
Test Steps	1. Set config 'services.cron_job.auth_token' to 'token@#$%^&*()_+-=[]{}\|;:,.<>?'. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'token@#$%^&*()_+-=[]{}|;:,.<>?'. 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = 'token@#$%^&*()_+-=[]{}|;:,.<>?'
Expected Result	- Response is 'allowed'
Actual Result	Response is 'allowed'
Status	Pass
Severity	High

Test Case ID	CJM-013
Title	Handle very long authorization tokens
Objective	To verify CronJobMiddleware works with very long tokens as the authorization value.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to a 1000-character token
Test Steps	1. Set config 'services.cron_job.auth_token' to a string consisting of 1000 'a' characters. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to the same long string. 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = 'aaaaaaaa...'(1000 times)

Expected Result	- Response is 'allowed'
Actual Result	Response is 'allowed'
Status	Pass
Severity	High

Test Case ID	CJM-014
Title	Handle whitespace in authorization tokens
Objective	To verify CronJobMiddleware matches tokens that contain whitespace.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'token with spaces'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'token with spaces'. 2. Create a GET request to '/'. 3. Set header 'x-authorization-token' to 'token with spaces'. 4. Pass request through CronJobMiddleware. 5. Observe response.
Test Data	- Header: x-authorization-token = 'token with spaces'
Expected Result	- Response is 'allowed'
Actual Result	Response is 'allowed'
Status	Pass
Severity	High

Test Case ID	CJM-015
Title	Strict matching of authorization token (no normalization)
Objective	To ensure CronJobMiddleware only authorizes requests when the token matches exactly, without normalization or trimming.
Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'exact_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'exact_token'. 2. For each invalid token: - 'Exact_Token' - 'exact_token ' - ' exact_token' - 'exact token' - 'exact_token_extra' 3. Create a GET request to '/' with each invalid token in the header. 4. Pass each request through CronJobMiddleware. 5. Observe response status.
Test Data	- Header: x-authorization-token set to each invalid token
Expected Result	- Each response status is 401
Actual Result	All responses have status 401
Status	Pass
Severity	High

Test Case ID	CJM-016
Title	Support for different HTTP methods with authorized token
Objective	To verify that CronJobMiddleware works with various HTTP methods when the correct token is provided.

Preconditions	- Application running - Database seeded - Config key 'services.cron_job.auth_token' set to 'test_token'
Test Steps	1. Set config 'services.cron_job.auth_token' to 'test_token'. 2. For each method: GET, POST, PUT, PATCH, DELETE, OPTIONS 3. Create a request to '/' using the method. 4. Set header 'x-authorization-token' to 'test_token'. 5. Pass request through CronJobMiddleware. 6. Observe response.
Test Data	- Header: x-authorization-token = 'test_token' - Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Expected Result	- Each response is 'allowed'
Actual Result	All responses are 'allowed'
Status	Pass
Severity	High

Test Case ID	CJM-017
Title	Middleware instantiation does not throw errors
Objective	To ensure CronJobMiddleware can be instantiated without exceptions or errors.
Preconditions	- Application running
Test Steps	1. Instantiate CronJobMiddleware.
Test Data	- None
Expected Result	- Object is instance of CronJobMiddleware
Actual Result	Object is instance of CronJobMiddleware
Status	Pass
Severity	Low

File: Currency-Test.txt

Test Case ID	C-001
Title	Verify Currency model extends Eloquent Model
Objective	Confirm that the Currency model is a subclass of Laravel's Eloquent Model
Preconditions	- Application running - Laravel framework installed - Crater\Models\Currency class exists
Test Steps	1. Check if Currency class inherits from Model using is_subclass_of(). 2. Assert the result is true.
Test Data	- Class: Crater\Models\Currency - Parent Class: Illuminate\Database\Eloquent\Model
Expected Result	- Currency class is confirmed to be a subclass of Eloquent Model.
Actual Result	Currency class correctly extends Eloquent Model.
Status	Pass
Severity	High

Test Case ID	C-002
Title	Verify Currency model uses HasFactory trait
Objective	Ensure the Currency model includes the HasFactory trait for factory support
Preconditions	- Application running - Laravel framework installed - Crater\Models\Currency class exists
Test Steps	1. Retrieve list of traits used by Currency class with class_uses(). 2. Assert HasFactory trait is in the list.
Test Data	- Trait: Illuminate\Database\Eloquent\Factories\HasFactory
Expected Result	- Currency model is confirmed to use the HasFactory trait.
Actual Result	Currency model uses HasFactory trait as expected.
Status	Pass
Severity	Medium

Test Case ID	C-003
Title	Validate guarded property configuration on Currency model
Objective	Check that the guarded property on the Currency model contains only 'id'
Preconditions	- Application running - Crater\Models\Currency class exists
Test Steps	1. Instantiate a new Currency model. 2. Use Reflection to access the 'guarded' property. 3. Retrieve the value of the 'guarded' property. 4. Assert the 'guarded' property is an array. 5. Assert 'guarded' property contains 'id'. 6. Assert the count of elements in 'guarded' property is exactly 1.
Test Data	- No input data, using new Currency model object
Expected Result	- 'guarded' property is array - 'guarded' includes 'id' - 'guarded' contains exactly one element
Actual Result	'guarded' property correctly set to ['id'] with only one element.
Status	Pass
Severity	High

Test Case ID	C-004
Title	Ensure guarded properties are not mass assignable on Currency model
Objective	Validate that guarded attributes, specifically 'id', are protected from mass assignment
Preconditions	- Application running - Crater\Models\Currency class exists
Test Steps	1. Create data array with keys 'id', 'name', and 'code'. 2. Instantiate a new Currency model. 3. Fill Currency model with the data array. 4. Assert 'id' remains null after filling. 5. Assert 'name' is set to 'US Dollar'. 6. Assert 'code' is set to 'USD'.
Test Data	- Input: ['id' => 1, 'name' => 'US Dollar', 'code' => 'USD']
Expected Result	- Currency id remains null (not mass-assignable) - Currency name equals 'US Dollar' - Currency code equals 'USD'
Actual Result	Mass assignment works as expected: 'id' is null, 'name' is 'US Dollar', 'code' is 'USD'.
Status	Pass
Severity	High

Test Case ID	C-005
Title	Verify Currency model default table name
Objective	Confirm that the Currency model uses the correct default table name
Preconditions	- Application running - Crater\Models\Currency class exists
Test Steps	1. Instantiate a new Currency model. 2. Retrieve the table name using getTable(). 3. Assert the table name equals 'currencies'.
Test Data	- No input data
Expected Result	- Table name is 'currencies'
Actual Result	Table name for Currency model confirmed to be 'currencies'.
Status	Pass
Severity	Medium

Test Case ID	C-006
Title	Test Currency model instantiation via Factory make method
Objective	Ensure the Currency model can be instantiated using its factory via the make() method
Preconditions	- Application running - Crater\Models\Currency class exists - CurrencyFactory is available or dynamically defined in test
Test Steps	1. Check for existence or dynamically create Database\Factories\CurrencyFactory. 2. Call Currency::factory()->make() to instantiate a Currency model. 3. Assert the returned object is instance of Currency. 4. Assert id property of the model is null. 5. Assert name property is not null. 6. Assert code property is not null.
Test Data	- Factory generates: - name: "Currency " + random(5) - code: random 3 upper-case letters

Expected Result	- Object is instance of Currency - id is null - name is not null - code is not null
Actual Result	Currency model successfully instantiated using factory; id is null, name and code are generated.
Status	Pass
Severity	Medium

File: CurrencyCollection-Test.txt

Test Case ID	CC-001
Title	Currency collection returns empty array for empty collection
Objective	Verify that CurrencyCollection returns an empty array when provided with an empty collection.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create an empty collection using `collect([])`. 2. Instantiate a CurrencyCollection with the empty collection. 3. Call `toArray` on the collection, passing a new Request. 4. Check if the result is an array. 5. Check if the result is empty.
Test Data	- Input: Empty collection (`[]`) - Expected: Result is an array and is empty (`[]`)
Expected Result	- The returned result is an array. - The result array is empty.
Actual Result	Returned array is empty as expected.
Status	Pass
Severity	Medium

Test Case ID	CC-002
Title	Currency collection handles single currency object
Objective	Ensure CurrencyCollection correctly serializes a single currency object.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection with one TestCurrencyModel with attributes: id=3, name='British Pound', code='GBP', symbol='£', precision=2. 2. Instantiate a CurrencyCollection with this collection. 3. Call `toArray` on the collection, passing a new Request. 4. Check that the result is an array of length 1. 5. Check that the first element contains id=3, name='British Pound', code='GBP'.
Test Data	- Input: Collection with one TestCurrencyModel - id: 3 - name: 'British Pound' - code: 'GBP' - symbol: '£' - precision: 2 - Expected: Array with one object having above properties
Expected Result	- The result is an array of count 1. - The [0] element has id=3, name='British Pound', code='GBP'.
Actual Result	The result array contains one element with the expected properties.
Status	Pass
Severity	Medium

Test Case ID	CC-003
Title	Currency collection handles empty string values in properties
Objective	Validate CurrencyCollection properly serializes currency objects with empty strings for properties.

Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection with a TestCurrencyModel: - id=5, name='', code='', symbol='', precision='' 2. Instantiate a CurrencyCollection with this collection. 3. Call `toArray` on the collection, passing a new Request. 4. Verify the result's first element has the provided values.
Test Data	- Input: Collection with TestCurrencyModel - id: 5 - name: '' - code: '' - symbol: '' - precision: '' - Expected: Array with one object mirroring input properties
Expected Result	- result[0]['id'] is 5 - result[0]['name'] is '' - result[0]['code'] is '' - result[0]['symbol'] is '' - result[0]['precision'] is ''
Actual Result	Properties with empty string values are correctly serialized.
Status	Pass
Severity	Low

Test Case ID	CC-004
Title	Currency collection can be serialized to JSON
Objective	Confirm CurrencyCollection can be serialized to valid JSON, preserving currency data.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection with one TestCurrencyModel with id=6, name='Japanese Yen', code='JPY', symbol='¥', precision=0. 2. Instantiate a CurrencyCollection with this collection. 3. Serialize the CurrencyCollection to JSON using `json_encode`. 4. Verify the JSON is valid. 5. Decode the JSON and check the values for id, name, and code.
Test Data	- Input: Collection with TestCurrencyModel - id: 6 - name: 'Japanese Yen' - code: 'JPY' - symbol: '¥' - precision: 0 - Expected: Valid JSON with proper values
Expected Result	- The JSON string is valid JSON. - Decoded array contains [0] with id=6, name='Japanese Yen', code='JPY'.
Actual Result	JSON serialization and data decoding work as expected.
Status	Pass
Severity	Medium

Test Case ID	CC-005
Title	Currency collection preserves collection metadata
Objective	Ensure CurrencyCollection keeps additional metadata in the response.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available

Test Steps	1. Create a collection with two TestCurrencyModel items: id=7, name='Currency 1'; id=8, name='Currency 2'. 2. Instantiate a CurrencyCollection with this collection. 3. Add additional metadata: `['meta' => ['total' => 2, 'page' => 1]]`. 4. Call `toResponse` on the collection. 5. Retrieve response data using `getData(true)`. 6. Check that the response data contains both 'data' and 'meta' keys. 7. Confirm meta contains total=2 and page=1. 8. Confirm data count is 2.
Test Data	- Input: Two-item collection; meta: total=2, page=1 - Expected: Response includes meta and data fields matching input
Expected Result	- Response includes keys 'data' and 'meta'. - 'meta' field matches expected meta. - 'data' contains 2 items.
Actual Result	Response structure and metadata preservation are correct.
Status	Pass
Severity	Medium

Test Case ID	CC-006
Title	CurrencyCollection inherits from ResourceCollection
Objective	Verify CurrencyCollection is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Preconditions	- Application running - Database seeded - CurrencyCollection and ResourceCollection classes available
Test Steps	1. Instantiate a CurrencyCollection with an empty collection. 2. Check if the instance is of ResourceCollection type.
Test Data	- Input: Empty collection - Expected: Instance of ResourceCollection
Expected Result	CurrencyCollection is an instance of ResourceCollection.
Actual Result	Confirmed inheritance from ResourceCollection.
Status	Pass
Severity	Low

Test Case ID	CC-007
Title	CurrencyCollection static make method functionality
Objective	Ensure CurrencyCollection::make() constructs a correct CurrencyCollection instance.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection with one TestCurrencyModel: id=11, name='Static Make'. 2. Construct a CurrencyCollection using static `make()` method. 3. Call `toArray` and verify output is correct.
Test Data	- Input: Collection with TestCurrencyModel (id=11, name='Static Make') - Expected: Output array with one object {id: 11, name: 'Static Make'}
Expected Result	- Output is array containing object with id=11, name='Static Make'
Actual Result	Static make method works correctly; output as expected.
Status	Pass
Severity	Low

Test Case ID	CC-008
Title	Currency collection handles large number of items
Objective	To ensure CurrencyCollection can process and serialize a large number of currency objects.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection of 50 TestCurrencyModel objects with ids 1-50, names 'Currency 1' to 'Currency 50', codes 'C1' to 'C50'. 2. Instantiate a CurrencyCollection with this large collection. 3. Call `toArray` on the collection. 4. Verify array length is 50. 5. Confirm first and last elements match expected values.
Test Data	- Input: Collection of 50 currencies, id=1-50, name="Currency n" - Expected: Array of 50; [0]=id=1,name='Currency 1'; [49]=id=50,name='Currency 50'
Expected Result	- Output is array of length 50. - The first element matches id=1, name='Currency 1'. - The last element matches id=50, name='Currency 50'.
Actual Result	All currency objects correctly serialized in large array.
Status	Pass
Severity	Medium

Test Case ID	CC-009
Title	Currency collection handles currency with special characters
Objective	Ensure CurrencyCollection correctly serializes currencies with special and non-ASCII characters in properties.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection with one TestCurrencyModel: - id=12 - name='Currency with spécial chàracters' - code='SP€CIAL' - symbol='¤' - precision=2 2. Instantiate a CurrencyCollection. 3. Call `toArray`. 4. Verify special characters are present in output.
Test Data	- Input: Currency with special and accented characters - Expected: Array with corresponding special character values
Expected Result	- result[0]['name'] is 'Currency with spécial chàracters' - result[0]['code'] is 'SP€CIAL' - result[0]['symbol'] is '¤'
Actual Result	Special characters correctly preserved in output.
Status	Pass
Severity	Low

Test Case ID	CC-010
Title	Currency collection response structure conforms to JsonResponse
Objective	Confirm that the CurrencyCollection response is a JsonResponse and has correct status/content-type.

Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection with one TestCurrencyModel (id=13, name='Response Test'). 2. Instantiate CurrencyCollection and call `toResponse(new Request())`. 3. Verify response is instance of Illuminate\Http\JsonResponse. 4. Check status code is 200. 5. Validate 'Content-Type' header contains 'application/json'.
Test Data	- Input: Collection with TestCurrencyModel (id=13, name='Response Test') - Expected: JsonResponse with status=200 and proper content-type
Expected Result	- Response is an instance of JsonResponse - Status code is 200 - Content-Type includes 'application/json'
Actual Result	Response conforms to expected JsonResponse structure.
Status	Pass
Severity	Medium

Test Case ID	CC-011
Title	Currency collection with different precision values
Objective	Verify CurrencyCollection properly handles various precision values, including negative, zero, and high values.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. For each test case: precision=0, 2, 4, -1 2. Create a collection with one TestCurrencyModel: id=14, name='Precision Test', code='PT', precision as above. 3. Instantiate CurrencyCollection and call `toArray`. 4. Assert that output's 'precision' matches input precision.
Test Data	- Input: Four separate currencies with precision 0, 2, 4, -1 - Expected: Output where precision value is preserved
Expected Result	- result[0]['precision'] matches input precision for each case (0, 2, 4, -1)
Actual Result	All precision values are precisely serialized.
Status	Pass
Severity	Low

Test Case ID	CC-012
Title	Currency collection handles boolean and integer swap_currency_symbol
Objective	Ensure CurrencyCollection correctly processes boolean and integer swap_currency_symbol values.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. For each test case: - swap_currency_symbol = true (expected true) - swap_currency_symbol = false (expected false) - swap_currency_symbol = 1 (expected 1) - swap_currency_symbol = 0 (expected 0) 2. Create a TestCurrencyModel with respective swap_currency_symbol. 3. Instantiate CurrencyCollection. 4. Call `toArray`. 5. Verify result matches expected value.

Test Data	- Input: Four currencies with swap_currency_symbol values of true, false, 1, and 0 - Expected: Corresponding output values for swap_currency_symbol
Expected Result	- result[0]['swap_currency_symbol'] is true, false, 1, and 0 respectively
Actual Result	swap_currency_symbol values are correctly processed and serialized.
Status	Pass
Severity	Low

Test Case ID	CC-013
Title	Currency collection with exchange_rate as decimal value
Objective	Validate CurrencyCollection preserves decimal values for exchange_rate.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection with one TestCurrencyModel: id=20, name='Exchange Test', exchange_rate=1.2345. 2. Instantiate CurrencyCollection and call `toArray`. 3. Verify exchange_rate in the result matches input.
Test Data	- Input: Currency with exchange_rate=1.2345 - Expected: exchange_rate value is 1.2345 in output
Expected Result	- result[0]['exchange_rate'] is 1.2345
Actual Result	exchange_rate correctly serialized as decimal.
Status	Pass
Severity	Low

Test Case ID	CC-014
Title	Currency collection preserves separator fields
Objective	Verify that thousand_separator and decimal_separator properties are correctly serialized.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available
Test Steps	1. Create a collection with one TestCurrencyModel: id=21, name='Separator Test', thousand_separator='.', decimal_separator=','. 2. Instantiate CurrencyCollection and call `toArray`. 3. Assert output fields match input separators.
Test Data	- Input: Currency with thousand_separator='.', decimal_separator=',' - Expected: Output matches input separators
Expected Result	- result[0]['thousand_separator'] is '.' - result[0]['decimal_separator'] is ','
Actual Result	Separator fields are correctly serialized.
Status	Pass
Severity	Low

Test Case ID	CC-015
Title	Currency collection works with generic object type
Objective	Ensure CurrencyCollection works with an object (stdClass) that contains all required currency properties.
Preconditions	- Application running - Database seeded - CurrencyCollection and CurrencyResource classes available

Test Steps	1. Create a stdClass object and set: - id=22, name='Object Currency', code='OC', symbol='O', precision=2, - thousand_separator=',', decimal_separator='.', swap_currency_symbol=false, - exchange_rate=1.0, created_at=null, updated_at=null. 2. Create a collection containing this object. 3. Instantiate CurrencyCollection and call `toArray`. 4. Assert output fields match input object.
Test Data	- Input: stdClass object with all currency properties set - Expected: Output array contains matching fields
Expected Result	- result[0]['id'] is 22 - result[0]['name'] is 'Object Currency' - result[0]['code'] is 'OC'
Actual Result	Generic object currency works as expected.
Status	Pass
Severity	Low

File: CurrencyResource-Test.txt

Test Case ID	CR-001
Title	Transform Currency Model into Array with All Expected Attributes
Objective	Verify that the CurrencyResource correctly transforms a currency model into an array containing all expected attributes with proper values.
Preconditions	- Application running - Database seeded - Mock currency object available with all attributes populated - CurrencyResource class implemented - PHPUnit/Pest and Mockery are installed
Test Steps	1. Create a mock currency object with the following attributes: - id: 1 - name: 'US Dollar' - code: 'USD' - symbol: '$' - precision: 2 - thousand_separator: ',' - decimal_separator: '.' - swap_currency_symbol: false - exchange_rate: 1.0 2. Instantiate a CurrencyResource object with the mock currency. 3. Mock an HTTP request object. 4. Call the toArray method on the CurrencyResource using the mocked request. 5. Assert that the resulting array has 9 elements. 6. Assert that each attribute in the array matches the corresponding value from the mock currency object.
Test Data	- Input: Currency object - id: 1 - name: 'US Dollar' - code: 'USD' - symbol: '$' - precision: 2 - thousand_separator: ',' - decimal_separator: '.' - swap_currency_symbol: false - exchange_rate: 1.0 - Expected values: - Array with all 9 attributes and correct values
Expected Result	- The result is an array with 9 elements. - Each array attribute matches the mock currency: - id == 1 - name == 'US Dollar' - code == 'USD' - symbol == '$' - precision == 2 - thousand_separator == ',' - decimal_separator == '.' - swap_currency_symbol == false - exchange_rate == 1.0
Actual Result	The CurrencyResource returns an array with all expected attributes, each correctly matching the input currency object's values.
Status	Pass
Severity	High

Test Case ID	CR-002
Title	Handle Null and Empty Values for Currency Attributes

Objective	Ensure CurrencyResource correctly handles null and empty values for currency attributes during transformation.
Preconditions	- Application running - Database seeded - Mock currency object available with null/empty attribute values - CurrencyResource class implemented - PHPUnit/Pest and Mockery are installed
Test Steps	1. Create a mock currency object with the following attributes: - id: null - name: '' - code: 'EUR' - symbol: '€' - precision: 2 - thousand_separator: null - decimal_separator: null - swap_currency_symbol: true - exchange_rate: 0.85 2. Instantiate a CurrencyResource object with the mock currency. 3. Mock an HTTP request object. 4. Call the toArray method on the CurrencyResource using the mocked request. 5. Assert that the resulting array has 9 elements. 6. Assert that each attribute in the array properly reflects the null/empty or provided value.
Test Data	- Input: Currency object - id: null - name: '' - code: 'EUR' - symbol: '€' - precision: 2 - thousand_separator: null - decimal_separator: null - swap_currency_symbol: true - exchange_rate: 0.85 - Expected values: - Array with all 9 attributes, some values null or empty as supplied
Expected Result	- The result is an array with 9 elements. - Array attributes match the mock currency: - id == null - name == '' - code == 'EUR' - symbol == '€' - precision == 2 - thousand_separator == null - decimal_separator == null - swap_currency_symbol == true - exchange_rate == 0.85
Actual Result	The CurrencyResource returns an array with all expected attributes, correctly handling and reflecting null/empty input values.
Status	Pass
Severity	High

Test Case ID	CR-003
Title	Validate All Expected Keys Present in Transformed Output
Objective	Confirm that CurrencyResource always includes all expected keys in the output array regardless of input data values.
Preconditions	- Application running - Database seeded - Mock currency object available with attributes populated - CurrencyResource class implemented - PHPUnit/Pest and Mockery are installed

Test Steps	1. Create a mock currency object with the following attributes: - id: 1 - name: 'Test Currency' - code: 'TST' - symbol: 'T' - precision: 2 - thousand_separator: ',' - decimal_separator: '.' - swap_currency_symbol: false - exchange_rate: 1.0 2. Instantiate a CurrencyResource object with the mock currency. 3. Mock an HTTP request object. 4. Call the toArray method on the CurrencyResource using the mocked request. 5. Define the expected keys: - id - name - code - symbol - precision - thousand_separator - decimal_separator - swap_currency_symbol - exchange_rate 6. Assert that each of these expected keys is present in the result array.
Test Data	- Input: Currency object with all attributes populated - Expected keys: - id - name - code - symbol - precision - thousand_separator - decimal_separator - swap_currency_symbol - exchange_rate
Expected Result	- The result array includes all nine expected keys, regardless of input values.
Actual Result	All nine expected keys are present in the CurrencyResource output array for the provided currency object.
Status	Pass
Severity	High

File: CustomFieldRequest-Test.txt

Test Case ID	CFR-001
Title	Verify that the authorize method always returns true
Objective	Ensure that the authorize() method of CustomFieldRequest consistently returns true for all requests, validating that any user is authorized to perform the action.
Preconditions	- Application running - All dependencies installed - CustomFieldRequest class available - No specific user authorization required
Test Steps	1. Instantiate a new CustomFieldRequest object. 2. Invoke the authorize() method on the CustomFieldRequest object. 3. Observe the returned value.
Test Data	- CustomFieldRequest instance with default properties - Expected value: true from authorize() method
Expected Result	- The authorize() method of CustomFieldRequest returns true.
Actual Result	The authorize() method returned true as expected.
Status	Pass
Severity	High

Test Case ID	CFR-002
Title	Validate that rules method returns predefined validation rules
Objective	Confirm that the rules() method of CustomFieldRequest returns the exact array of required field validation rules for submission.
Preconditions	- Application running - All dependencies installed - CustomFieldRequest class available
Test Steps	1. Instantiate a new CustomFieldRequest object. 2. Invoke the rules() method on the CustomFieldRequest object. 3. Compare the returned array to the expected validation rules.
Test Data	- CustomFieldRequest instance with default properties - Expected rules array: - 'name' => 'required' - 'label' => 'required' - 'model_type' => 'required' - 'order' => 'required' - 'type' => 'required' - 'is_required' => 'required|boolean' - 'options' => 'array' - 'placeholder' => 'string|nullable'
Expected Result	- The rules() method of CustomFieldRequest returns the predefined validation rules array as specified above.
Actual Result	The rules() method returned the exact validation rules array as expected.
Status	Pass
Severity	High

File: CustomFieldValue-Test.txt

Test Case ID	CFV-001
Title	setTimeAnswerAttribute sets time_answer for valid time strings
Objective	Verify setTimeAnswerAttribute correctly sets time_answer for different valid time string formats.
Preconditions	- Application running - Database seeded - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call setTimeAnswerAttribute with '10:30 AM'. 3. Assert that time_answer equals '10:30:00'. 4. Instantiate CustomFieldValue model. 5. Call setTimeAnswerAttribute with '14:45:15'. 6. Assert that time_answer equals '14:45:15'. 7. Instantiate CustomFieldValue model. 8. Call setTimeAnswerAttribute with '2pm'. 9. Assert that time_answer equals '14:00:00'.
Test Data	- Input: '10:30 AM', '14:45:15', '2pm' - Expected: '10:30:00', '14:45:15', '14:00:00'
Expected Result	- time_answer is '10:30:00' after setting '10:30 AM' - time_answer is '14:45:15' after setting '14:45:15' - time_answer is '14:00:00' after setting '2pm'
Actual Result	- time_answer correctly set as expected in all cases.
Status	Pass
Severity	High

Test Case ID	CFV-002
Title	setTimeAnswerAttribute sets time_answer to null for empty string
Objective	Verify setTimeAnswerAttribute sets time_answer to null when an empty string is provided.
Preconditions	- Application running - Database seeded - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call setTimeAnswerAttribute with an empty string ''. 3. Assert that time_answer is null.
Test Data	- Input: '' - Expected: null
Expected Result	- time_answer is null after setting with empty string
Actual Result	- time_answer is null as expected.
Status	Pass
Severity	High

Test Case ID	CFV-003
Title	setTimeAnswerAttribute sets time_answer to null for null input
Objective	Verify setTimeAnswerAttribute sets time_answer to null when input is null.
Preconditions	- Application running - Database seeded - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call setTimeAnswerAttribute with null. 3. Assert that time_answer is null.

Test Data	- Input: null - Expected: null
Expected Result	- time_answer is null after setting with null input
Actual Result	- time_answer is null as expected.
Status	Pass
Severity	High

Test Case ID	CFV-004
Title	setTimeAnswerAttribute handles various valid time formats
Objective	Verify setTimeAnswerAttribute can parse other valid time formats correctly.
Preconditions	- Application running - Database seeded - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call setTimeAnswerAttribute with '5:00 PM'. 3. Assert that time_answer equals '17:00:00'. 4. Instantiate CustomFieldValue model. 5. Call setTimeAnswerAttribute with '08:00'. 6. Assert that time_answer equals '08:00:00'.
Test Data	- Input: '5:00 PM', '08:00' - Expected: '17:00:00', '08:00:00'
Expected Result	- time_answer is '17:00:00' after '5:00 PM' - time_answer is '08:00:00' after '08:00'
Actual Result	- Values set as expected.
Status	Pass
Severity	High

Test Case ID	CFV-005
Title	setTimeAnswerAttribute handles midnight and noon correctly
Objective	Verify setTimeAnswerAttribute processes '12:00 AM' and '12:00 PM' accurately.
Preconditions	- Application running - Database seeded - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call setTimeAnswerAttribute with '12:00 AM'. 3. Assert that time_answer equals '00:00:00'. 4. Instantiate CustomFieldValue model. 5. Call setTimeAnswerAttribute with '12:00 PM'. 6. Assert that time_answer equals '12:00:00'.
Test Data	- Input: '12:00 AM', '12:00 PM' - Expected: '00:00:00', '12:00:00'
Expected Result	- time_answer is '00:00:00' after '12:00 AM' - time_answer is '12:00:00' after '12:00 PM'
Actual Result	- Values set as expected.
Status	Pass
Severity	High

Test Case ID	CFV-006
Title	setTimeAnswerAttribute handles 24-hour format

Objective	Assert setTimeAnswerAttribute accepts and saves 24-hour time formats correctly.
Preconditions	- Application running - Database seeded - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call setTimeAnswerAttribute with '23:59:59'. 3. Assert that time_answer equals '23:59:59'. 4. Instantiate CustomFieldValue model. 5. Call setTimeAnswerAttribute with '00:00:00'. 6. Assert that time_answer equals '00:00:00'.
Test Data	- Input: '23:59:59', '00:00:00' - Expected: '23:59:59', '00:00:00'
Expected Result	- time_answer is '23:59:59' after '23:59:59' - time_answer is '00:00:00' after '00:00:00'
Actual Result	- Values set as expected.
Status	Pass
Severity	High

Test Case ID	CFV-007
Title	company relationship returns a BelongsTo instance
Objective	Confirm the company relationship method returns a BelongsTo object.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call company() relationship method. 3. Assert the returned object is an instance of BelongsTo.
Test Data	- Method: company()
Expected Result	- Returned object is instance of BelongsTo.
Actual Result	- Returns BelongsTo instance as expected.
Status	Pass
Severity	Medium

Test Case ID	CFV-008
Title	customField relationship returns a BelongsTo instance
Objective	Confirm the customField relationship method returns a BelongsTo object.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call customField() relationship method. 3. Assert the returned object is an instance of BelongsTo.
Test Data	- Method: customField()
Expected Result	- Returned object is instance of BelongsTo.
Actual Result	- Returns BelongsTo instance as expected.
Status	Pass
Severity	Medium

Test Case ID	CFV-009
Title	customFieldValuable relationship returns a MorphTo instance

Objective	Confirm the customFieldValuable relationship method returns a MorphTo object.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Call customFieldValuable() relationship method. 3. Assert the returned object is an instance of MorphTo.
Test Data	- Method: customFieldValuable()
Expected Result	- Returned object is instance of MorphTo.
Actual Result	- Returns MorphTo instance as expected.
Status	Pass
Severity	Medium

Test Case ID	CFV-010
Title	guarded attributes are set correctly
Objective	Ensure "guarded" property includes "id" to protect primary key from mass-assignment.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Use ReflectionClass to get "guarded" property. 3. Assert "guarded" includes 'id'.
Test Data	- Property checked: guarded
Expected Result	- 'id' is present in the guarded attributes.
Actual Result	- 'id' present in guarded array.
Status	Pass
Severity	Medium

Test Case ID	CFV-011
Title	dates attributes are set correctly
Objective	Ensure "dates" property contains required fields for date casting.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Use ReflectionClass to get "dates" property. 3. Assert "dates" includes 'date_answer'. 4. Assert "dates" includes 'date_time_answer'.
Test Data	- Property checked: dates
Expected Result	- 'date_answer' and 'date_time_answer' present in dates array.
Actual Result	- Both present as expected.
Status	Pass
Severity	Medium

Test Case ID	CFV-012
Title	appends attributes are set correctly
Objective	Verify "appends" property includes "defaultAnswer" for computed attributes.
Preconditions	- Application running - CustomFieldValue model is available

Test Steps	1. Instantiate CustomFieldValue model. 2. Use ReflectionClass to get "appends" property. 3. Assert "appends" includes 'defaultAnswer'.
Test Data	- Property checked: appends
Expected Result	- 'defaultAnswer' present in appends array.
Actual Result	- Present as expected.
Status	Pass
Severity	Medium

Test Case ID	CFV-013
Title	model can be instantiated without errors
Objective	Ensure CustomFieldValue model instantiates successfully.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Assert resulting object is instance of CustomFieldValue.
Test Data	- Instantiation
Expected Result	- Object instantiated without error and of correct type.
Actual Result	- Instantiates correctly.
Status	Pass
Severity	Low

Test Case ID	CFV-014
Title	model can set and get text_answer attribute
Objective	Verify the model allows setting and retrieving the text_answer property.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Set text_answer to 'Test text answer'. 3. Assert that text_answer equals 'Test text answer'.
Test Data	- Input: text_answer = 'Test text answer'
Expected Result	- text_answer returns 'Test text answer' after set
Actual Result	- Value is returned correctly.
Status	Pass
Severity	Low

Test Case ID	CFV-015
Title	model can set and get number_answer attribute
Objective	Verify the model allows setting and retrieving the number_answer property.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Set number_answer to 12345. 3. Assert that number_answer equals 12345.
Test Data	- Input: number_answer = 12345
Expected Result	- number_answer returns 12345 after set
Actual Result	- Value is returned correctly.
Status	Pass

Severity	Low

Test Case ID	CFV-016
Title	model can set and get boolean_answer attribute
Objective	Verify setting and getting boolean_answer property works as expected.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Set boolean_answer to true and assert that it returns true. 3. Set boolean_answer to false and assert that it returns false.
Test Data	- Input: boolean_answer = true, boolean_answer = false
Expected Result	- boolean_answer returns true after set to true - boolean_answer returns false after set to false
Actual Result	- Both values returned correctly.
Status	Pass
Severity	Low

Test Case ID	CFV-017
Title	model uses HasFactory trait
Objective	Ensure HasFactory trait is used by CustomFieldValue.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Get all used traits via class_uses(). 3. Assert HasFactory trait is among them.
Test Data	- Traits checked
Expected Result	- HasFactory trait present
Actual Result	- Trait confirmed present.
Status	Pass
Severity	Low

Test Case ID	CFV-018
Title	model has all required relationship methods
Objective	Verify company, customField, and customFieldValuable methods exist.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Assert method_exists() on 'company'. 3. Assert method_exists() on 'customField'. 4. Assert method_exists() on 'customFieldValuable'.
Test Data	- Methods: company, customField, customFieldValuable
Expected Result	- All methods exist.
Actual Result	- All methods confirmed present.
Status	Pass
Severity	Medium

Test Case ID	CFV-019
Title	model has getDefaultAnswerAttribute accessor

Objective	Confirm that getDefaultAnswerAttribute exists on model.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Assert method_exists() for getDefaultAnswerAttribute.
Test Data	- Method: getDefaultAnswerAttribute
Expected Result	- Method exists
Actual Result	- Method confirmed present.
Status	Pass
Severity	Low

Test Case ID	CFV-020
Title	model has setTimeAnswerAttribute mutator
Objective	Confirm that setTimeAnswerAttribute mutator exists.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Assert method_exists() for setTimeAnswerAttribute.
Test Data	- Method: setTimeAnswerAttribute
Expected Result	- Method exists
Actual Result	- Method confirmed present.
Status	Pass
Severity	Low

Test Case ID	CFV-021
Title	setTimeAnswerAttribute can handle multiple calls with different formats
Objective	Ensure model updates time_answer correctly on consecutive calls with different time formats.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Set with '9:00 AM' and assert '09:00:00'. 3. Set with '3:30 PM' and assert '15:30:00'. 4. Set with '18:45' and assert '18:45:00'.
Test Data	- Inputs: '9:00 AM', '3:30 PM', '18:45'
Expected Result	- time_answer is '09:00:00', then '15:30:00', then '18:45:00'
Actual Result	- Values updated and returned as expected.
Status	Pass
Severity	High

Test Case ID	CFV-022
Title	setTimeAnswerAttribute handles edge case times
Objective	Confirm model parses and stores edge time values correctly.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Set with '11:59 PM' and assert '23:59:00'. 3. Instantiate new CustomFieldValue. 4. Set with '1:00 AM' and assert '01:00:00'.

Test Data	- Inputs: '11:59 PM', '1:00 AM'
Expected Result	- time_answer is '23:59:00' after '11:59 PM' - time_answer is '01:00:00' after '1:00 AM'
Actual Result	- Values set correctly.
Status	Pass
Severity	High

Test Case ID	CFV-023
Title	CustomFieldValue extends Eloquent Model
Objective	Verify that CustomFieldValue model inherits from Eloquent Model correctly.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Assert object is instance of Model class.
Test Data	- Inheritance check
Expected Result	- Instance of Eloquent Model
Actual Result	- Is instance of Model.
Status	Pass
Severity	Low

Test Case ID	CFV-024
Title	model can set and get type attribute
Objective	Verify setting and getting the "type" property for different values.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Set type to 'TEXT' and assert value. 3. Set type to 'NUMBER' and assert value. 4. Set type to 'BOOLEAN' and assert value.
Test Data	- Inputs: 'TEXT', 'NUMBER', 'BOOLEAN'
Expected Result	- type property matches set value in each case
Actual Result	- type is returned as expected for each value.
Status	Pass
Severity	Low

Test Case ID	CFV-025
Title	guarded does not prevent setting common fields
Objective	Confirm that guarded attributes do not include text_answer, number_answer, boolean_answer, or time_answer.
Preconditions	- Application running - CustomFieldValue model is available
Test Steps	1. Instantiate CustomFieldValue model. 2. Use ReflectionClass to check guarded property. 3. Assert guarded does not contain 'text_answer'. 4. Assert guarded does not contain 'number_answer'. 5. Assert guarded does not contain 'boolean_answer'. 6. Assert guarded does not contain 'time_answer'.
Test Data	- Guarded checked for common fields
Expected Result	- None of the common fields are in guarded array.

Actual Result	- Confirmed common fields are not guarded.
Status	Pass
Severity	Medium

File: CustomFieldValueCollection-Test.txt

Test Case ID	CFVC-001
Title	toArray returns an empty array when the underlying collection is empty
Objective	Verify that the toArray method returns an empty array when the collection contains no items.
Preconditions	- Application running - Database seeded (if applicable for collection data) - CustomFieldValueCollection and TestCustomFieldValueResource classes available
Test Steps	1. Create a GET request to '/test'. 2. Create an empty Collection object. 3. Instantiate a CustomFieldValueCollection using the empty collection, setting 'collects' to TestCustomFieldValueResource. 4. Call toArray() on the testCollection with the request. 5. Assert that the result is an array. 6. Assert that the result is empty.
Test Data	- Request: GET /test - Collection: []
Expected Result	- Result is an array. - Result is empty ([]).
Actual Result	- Result is an empty array ([]).
Status	Pass
Severity	Medium

Test Case ID	CFVC-002
Title	toArray transforms multiple items correctly using a defined resource type
Objective	Ensure the toArray method correctly transforms multiple items, using the TestCustomFieldValueResource resource, and includes correct request parameter values.
Preconditions	- Application running - Database seeded with test items - TestCustomFieldValueResource class available
Test Steps	1. Create a GET request to '/test' with query parameter 'param' set to 'test_value'. 2. Create a Collection object with two objects: {id: 1, name: 'Item A', value: 'Alpha'} and {id: 2, name: 'Item B', value: 'Beta'}. 3. Instantiate CustomFieldValueCollection with the above collection, setting 'collects' to TestCustomFieldValueResource. 4. Call toArray() with the request. 5. Assert result is an array. 6. Assert result has 2 elements. 7. Assert first element matches expected output. 8. Assert second element matches expected output.
Test Data	- Request: GET /test?param=test_value - Collection: [(object)['id' => 1, 'name' => 'Item A', 'value' => 'Alpha'], (object)['id' => 2, 'name' => 'Item B', 'value' => 'Beta']]

Expected Result	- Result is array of length 2.
	- First item:
	{
	'id' => 1,
	'name' => 'Item A',
	'value' => 'Alpha',
	'request_param' => 'test_value',
	'type' => 'unknown'
	}
	- Second item:
	{
	'id' => 2,
	'name' => 'Item B',
	'value' => 'Beta',
	'request_param' => 'test_value',
	'type' => 'unknown'
	}

Actual Result	- Array of 2 elements with expected keys and values.
Status	Pass
Severity	Medium

Test Case ID	CFVC-003
Title	toArray correctly passes the request object to the underlying resource transformations
Objective	Verify that different request parameters are correctly propagated to resource transformations.
Preconditions	- Application running - Database seeded - TestCustomFieldValueResource class available
Test Steps	1. Create two GET requests: '/test?param=request_one' and '/another?param=request_two'. 2. Create a Collection with one object: {id: 3, value: 'Gamma'}. 3. Instantiate CustomFieldValueCollection, setting 'collects' to TestCustomFieldValueResource. 4. Call toArray() with first request; assert 'request_param' is 'request_one'. 5. Call toArray() with second request; assert 'request_param' is 'request_two'.
Test Data	- Requests: '/test?param=request_one', '/another?param=request_two' - Collection: [(object)['id' => 3, 'value' => 'Gamma']]
Expected Result	- result1[0]['request_param'] is 'request_one' - result2[0]['request_param'] is 'request_two'
Actual Result	- 'request_param' field reflects correct values for each request.
Status	Pass
Severity	Medium

Test Case ID	CFVC-004
Title	toArray handles mixed data types in collection when using a defined resource
Objective	Ensure toArray correctly transforms both objects and arrays in the collection, maintaining data consistency.
Preconditions	- Application running - Database seeded with mixed type items - TestCustomFieldValueResource class available

Test Steps	1. Create a GET request to '/test'. 2. Create a Collection with: - (object)['id' => 10, 'name' => 'Object Item'] - ['id' => 20, 'name' => 'Array Item', 'value' => 'Twenty'] 3. Instantiate CustomFieldValueCollection, setting 'collects' to TestCustomFieldValueResource. 4. Call toArray() with the request. 5. Assert result is array of length 2. 6. Assert first element matches expected object. 7. Assert second element matches expected array.
Test Data	- Request: GET /test - Collection: [(object)['id' => 10, 'name' => 'Object Item'], ['id' => 20, 'name' => 'Array Item', 'value' => 'Twenty']]

Expected Result	- First item:
	{
	'id' => 10,
	'name' => 'Object Item',
	'value' => null,
	'request_param' => 'default',
	'type' => 'unknown'
	}
	- Second item:
	{
	'id' => 20,
	'name' => 'Array Item',
	'value' => 'Twenty',
	'request_param' => 'default',
	'type' => 'unknown'
	}

Actual Result	- Both items transformed with expected keys, types and values.
Status	Pass
Severity	Medium

Test Case ID	CFVC-005
Title	toArray returns meta data when present in the collection instance
Objective	Ensure toArray includes 'meta' information when provided by the collection's with() method.
Preconditions	- Application running - Database seeded - TestCustomFieldValueResource class available
Test Steps	1. Create a GET request to '/meta'. 2. Create a Collection with one object: {id: 1, name: 'Item 1'}. 3. Extend CustomFieldValueCollection with 'with()' returning 'meta' data. 4. Override toArray() to merge output with 'with()' data. 5. Call toArray() and assert output contains both 'data' and 'meta' keys. 6. Assert that 'data' contains correct item. 7. Assert that 'meta' equals expected value.
Test Data	- Request: GET /meta - Collection: [(object)['id' => 1, 'name' => 'Item 1']] - Meta: ['key' => 'value']
Expected Result	- Result is array containing keys 'data' and 'meta'. - data[0]['id'] == 1 - meta equals ['key' => 'value']
Actual Result	- Output contains correct 'data' and 'meta' values.
Status	Pass
Severity	Medium

Test Case ID	CFVC-006
Title	toArray returns additional data when present in the collection instance
Objective	Verify toArray includes any extra fields provided by the collection's with() method in the result.
Preconditions	- Application running - Database seeded - TestCustomFieldValueResource class available
Test Steps	1. Create a GET request to '/additional'. 2. Create a Collection with one object: {id: 1, name: 'Item 1'}. 3. Extend CustomFieldValueCollection with 'with()' returning additional data. 4. Override toArray() to merge output with 'with()' data. 5. Call toArray() and assert output contains both 'data' and 'additional_field'. 6. Assert that 'data' contains correct item. 7. Assert that 'additional_field' equals expected value.
Test Data	- Request: GET /additional - Collection: [(object)['id' => 1, 'name' => 'Item 1']] - Additional field: 'extra_data'
Expected Result	- Result contains 'data' and 'additional_field' keys. - data[0]['id'] == 1 - additional_field == 'extra_data'
Actual Result	- Output includes correct 'data' and 'additional_field'.
Status	Pass
Severity	Medium

Test Case ID	CFVC-007
Title	CustomFieldValueCollection extends ResourceCollection
Objective	Verify that CustomFieldValueCollection is an instance of ResourceCollection.
Preconditions	- Application running - CustomFieldValueCollection class available - ResourceCollection class available
Test Steps	1. Create empty Collection. 2. Instantiate CustomFieldValueCollection with the collection. 3. Assert instance is of \Illuminate\Http\Resources\Json\ResourceCollection type.
Test Data	- Collection: []
Expected Result	- customCollection is instanceof ResourceCollection.
Actual Result	- customCollection is confirmed to be instance of ResourceCollection.
Status	Pass
Severity	Low

Test Case ID	CFVC-008
Title	CustomFieldValueCollection can be instantiated
Objective	Verify that a CustomFieldValueCollection object can be successfully created.
Preconditions	- Application running - CustomFieldValueCollection class available
Test Steps	1. Create an empty Collection. 2. Instantiate CustomFieldValueCollection with the collection. 3. Assert instance is of CustomFieldValueCollection type.
Test Data	- Collection: []
Expected Result	- customCollection is instanceof CustomFieldValueCollection.

Actual Result	- customCollection is confirmed to be instance of CustomFieldValueCollection.
Status	Pass
Severity	Low

Test Case ID	CFVC-009
Title	toArray handles single item collection
Objective	Verify toArray can correctly handle a collection with a single item.
Preconditions	- Application running - Database seeded with single test item - TestCustomFieldValueResource class available
Test Steps	1. Create a GET request to '/test'. 2. Create a Collection with one object: {id: 1, name: 'Single Item', value: 'One'}. 3. Instantiate CustomFieldValueCollection. 4. Call toArray() with request. 5. Assert result is array containing one element. 6. Assert that item matches expected values for id and name.
Test Data	- Request: GET /test - Collection: [(object)['id' => 1, 'name' => 'Single Item', 'value' => 'One']]
Expected Result	- Result is array of length 1. - result[0]['id'] == 1 - result[0]['name'] == 'Single Item'
Actual Result	- Output matches expected values for id and name.
Status	Pass
Severity	Medium

Test Case ID	CFVC-010
Title	toArray handles large collection
Objective	Ensure toArray performs correctly and efficiently on large collections.
Preconditions	- Application running - Database seeded with sufficient number of items - TestCustomFieldValueResource class available
Test Steps	1. Create a GET request to '/test'. 2. Create a Collection containing 10 objects, with id from 1 to 10, and value fields matching item number. 3. Instantiate CustomFieldValueCollection. 4. Call toArray() with request. 5. Assert result is array of length 10. 6. Assert first and last items have correct ids.
Test Data	- Request: GET /test - Collection: [(object)['id' => 1, 'name' => 'Item 1', 'value' => 'Value 1'], ..., (object)['id' => 10, 'name' => 'Item 10', 'value' => 'Value 10']]
Expected Result	- Result is array of length 10. - result[0]['id'] == 1 - result[9]['id'] == 10
Actual Result	- Output has 10 items, with correct ids on boundaries.
Status	Pass
Severity	Medium

File: CustomFieldValueResource-Test.txt

Test Case ID	CFVR-001
Title	Verify toArray returns correct structure with all fields
Objective	Ensure that the toArray method in CustomFieldValueResource returns array with all input fields, mapping values correctly.
Preconditions	- Application running - CustomFieldValueResource class loaded - Database seeded or mocked object instantiation - No relationship methods called
Test Steps	1. Instantiate a data object with all possible custom field values filled. 2. Create an anonymous class extending CustomFieldValueResource using this data. 3. Call the toArray method with a sample request. 4. Verify the returned array structure contains all expected fields with correct types and values. 5. Assert each field's value in the return array matches input test data.
Test Data	- Input: data object with keys (id=1, custom_field_valuable_type='App\\Models\\Invoice', custom_field_valuable_id=10, type='Input', boolean_answer=true, date_answer='2023-01-01', time_answer='10:30:00', string_answer='Test String', number_answer=123.45, date_time_answer='2023-01-01 10:30:00', custom_field_id=5, company_id=1, defaultAnswer='Test String') - Expected values: - Result['id'] = 1 - Result['custom_field_valuable_type'] = 'App\\Models\\Invoice' - Result['custom_field_valuable_id'] = 10 - Result['type'] = 'Input' - Result['boolean_answer'] = true - Result['string_answer'] = 'Test String' - Result['number_answer'] = 123.45 - Result['custom_field_id'] = 5 - Result['company_id'] = 1
Expected Result	- Returned value is an array. - Each field matches the value set in test data.
Actual Result	- Returned array contains all expected fields with correct values.
Status	Pass
Severity	High

Test Case ID	CFVR-002
Title	Verify toArray handles null values correctly
Objective	Ensure toArray returns expected array structure when custom field values are null.
Preconditions	- Application running - CustomFieldValueResource class loaded - Database seeded or mocked object instantiation
Test Steps	1. Instantiate a data object with all optional values set to null. 2. Create an anonymous class extending CustomFieldValueResource using this data. 3. Call the toArray method. 4. Verify returned array for null values on appropriate fields.

Test Data	- Input: data object with (id=2, custom_field_valuable_type='App\\Models\\Order', custom_field_valuable_id=20, type='Input', boolean_answer=null, date_answer=null, time_answer=null, string_answer=null, number_answer=null, date_time_answer=null, custom_field_id=6, company_id=2, defaultAnswer=null) - Expected values: - Result['boolean_answer'] = null - Result['date_answer'] = null - Result['string_answer'] = null - Result['number_answer'] = null - Result['default_answer'] = null
Expected Result	- All specified fields in output array are null.
Actual Result	- Returned array has null values for specified fields.
Status	Pass
Severity	Medium

Test Case ID	CFVR-003
Title	Verify dateTimeFormat returns null for null default_answer
Objective	Confirm that dateTimeFormat returns null when the default_answer property is null.
Preconditions	- Application running - CustomFieldValueResource class loaded - Mocked object with dateTimeFormat method
Test Steps	1. Instantiate a data object with default_answer set to null and type 'Date'. 2. Create an anonymous class extending CustomFieldValueResource. 3. Call dateTimeFormat. 4. Verify output is null.
Test Data	- Input: data object (default_answer=null, type='Date', company_id=1) - Expected value: null
Expected Result	- dateTimeFormat returns null.
Actual Result	- dateTimeFormat returns null.
Status	Pass
Severity	Medium

Test Case ID	CFVR-004
Title	Verify dateTimeFormat returns string value for text type
Objective	Ensure dateTimeFormat outputs string value when default_answer is string and type is 'Input'.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate a data object with default_answer set to 'Just a string value', type 'Input'. 2. Create anonymous CustomFieldValueResource class. 3. Call dateTimeFormat. 4. Verify output is the given string.
Test Data	- Input: data object (default_answer='Just a string value', type='Input', company_id=1) - Expected value: 'Just a string value'
Expected Result	- dateTimeFormat returns the string 'Just a string value'.
Actual Result	- dateTimeFormat returns 'Just a string value'.
Status	Pass
Severity	Medium

Test Case ID	CFVR-005
Title	Verify dateTimeFormat returns number value for number type
Objective	Ensure dateTimeFormat returns numeric value when default_answer is a number and type is 'Number'.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate a data object with default_answer set to 123.45, type 'Number'. 2. Create anonymous CustomFieldValueResource class. 3. Call dateTimeFormat. 4. Verify output is 123.45.
Test Data	- Input: data object (default_answer=123.45, type='Number', company_id=1) - Expected value: 123.45
Expected Result	- dateTimeFormat returns numeric value 123.45.
Actual Result	- dateTimeFormat returns 123.45.
Status	Pass
Severity	Medium

Test Case ID	CFVR-006
Title	Verify dateTimeFormat returns boolean value for boolean type
Objective	Confirm dateTimeFormat returns correct boolean when default_answer is true and type is 'Switch'.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate a data object with default_answer=true, type='Switch'. 2. Create anonymous CustomFieldValueResource class. 3. Call dateTimeFormat. 4. Verify output is true.
Test Data	- Input: data object (default_answer=true, type='Switch', company_id=1) - Expected value: true
Expected Result	- dateTimeFormat returns true.
Actual Result	- dateTimeFormat returns true.
Status	Pass
Severity	Medium

Test Case ID	CFVR-007
Title	Verify CustomFieldValueResource extends JsonResource
Objective	Validate inheritance; CustomFieldValueResource class extends Laravel's JsonResource.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate a data object. 2. Create instance of CustomFieldValueResource. 3. Verify instance is of type JsonResource.
Test Data	- Input: data object (id=1) - Expected value: instance is of type Illuminate\Http\Resources\Json\JsonResource
Expected Result	- Instantiated object is an instance of JsonResource.
Actual Result	- Instantiated object is an instance of JsonResource.
Status	Pass

Severity	Medium

Test Case ID	CFVR-008
Title	Verify CustomFieldValueResource can be instantiated
Objective	Confirm object instantiation of CustomFieldValueResource with typical field types.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate a data object with id and type. 2. Create instance of CustomFieldValueResource. 3. Verify object is instance of CustomFieldValueResource.
Test Data	- Input: data object (id=1, type='Input') - Expected value: instance is of type CustomFieldValueResource
Expected Result	- Instantiated object is an instance of CustomFieldValueResource.
Actual Result	- Instantiated object is an instance of CustomFieldValueResource.
Status	Pass
Severity	Medium

Test Case ID	CFVR-009
Title	Verify toArray handles different custom field types
Objective	Validate the toArray output for all supported custom field types.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Define array of supported custom field types. 2. Loop through each type: a. Instantiate a data object with type set. b. Create anonymous CustomFieldValueResource class. c. Call toArray with dummy request. d. Verify output array’s ‘type’ field matches input type.
Test Data	- Input: types = ['Input', 'TextArea', 'Phone', 'Number', 'Dropdown', 'Switch', 'Date', 'Time', 'DateTime'] - Expected: For each, output['type'] = type
Expected Result	- For each type, toArray returns array with 'type' equal to input value.
Actual Result	- For each type, output matches expected 'type'.
Status	Pass
Severity	Medium

Test Case ID	CFVR-010
Title	Verify toArray handles various ID values
Objective	Check toArray correctly outputs various ID values for custom field, valuable object, and company.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate data object with high numeric values for ids. 2. Create anonymous CustomFieldValueResource class. 3. Call toArray with sample request. 4. Verify returned array’s id, custom_field_valuable_id, custom_field_id, and company_id.

Test Data	- Input: data object (id=999, custom_field_valuable_id=888, custom_field_id=777, company_id=666, type='Input') - Expected: - output['id'] = 999 - output['custom_field_valuable_id'] = 888 - output['custom_field_id'] = 777 - output['company_id'] = 666
Expected Result	- Returned array contains specified id values, matching input.
Actual Result	- Array returned matches expected id values.
Status	Pass
Severity	Medium

Test Case ID	CFVR-011
Title	Verify resource handles minimal data object
Objective	Ensure toArray functions with minimal data (only id supplied).
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate data object with only 'id' field set. 2. Create anonymous CustomFieldValueResource class. 3. Call toArray with sample request. 4. Verify output contains only the 'id' field.
Test Data	- Input: data object (id=1) - Expected: output = ['id' => 1]
Expected Result	- Returned array contains only the 'id' field.
Actual Result	- Array returned contains only the 'id' field with value 1.
Status	Pass
Severity	Low

Test Case ID	CFVR-012
Title	Verify multiple resource instances work independently
Objective	Confirm multiple CustomFieldValueResource instances do not share data and work independently.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate two separate data objects with different id and string_answer values. 2. Create two separate anonymous CustomFieldValueResource instances. 3. Call toArray on both resources. 4. Verify first resource output matches first data; second output matches second.
Test Data	- Input: - data1: id=1, string_answer='First' - data2: id=2, string_answer='Second' - Expected: - resource1: output['id']=1, output['string_answer']='First' - resource2: output['id']=2, output['string_answer']='Second'
Expected Result	- Each resource output matches its respective input data.
Actual Result	- Outputs returned for both resources match respective input data.
Status	Pass
Severity	Medium

Test Case ID	CFVR-013

Title	Verify resource toArray method is callable
Objective	Ensure CustomFieldValueResource has callable toArray method.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate data object. 2. Instantiate CustomFieldValueResource. 3. Assert that toArray method exists. 4. Assert that toArray method is callable.
Test Data	- Input: data object (id=1) - Expected: - method_exists($resource, 'toArray') = true - is_callable([$resource, 'toArray']) = true
Expected Result	- toArray method exists and is callable.
Actual Result	- toArray method exists and is callable.
Status	Pass
Severity	Medium

Test Case ID	CFVR-014
Title	Verify resource has dateTimeFormat method
Objective	Ensure CustomFieldValueResource provides dateTimeFormat method.
Preconditions	- Application running - CustomFieldValueResource class loaded
Test Steps	1. Instantiate data object. 2. Instantiate CustomFieldValueResource. 3. Assert that dateTimeFormat method exists on the instance.
Test Data	- Input: data object (id=1) - Expected: method_exists($resource, 'dateTimeFormat') = true
Expected Result	- dateTimeFormat method exists.
Actual Result	- dateTimeFormat method exists.
Status	Pass
Severity	Medium

File: CustomPathGenerator-Test.txt

Test Case ID	CPG-001
Title	Verify CustomPathGenerator Implements PathGenerator Interface
Objective	To ensure that the CustomPathGenerator class implements the PathGenerator interface.
Preconditions	- Application running - CustomPathGenerator class available - PathGenerator interface is defined
Test Steps	1. Instantiate a new CustomPathGenerator object. 2. Check if the object is an instance of PathGenerator.
Test Data	- Class: CustomPathGenerator
Expected Result	- The CustomPathGenerator instance is recognized as an instance of PathGenerator.
Actual Result	- The CustomPathGenerator instance is recognized as an instance of PathGenerator.
Status	Pass
Severity	Medium

Test Case ID	CPG-002
Title	CustomPathGenerator Can Be Instantiated
Objective	To verify that CustomPathGenerator can be instantiated without error.
Preconditions	- Application running - CustomPathGenerator class available
Test Steps	1. Attempt to instantiate a new CustomPathGenerator object. 2. Confirm the object is an instance of CustomPathGenerator.
Test Data	- Class: CustomPathGenerator
Expected Result	- The new object is an instance of CustomPathGenerator.
Actual Result	- The new object is an instance of CustomPathGenerator.
Status	Pass
Severity	Medium

Test Case ID	CPG-003
Title	getPath Returns Correct Path for Invoice Model
Objective	To confirm getPath() returns the string 'Invoices/' for Invoice model type.
Preconditions	- Application running - Invoice model is defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with model type Invoice::class and key 123. 3. Call getPath() with the media object.
Test Data	- model_type: Invoice::class - key: 123
Expected Result	- getPath() returns 'Invoices/'
Actual Result	- getPath() returns 'Invoices/'
Status	Pass
Severity	High

Test Case ID	CPG-004

Title	getPath Returns Correct Path for Estimate Model
Objective	To confirm getPath() returns the string 'Estimates/' for Estimate model type.
Preconditions	- Application running - Estimate model is defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with model type Estimate::class and key 456. 3. Call getPath() with the media object.
Test Data	- model_type: Estimate::class - key: 456
Expected Result	- getPath() returns 'Estimates/'
Actual Result	- getPath() returns 'Estimates/'
Status	Pass
Severity	High

Test Case ID	CPG-005
Title	getPath Returns Correct Path for Payment Model
Objective	To confirm getPath() returns the string 'Payments/' for Payment model type.
Preconditions	- Application running - Payment model is defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with model type Payment::class and key 789. 3. Call getPath() with the media object.
Test Data	- model_type: Payment::class - key: 789
Expected Result	- getPath() returns 'Payments/'
Actual Result	- getPath() returns 'Payments/'
Status	Pass
Severity	High

Test Case ID	CPG-006
Title	getPath Returns Media Key Path for Unknown Model Type
Objective	To confirm getPath() returns a path using the media key for unknown model types.
Preconditions	- Application running
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with model_type 'App\Models\User' and key 999. 3. Call getPath() with the media object.
Test Data	- model_type: 'App\Models\User' - key: 999
Expected Result	- getPath() returns '999/'
Actual Result	- getPath() returns '999/'
Status	Pass
Severity	Medium

Test Case ID	CPG-007
Title	getPath Handles Null Model Type
Objective	To verify getPath() correctly handles a media object with a null model_type.
Preconditions	- Application running

Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with model_type null and key 111. 3. Call getPath() with the media object.
Test Data	- model_type: null - key: 111
Expected Result	- getPath() returns '111/'
Actual Result	- getPath() returns '111/'
Status	Pass
Severity	Medium

Test Case ID	CPG-008
Title	getPathForConversions Returns Correct Path for Invoice Model
Objective	To verify getPathForConversions() returns 'Invoices/conversations/' for Invoice media.
Preconditions	- Application running - Invoice model is defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with Invoice::class and key 123. 3. Call getPathForConversions() with the media object.
Test Data	- model_type: Invoice::class - key: 123
Expected Result	- getPathForConversions() returns 'Invoices/conversations/'
Actual Result	- getPathForConversions() returns 'Invoices/conversations/'
Status	Pass
Severity	High

Test Case ID	CPG-009
Title	getPathForConversions Returns Correct Path for Estimate Model
Objective	To verify getPathForConversions() returns 'Estimates/conversations/' for Estimate media.
Preconditions	- Application running - Estimate model defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with Estimate::class and key 456. 3. Call getPathForConversions() with the media object.
Test Data	- model_type: Estimate::class - key: 456
Expected Result	- getPathForConversions() returns 'Estimates/conversations/'
Actual Result	- getPathForConversions() returns 'Estimates/conversations/'
Status	Pass
Severity	High

Test Case ID	CPG-010
Title	getPathForConversions Returns Correct Path for Payment Model
Objective	To verify getPathForConversions() returns 'Payments/conversations/' for Payment media.
Preconditions	- Application running - Payment model defined

Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with Payment::class and key 789. 3. Call getPathForConversions() with the media object.
Test Data	- model_type: Payment::class - key: 789
Expected Result	- getPathForConversions() returns 'Payments/conversations/'
Actual Result	- getPathForConversions() returns 'Payments/conversations/'
Status	Pass
Severity	High

Test Case ID	CPG-011
Title	getPathForConversions Returns Media Key Path for Unknown Model
Objective	To confirm getPathForConversions() returns the media key path for an unknown model.
Preconditions	- Application running
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with model_type 'App\Models\Product' and key 555. 3. Call getPathForConversions() with the media object.
Test Data	- model_type: 'App\Models\Product' - key: 555
Expected Result	- getPathForConversions() returns '555/conversations/'
Actual Result	- getPathForConversions() returns '555/conversations/'
Status	Pass
Severity	Medium

Test Case ID	CPG-012
Title	getPathForResponsiveImages Returns Correct Path for Invoice Model
Objective	To ensure getPathForResponsiveImages() returns 'Invoices/responsive-images/' for Invoice media.
Preconditions	- Application running - Invoice model defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with Invoice::class and key 123. 3. Call getPathForResponsiveImages() with the media object.
Test Data	- model_type: Invoice::class - key: 123
Expected Result	- getPathForResponsiveImages() returns 'Invoices/responsive-images/'
Actual Result	- getPathForResponsiveImages() returns 'Invoices/responsive-images/'
Status	Pass
Severity	High

Test Case ID	CPG-013
Title	getPathForResponsiveImages Returns Correct Path for Estimate Model
Objective	To ensure getPathForResponsiveImages() returns 'Estimates/responsive-images/' for Estimate media.
Preconditions	- Application running - Estimate model defined

Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with Estimate::class and key 456. 3. Call getPathForResponsiveImages() with the media object.
Test Data	- model_type: Estimate::class - key: 456
Expected Result	- getPathForResponsiveImages() returns 'Estimates/responsive-images/'
Actual Result	- getPathForResponsiveImages() returns 'Estimates/responsive-images/'
Status	Pass
Severity	High

Test Case ID	CPG-014
Title	getPathForResponsiveImages Returns Correct Path for Payment Model
Objective	To ensure getPathForResponsiveImages() returns 'Payments/responsive-images/' for Payment media.
Preconditions	- Application running - Payment model defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with Payment::class and key 789. 3. Call getPathForResponsiveImages() with the media object.
Test Data	- model_type: Payment::class - key: 789
Expected Result	- getPathForResponsiveImages() returns 'Payments/responsive-images/'
Actual Result	- getPathForResponsiveImages() returns 'Payments/responsive-images/'
Status	Pass
Severity	High

Test Case ID	CPG-015
Title	getPathForResponsiveImages Returns Media Key Path for Unknown Model
Objective	To verify getPathForResponsiveImages() uses media key for unknown model types.
Preconditions	- Application running
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with model_type 'App\Models\Comment' and key 777. 3. Call getPathForResponsiveImages() with the media object.
Test Data	- model_type: 'App\Models\Comment' - key: 777
Expected Result	- getPathForResponsiveImages() returns '777/responsive-images/'
Actual Result	- getPathForResponsiveImages() returns '777/responsive-images/'
Status	Pass
Severity	Medium

Test Case ID	CPG-016
Title	All Path Methods Work Consistently with Same Media Object
Objective	To verify that getPath, getPathForConversions, and getPathForResponsiveImages return expected paths for a single Invoice media object.
Preconditions	- Application running - Invoice model defined

Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with Invoice::class and key 100. 3. Call getPath(), getPathForConversions(), and getPathForResponsiveImages() with the media object.
Test Data	- model_type: Invoice::class - key: 100
Expected Result	- getPath() returns 'Invoices/' - getPathForConversions() returns 'Invoices/conversations/' - getPathForResponsiveImages() returns 'Invoices/responsive-images/'
Actual Result	- getPath() returns 'Invoices/' - getPathForConversions() returns 'Invoices/conversations/' - getPathForResponsiveImages() returns 'Invoices/responsive-images/'
Status	Pass
Severity	High

Test Case ID	CPG-017
Title	Methods Use Media Key Correctly for Unknown Models
Objective	To ensure getPath() uses the correct key for unknown model types.
Preconditions	- Application running
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with model_type 'App\Models\User' and key 1. 3. Create another media object with model_type 'App\Models\Post' and key 2. 4. Call getPath() with both media objects.
Test Data	- media1: model_type: 'App\Models\User', key: 1 - media2: model_type: 'App\Models\Post', key: 2
Expected Result	- getPath(media1) returns '1/' - getPath(media2) returns '2/'
Actual Result	- getPath(media1) returns '1/' - getPath(media2) returns '2/'
Status	Pass
Severity	Medium

Test Case ID	CPG-018
Title	All Path Methods Return Paths Ending with Slash
Objective	To ensure all path methods return strings ending with '/'.
Preconditions	- Application running - Invoice model defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a test media object with Invoice::class and key 123. 3. Call getPath(), getPathForConversions(), and getPathForResponsiveImages() with the media object. 4. Check if returned paths end with '/'
Test Data	- model_type: Invoice::class - key: 123
Expected Result	- getPath() ends with '/' - getPathForConversions() ends with '/' - getPathForResponsiveImages() ends with '/'
Actual Result	- getPath(), getPathForConversions(), and getPathForResponsiveImages() all end with '/'
Status	Pass
Severity	Medium

Test Case ID	CPG-019
Title	Generator Handles Multiple Model Types Correctly
Objective	To confirm getPath() returns correct folders for multiple standard model types.
Preconditions	- Application running - Invoice, Estimate, Payment models defined
Test Steps	1. Instantiate CustomPathGenerator. 2. For each model_type in [Invoice::class, Estimate::class, Payment::class], create media with key 999. 3. Call getPath() for each and verify the result.
Test Data	- Invoice::class => 'Invoices/' - Estimate::class => 'Estimates/' - Payment::class => 'Payments/'
Expected Result	- getPath(Invoice) returns 'Invoices/' - getPath(Estimate) returns 'Estimates/' - getPath(Payment) returns 'Payments/'
Actual Result	- getPath(Invoice) returns 'Invoices/' - getPath(Estimate) returns 'Estimates/' - getPath(Payment) returns 'Payments/'
Status	Pass
Severity	High

Test Case ID	CPG-020
Title	Generator Instance Can Be Reused for Multiple Media Objects
Objective	To verify the same CustomPathGenerator instance can process multiple media objects.
Preconditions	- Application running - Invoice and Estimate models defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create media objects: one with Invoice::class and key 1; another with Estimate::class and key 2. 3. Call getPath() for both media objects.
Test Data	- media1: model_type: Invoice::class, key: 1 - media2: model_type: Estimate::class, key: 2
Expected Result	- getPath(media1) returns 'Invoices/' - getPath(media2) returns 'Estimates/'
Actual Result	- getPath(media1) returns 'Invoices/' - getPath(media2) returns 'Estimates/'
Status	Pass
Severity	Medium

Test Case ID	CPG-021
Title	Generator Has All Required Methods
Objective	To confirm the existence of all core path methods in CustomPathGenerator.
Preconditions	- Application running - CustomPathGenerator class defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Verify that the methods 'getPath', 'getPathForConversions', and 'getPathForResponsiveImages' exist.
Test Data	- Class: CustomPathGenerator
Expected Result	- 'getPath' method exists. - 'getPathForConversions' method exists. - 'getPathForResponsiveImages' method exists.

Actual Result	- All required methods exist.
Status	Pass
Severity	High

Test Case ID	CPG-022
Title	Generator Handles String Media Keys
Objective	To verify getPath() properly handles and returns string keys in the folder path.
Preconditions	- Application running
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a media object with model_type 'App\Models\Article' and key 'abc-123'. 3. Call getPath() with the media object.
Test Data	- model_type: 'App\Models\Article' - key: 'abc-123'
Expected Result	- getPath() returns 'abc-123/'
Actual Result	- getPath() returns 'abc-123/'
Status	Pass
Severity	Medium

Test Case ID	CPG-023
Title	Generator Handles Large Media Keys
Objective	To ensure getPath() can process and return large integer media keys.
Preconditions	- Application running
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a media object with model_type 'App\Models\File' and key 999999999. 3. Call getPath() with the media object.
Test Data	- model_type: 'App\Models\File' - key: 999999999
Expected Result	- getPath() returns '999999999/'
Actual Result	- getPath() returns '999999999/'
Status	Pass
Severity	Medium

Test Case ID	CPG-024
Title	Conversations Path Has Correct Structure
Objective	To verify getPathForConversions() returns a path containing 'conversations' and starts with the model folder.
Preconditions	- Application running - Invoice model defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a media object with Invoice::class and key 100. 3. Call getPathForConversions() with the media object. 4. Check that path contains 'conversations' and starts with 'Invoices/'.
Test Data	- model_type: Invoice::class - key: 100
Expected Result	- Path returned by getPathForConversions() contains 'conversations'. - Path starts with 'Invoices/'.
Actual Result	- Path contains 'conversations' and starts with 'Invoices/'.

Status	Pass
Severity	Medium

Test Case ID	CPG-025
Title	Responsive Images Path Has Correct Structure
Objective	To verify getPathForResponsiveImages() contains 'responsive-images' and starts with the model folder.
Preconditions	- Application running - Payment model defined
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a media object with Payment::class and key 200. 3. Call getPathForResponsiveImages() with the media object. 4. Check that path contains 'responsive-images' and starts with 'Payments/'.
Test Data	- model_type: Payment::class - key: 200
Expected Result	- Path returned contains 'responsive-images'. - Path starts with 'Payments/'.
Actual Result	- Path contains 'responsive-images' and starts with 'Payments/'.
Status	Pass
Severity	Medium

Test Case ID	CPG-026
Title	Generator Handles Zero as Media Key
Objective	To ensure getPath() returns '0/' for media objects with key 0 and unknown model.
Preconditions	- Application running
Test Steps	1. Instantiate CustomPathGenerator. 2. Create a media object with model_type 'App\Models\Test' and key 0. 3. Call getPath() with the media object.
Test Data	- model_type: 'App\Models\Test' - key: 0
Expected Result	- getPath() returns '0/'
Actual Result	- getPath() returns '0/'
Status	Pass
Severity	Medium

File: CustomerCollection-Test.txt

Test Case ID	CC-001
Title	toArray returns an empty array when initialized with an empty collection
Objective	Verify that the TestCustomerCollection::toArray method returns an empty array when given an empty collection.
Preconditions	- Application running - Database seeded - Collection instance is empty
Test Steps	1. Initialize a DummyRequest object. 2. Create an empty Illuminate\Database\Eloquent\Collection. 3. Instantiate TestCustomerCollection with the empty collection. 4. Call toArray on TestCustomerCollection passing DummyRequest.
Test Data	- Collection: [] (empty) - Request: DummyRequest with no parameters - Expected values: []
Expected Result	TestCustomerCollection::toArray returns an empty array ([]).
Actual Result	TestCustomerCollection::toArray returned an empty array ([]).
Status	Pass
Severity	Medium

Test Case ID	CC-002
Title	toArray transforms a collection of simple objects using default behavior
Objective	Verify that TestCustomerCollection::toArray correctly transforms a collection of TestCustomerModel objects into an array of associative arrays.
Preconditions	- Application running - Database seeded - TestCustomerModel objects available
Test Steps	1. Initialize DummyRequest object. 2. Create two TestCustomerModel objects with specific data. 3. Add both models to a Collection. 4. Instantiate TestCustomerCollection with the Collection. 5. Call toArray on TestCustomerCollection passing DummyRequest.
Test Data	- TestCustomerModel #1: id: 1, name: "Customer A", email: "a@example.com", status: "active", phone: "1234567890" - TestCustomerModel #2: id: 2, name: "Customer B", email: "b@example.com", status: "inactive", phone: "0987654321" - Expected values: [['id'=>1, 'name'=>'Customer A', 'email'=>'a@example.com', 'status'=>'active', 'phone'=>'1234567890'], ['id'=>2, 'name'=>'Customer B', 'email'=>'b@example.com', 'status'=>'inactive', 'phone'=>'0987654321']]
Expected Result	toArray returns an array of the two models' data in expected structure.
Actual Result	toArray returned the correct array structure for both customer objects.
Status	Pass
Severity	Medium

Test Case ID	CC-003
Title	toArray processes a collection of JsonResource instances

Objective	Validate that TestCustomerCollection::toArray correctly transforms a Collection of TestCustomerResource instances into the expected array.
Preconditions	- Application running - Database seeded - TestCustomerResource instances available
Test Steps	1. Initialize DummyRequest object. 2. Create two TestCustomerModel objects and wrap each in a TestCustomerResource. 3. Add both resources to a Collection. 4. Instantiate TestCustomerCollection with the Collection. 5. Call toArray on TestCustomerCollection passing DummyRequest.
Test Data	- TestCustomerModel #1: id: 101, name: "Alpha Customer", email: "alpha@example.com", status: "active", phone: "1111111111" - TestCustomerModel #2: id: 102, name: "Beta Customer", email: "beta@example.com", status: "active", phone: "2222222222" - Expected values: [['id'=>101, 'name'=>'Alpha Customer', 'email'=>'alpha@example.com', 'status'=>'active', 'phone'=>'1111111111'], ['id'=>102, 'name'=>'Beta Customer', 'email'=>'beta@example.com', 'status'=>'active', 'phone'=>'2222222222']]
Expected Result	toArray returns the expected array structure for wrapped resource objects.
Actual Result	toArray returned data for both resources matching expected array.
Status	Pass
Severity	Medium

Test Case ID	CC-004
Title	toArray passes request object to nested transformations for custom resource types
Objective	Ensure that TestCustomerCollection::toArray correctly passes the request to custom nested JsonResource objects, and that custom keys from the request are appended.
Preconditions	- Application running - Database seeded - Request object contains extra parameter - Custom JsonResource is used
Test Steps	1. Initialize DummyRequest with parameter: append_param = "extra_data". 2. Create a TestCustomerModel with specified attributes. 3. Define a custom JsonResource that appends 'appended' field from request if present. 4. Add the custom resource to a Collection. 5. Instantiate TestCustomerCollection with the Collection. 6. Call toArray on TestCustomerCollection passing custom DummyRequest.
Test Data	- Model: id: 1, name: "Custom Customer", email: "custom@example.com", status: "active", phone: "5555555555" - DummyRequest: ['append_param' => 'extra_data'] - Expected values: [['id' => 1, 'name' => 'Custom Customer', 'email' => 'custom@example.com', 'status' => 'active', 'phone' => '5555555555', 'appended' => 'extra_data',]]

Expected Result	toArray returns model data with 'appended' field sourced from DummyRequest.
Actual Result	toArray returned expected model data with appended key/value.
Status	Pass
Severity	Medium

Test Case ID	CC-005
Title	toArray handles an empty and valid request object
Objective	Confirm toArray transforms a collection with a single TestCustomerModel and an empty DummyRequest properly.
Preconditions	- Application running - Database seeded - Empty DummyRequest provided
Test Steps	1. Initialize DummyRequest with no parameters. 2. Create TestCustomerModel with attributes. 3. Clone model into separate instance for clarity. 4. Add model to Collection. 5. Instantiate TestCustomerCollection with Collection. 6. Call toArray with DummyRequest.
Test Data	- Model: id: 3, name: "Customer C", email: "c@example.com", status: "active", phone: "3333333333" - DummyRequest: {} - Expected values: [['id' => 3, 'name' => 'Customer C', 'email' => 'c@example.com', 'status' => 'active', 'phone' => '3333333333',]]
Expected Result	toArray returns correct single element array with model data.
Actual Result	toArray returned single-element array matching model data.
Status	Pass
Severity	Medium

Test Case ID	CC-006
Title	toArray handles collection of arrays
Objective	Test that TestCustomerCollection::toArray correctly transforms a collection containing raw arrays.
Preconditions	- Application running - Database seeded - Collection contains arrays
Test Steps	1. Initialize DummyRequest. 2. Define two associative arrays with customer data. 3. Add both arrays to Collection. 4. Instantiate TestCustomerCollection with Collection. 5. Call toArray with DummyRequest.
Test Data	- Array #1: id: 10, name: "Arr Customer", email: "arr@example.com", status: "active", phone: "4444444444" - Array #2: id: 11, name: "Arr Customer 2", email: "arr2@example.com", status: "inactive", phone: "5555555555" - Expected values: [array1, array2]
Expected Result	toArray returns a list of arrays matching the input data.

Actual Result	toArray returned correctly the two arrays in output array.
Status	Pass
Severity	Medium

Test Case ID	CC-007
Title	toArray handles collection of stdClass objects
Objective	Verify that toArray transforms a collection of stdClass objects into arrays of property values.
Preconditions	- Application running - Database seeded - Collection contains stdClass instances
Test Steps	1. Initialize DummyRequest. 2. Define two stdClass objects with customer properties. 3. Add both stdClass objects to Collection. 4. Instantiate TestCustomerCollection with Collection. 5. Call toArray with DummyRequest.
Test Data	- stdClass #1: id: 20, name: "Std Customer", email: "std@example.com", status: "active", phone: "6666666666" - stdClass #2: id: 21, name: "Std Customer 2", email: "std2@example.com", status: "inactive", phone: "7777777777" - Expected values: [['id' => 20, 'name' => 'Std Customer', 'email' => 'std@example.com', 'status' => 'active', 'phone' => '6666666666',], ['id' => 21, 'name' => 'Std Customer 2', 'email' => 'std2@example.com', 'status' => 'inactive', 'phone' => '7777777777',],]
Expected Result	toArray returns arrays generated from stdClass property values.
Actual Result	toArray returned arrays matching stdClass property values.
Status	Pass
Severity	Medium

Test Case ID	CC-008
Title	toArray handles collection of mixed types
Objective	Ensure toArray can process a collection with mixed element types (model, array, stdClass, JsonResource).
Preconditions	- Application running - Database seeded - Collection contains mixed types
Test Steps	1. Initialize DummyRequest. 2. Create TestCustomerModel, array, stdClass, and TestCustomerResource objects with customer data. 3. Add all objects to Collection. 4. Instantiate TestCustomerCollection with Collection. 5. Call toArray with DummyRequest.

Test Data	- Model: id: 30, name: 'Model Customer', ... - Array: id: 31, name: 'Array Customer', ... - stdClass: id: 32, name: 'StdClass Customer', ... - Resource: id: 33, name: 'Resource Customer', ... - Expected values: Array of all above customer data in associative array form
Expected Result	toArray returns array of associative arrays for each collection element, matching types and data.
Actual Result	toArray returned correct array for all mixed types.
Status	Pass
Severity	Medium

Test Case ID	CC-009
Title	toArray handles collection of models, arrays, and stdClass with null fields
Objective	Validate that toArray transforms collection elements with null properties correctly.
Preconditions	- Application running - Database seeded - Models, arrays, stdClass objects contain null values
Test Steps	1. Initialize DummyRequest. 2. Prepare TestCustomerModel, array, stdClass all with id but other fields null. 3. Add objects to Collection. 4. Instantiate TestCustomerCollection with Collection. 5. Call toArray with DummyRequest.
Test Data	- Model: id: 40, name: null, email: null, status: null, phone: null - Array: id: 41, name: null, email: null, status: null, phone: null - stdClass: id: 42, name: null, email: null, status: null, phone: null - Expected values: Array of associative arrays with id + null fields
Expected Result	toArray returns array containing all null values (except id) as per input.
Actual Result	toArray returned associative arrays filled correctly with null fields for each input object.
Status	Pass
Severity	Medium

Test Case ID	CC-010
Title	toArray handles collection with only explicit nulls
Objective	Check that toArray returns [null, null] when collection contains only nulls.
Preconditions	- Application running - Database seeded - Collection contains only null elements
Test Steps	1. Initialize DummyRequest. 2. Create Collection containing [null, null]. 3. Instantiate TestCustomerCollection with Collection. 4. Call toArray with DummyRequest.
Test Data	- Collection: [null, null] - Expected values: [null, null]
Expected Result	toArray returns [null, null].
Actual Result	toArray returned [null, null] as expected.
Status	Pass
Severity	Medium

Test Case ID	CC-011
Title	toArray handles collection with mixed null and valid values

Objective	Confirm that toArray returns expected output for a collection with a mix of null and valid models.
Preconditions	- Application running - Database seeded - Collection includes null and TestCustomerModel
Test Steps	1. Initialize DummyRequest. 2. Create TestCustomerModel with actual data. 3. Create Collection as [null, model, null]. 4. Instantiate TestCustomerCollection with Collection. 5. Call toArray with DummyRequest.
Test Data	- Collection: [null, model:{id:50, name:'Null Mix', email:'nullmix@example.com', status:'active', phone:'5555555555'}, null] - Expected values: [null, {id:50, name:'Null Mix', email:'nullmix@example.com', status:'active', phone:'5555555555'}, null]
Expected Result	toArray returns array with null at first and last index and model data at center index.
Actual Result	toArray returned [null, , null], where matches model data.
Status	Pass
Severity	Medium

Test Case ID	CC-012
Title	TestCustomerModel toArray returns correct array
Objective	Ensure that calling toArray on TestCustomerModel produces an array with correct properties.
Preconditions	- Application running - Database seeded - TestCustomerModel instance available
Test Steps	1. Create TestCustomerModel with specified attributes. 2. Call toArray on the model.
Test Data	- Model: id: 123, name: 'Test Name', email: 'test@example.com', status: 'inactive', phone: '555-5555' - Expected values: ['id'=>123, 'name'=>'Test Name', 'email'=>'test@example.com', 'status'=>'inactive', 'phone'=>'555-5555']
Expected Result	toArray returns associative array matching model properties.
Actual Result	toArray returned associative array matching input data.
Status	Pass
Severity	Low

Test Case ID	CC-013
Title	TestCustomerResource toArray returns correct array
Objective	Validate that TestCustomerResource::toArray returns the correct array when wrapping a TestCustomerModel.
Preconditions	- Application running - Database seeded - TestCustomerModel and TestCustomerResource available
Test Steps	1. Create TestCustomerModel with attributes. 2. Create TestCustomerResource passing model. 3. Call toArray on resource with DummyRequest.
Test Data	- Model: id: 321, name: 'Resource Name', email: 'resource@example.com', status: 'active', phone: '777-7777' - Expected values: ['id'=>321, 'name'=>'Resource Name', 'email'=>'resource@example.com', 'status'=>'active', 'phone'=>'777-7777']

Expected Result	toArray returns associative array matching wrapped model's properties.
Actual Result	toArray returned array matching model properties.
Status	Pass
Severity	Low

Test Case ID	CC-014
Title	TestCustomerCollection toArray returns correct array for single model
Objective	Confirm that TestCustomerCollection::toArray converts a single model in a collection to a one-element array.
Preconditions	- Application running - Database seeded - Collection contains single TestCustomerModel
Test Steps	1. Create TestCustomerModel with customer attributes. 2. Add to Collection. 3. Instantiate TestCustomerCollection with Collection. 4. Call toArray with DummyRequest.
Test Data	- Model: id: 555, name: 'Single Model', email: 'single@example.com', status: 'active', phone: '1231231234' - Expected values: [['id' => 555, 'name' => 'Single Model', 'email' => 'single@example.com', 'status' => 'active', 'phone' => '1231231234',]]
Expected Result	toArray returns single-element array with model properties.
Actual Result	toArray returned array matching model properties.
Status	Pass
Severity	Low

Test Case ID	CC-015
Title	TestCustomerCollection toArray returns correct array for empty collection
Objective	Validate that toArray on TestCustomerCollection returns empty array when collection is empty.
Preconditions	- Application running - Database seeded - Collection is empty
Test Steps	1. Create empty Collection. 2. Instantiate TestCustomerCollection with Collection. 3. Call toArray with DummyRequest.
Test Data	- Collection: [] - Expected values: []
Expected Result	toArray returns empty array.
Actual Result	toArray returned empty array.
Status	Pass
Severity	Low

Test Case ID	CC-016
Title	DummyRequest has, get, all methods function correctly

Objective	Verify that DummyRequest provides correct outputs for has(), get(), and all() methods.
Preconditions	- Application running - DummyRequest initialized with test parameters
Test Steps	1. Create DummyRequest with parameters: ['foo' => 'bar', 'baz' => 'qux']. 2. Call has('foo'), has('baz'), has('notfound'). 3. Call get('foo'), get('baz'), get('notfound', 'default'). 4. Call all(), all(['foo']), and all(['baz','notfound']).
Test Data	- DummyRequest with ['foo' => 'bar', 'baz' => 'qux'] - Expected values: - has('foo'): true - has('baz'): true - has('notfound'): false - get('foo'): 'bar' - get('baz'): 'qux' - get('notfound', 'default'): 'default' - all(): ['foo'=>'bar','baz'=>'qux'] - all(['foo']): ['foo'=>'bar'] - all(['baz','notfound']): ['baz'=>'qux', 'notfound'=>null]
Expected Result	All DummyRequest accessor methods return expected values.
Actual Result	All DummyRequest methods returned correct values as described.
Status	Pass
Severity	Low

Test Case ID	CC-017
Title	TestCustomerModel handles missing keys in constructor
Objective	Ensure TestCustomerModel assigns default values for missing keys in the constructor.
Preconditions	- Application running - No override array provided to constructor
Test Steps	1. Create TestCustomerModel with empty array ([]). 2. Verify the default property values.
Test Data	- Input: [] - Expected values: - id: 999 - name: 'Dummy' - email: 'dummy@example.com' - status: 'active' - phone: null
Expected Result	Model instance properties match default values.
Actual Result	All properties contained correct default values.
Status	Pass
Severity	Low

Test Case ID	CC-018
Title	TestCustomerCollection handles collection with only nulls
Objective	Validate TestCustomerCollection::toArray returns array of nulls when every collection element is null.
Preconditions	- Application running - Collection with only nulls
Test Steps	1. Create Collection: [null, null, null]. 2. Instantiate TestCustomerCollection with Collection. 3. Call toArray with DummyRequest.

Test Data	- Collection: [null, null, null] - Expected values: [null, null, null]
Expected Result	toArray returns [null, null, null].
Actual Result	toArray returned [null, null, null].
Status	Pass
Severity	Medium

File: CustomerEstimateStatusRequest-Test.txt

Test Case ID	CESR-001
Title	Authorize method always grants access
Objective	Verify CustomerEstimateStatusRequest::authorize() returns true, ensuring request authorization.
Preconditions	- Application running - Required PHP dependencies installed - CustomerEstimateStatusRequest class available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Call the authorize() method. 3. Assert that the returned value is true.
Test Data	- No input data required.
Expected Result	- authorize() returns true.
Actual Result	authorize() returned true.
Status	Pass
Severity	High

Test Case ID	CESR-002
Title	Validation rules include 'status' field with correct structure
Objective	Verify rules() returns array with 'status' key, and correct validation rules for status.
Preconditions	- Application running - CustomerEstimateStatusRequest class available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Call the rules() method. 3. Assert 'status' key exists in returned array. 4. Assert rules for 'status' is an array. 5. Assert 'required' exists in rules for 'status'. 6. Assert rules for 'status' include the string 'in:ACCEPTED,REJECTED'.
Test Data	- No input data required.
Expected Result	- 'status' key exists in rules array. - Rules for 'status' is an array. - Rule 'required' present in array. - Rules include 'in:ACCEPTED,REJECTED'.
Actual Result	All expected rules and keys are present.
Status	Pass
Severity	High

Test Case ID	CESR-003
Title	Validation rules only specify 'status' key
Objective	Confirm rules() returns an array containing only the 'status' key.
Preconditions	- Application running - CustomerEstimateStatusRequest class available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Call the rules() method. 3. Assert only one key exists in rules array. 4. Assert that the key is 'status'.
Test Data	- No input data required.
Expected Result	- Rules array contains exactly one key: 'status'.
Actual Result	Rules array only contains the 'status' key.

Status	Pass
Severity	Medium

Test Case ID	CESR-004
Title	Validation passes with status set to 'ACCEPTED'
Objective	Ensure validation is successful when status is 'ACCEPTED'.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: 'status' => 'ACCEPTED'. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns false. 5. Assert validator passes() returns true.
Test Data	- Input: ['status' => 'ACCEPTED']
Expected Result	- Validation does not fail. - Validation passes.
Actual Result	Validation passed as expected.
Status	Pass
Severity	High

Test Case ID	CESR-005
Title	Validation passes with status set to 'REJECTED'
Objective	Ensure validation is successful when status is 'REJECTED'.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: 'status' => 'REJECTED'. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns false. 5. Assert validator passes() returns true.
Test Data	- Input: ['status' => 'REJECTED']
Expected Result	- Validation does not fail. - Validation passes.
Actual Result	Validation passed as expected.
Status	Pass
Severity	High

Test Case ID	CESR-006
Title	Validation fails when status is missing
Objective	Verify validation rejects requests missing status field.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare empty data: []. 3. Run Validator::make with empty data and rules(). 4. Assert validator fails() returns true. 5. Assert errors include the 'status' key.
Test Data	- Input: []
Expected Result	- Validation fails. - Errors contain 'status' key.

Actual Result	Validation failed and errors include 'status'.
Status	Pass
Severity	High

Test Case ID	CESR-007
Title	Validation fails with invalid status value
Objective	Confirm validation rejects values for status other than 'ACCEPTED' or 'REJECTED'.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: ['status' => 'PENDING']. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns true. 5. Assert errors include the 'status' key.
Test Data	- Input: ['status' => 'PENDING']
Expected Result	- Validation fails. - Errors contain 'status' key.
Actual Result	Validation failed and 'status' key is present in errors.
Status	Pass
Severity	High

Test Case ID	CESR-008
Title	Validation fails with empty string status
Objective	Ensure validation rejects empty string as a status value.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: ['status' => '']. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns true. 5. Assert errors include the 'status' key.
Test Data	- Input: ['status' => '']
Expected Result	- Validation fails. - Errors contain 'status' key.
Actual Result	Validation failed and 'status' key is present in errors.
Status	Pass
Severity	High

Test Case ID	CESR-009
Title	Validation fails with null status
Objective	Ensure validation rejects null as a status value.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: ['status' => null]. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns true. 5. Assert errors include the 'status' key.
Test Data	- Input: ['status' => null]

Expected Result	- Validation fails. - Errors contain 'status' key.
Actual Result	Validation failed and 'status' key is present in errors.
Status	Pass
Severity	High

Test Case ID	CESR-010
Title	Validation fails with numeric status value
Objective	Confirm validation rejects numeric status values.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: ['status' => 123]. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns true. 5. Assert errors include the 'status' key.
Test Data	- Input: ['status' => 123]
Expected Result	- Validation fails. - Errors contain 'status' key.
Actual Result	Validation failed and 'status' key is present in errors.
Status	Pass
Severity	High

Test Case ID	CESR-011
Title	Validation fails with lowercase accepted value
Objective	Ensure validation is case sensitive for status value 'ACCEPTED'.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: ['status' => 'accepted']. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns true. 5. Assert errors include the 'status' key.
Test Data	- Input: ['status' => 'accepted']
Expected Result	- Validation fails. - Errors contain 'status' key.
Actual Result	Validation failed and 'status' is in errors.
Status	Pass
Severity	High

Test Case ID	CESR-012
Title	Validation fails with mixed case rejected value
Objective	Confirm validation is case sensitive for status value 'REJECTED'.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: ['status' => 'Rejected']. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns true. 5. Assert errors include the 'status' key.

Test Data	- Input: ['status' => 'Rejected']
Expected Result	- Validation fails. - Errors contain 'status' key.
Actual Result	Validation failed and 'status' is in errors.
Status	Pass
Severity	High

Test Case ID	CESR-013
Title	Extra fields do not affect status validation
Objective	Verify that extra fields are ignored if status is valid.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: - 'status' => 'ACCEPTED' - 'extra_field' => 'some_value' - 'another_field' => 123 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns false. 5. Assert validator passes() returns true.
Test Data	- Input: - 'status' => 'ACCEPTED' - 'extra_field' => 'some_value' - 'another_field' => 123
Expected Result	- Validation does not fail. - Validation passes.
Actual Result	Validation passed; extra fields did not interfere.
Status	Pass
Severity	Medium

Test Case ID	CESR-014
Title	Error message is provided when status is missing
Objective	Confirm error message is present and not empty when status is not provided.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare empty data: []. 3. Run Validator::make with empty data and rules(). 4. Assert validator fails() returns true. 5. Assert errors has 'status'. 6. Assert get('status') returns a non-empty error message.
Test Data	- Input: []
Expected Result	- Validator fails. - Errors include 'status'. - Returned error message for 'status' is not empty.
Actual Result	Validator fails; 'status' error message present and not empty.
Status	Pass
Severity	High

Test Case ID	CESR-015
Title	Error message is provided for invalid status value

Objective	Check that error message for status is present and not empty when value is invalid.
Preconditions	- Application running - CustomerEstimateStatusRequest and Validator available
Test Steps	1. Instantiate CustomerEstimateStatusRequest. 2. Prepare data: ['status' => 'INVALID']. 3. Run Validator::make with data and rules(). 4. Assert validator fails() returns true. 5. Assert errors has 'status'. 6. Assert get('status') returns a non-empty error message.
Test Data	- Input: ['status' => 'INVALID']
Expected Result	- Validator fails. - Errors include 'status'. - Returned error message for 'status' is not empty.
Actual Result	Validator fails; 'status' error message present and not empty.
Status	Pass
Severity	High

File: CustomerLoginRequest-Test.txt

Test Case ID	CLR-001
Title	Authorize Method Returns True
Objective	Verify that the authorize() method of CustomerLoginRequest returns true, allowing customer login requests to be authorized.
Preconditions	- Application running - All dependencies loaded - CustomerLoginRequest class available - No special authentication state required
Test Steps	1. Instantiate a CustomerLoginRequest object. 2. Call the authorize() method on the object. 3. Capture the returned result.
Test Data	- CustomerLoginRequest instance (default, no parameters)
Expected Result	- The authorize() method returns true.
Actual Result	The authorize() method returns true as expected.
Status	Pass
Severity	High

Test Case ID	CLR-002
Title	Rules Method Returns Correct Validation Rules
Objective	Verify that the rules() method of CustomerLoginRequest returns the expected validation rules for email and password fields.
Preconditions	- Application running - All dependencies loaded - CustomerLoginRequest class available - No special authentication state required
Test Steps	1. Instantiate a CustomerLoginRequest object. 2. Call the rules() method on the object. 3. Compare the returned array with the expected validation rules.
Test Data	- CustomerLoginRequest instance (default, no parameters) - Expected rules: - 'email' => ['required', 'string'] - 'password' => ['required', 'string']
Expected Result	- The rules() method returns an array: - The 'email' key contains ['required', 'string'] - The 'password' key contains ['required', 'string']
Actual Result	The rules() method returns the expected validation array for both 'email' and 'password'.
Status	Pass
Severity	High

File: CustomerMailResetPasswordNotification-Test.txt

Test Case ID	CMRPN-001
Title	Constructor sets token correctly via parent class
Objective	Verify that the CustomerMailResetPasswordNotification constructor sets the token property in the parent class accurately.
Preconditions	- Application running - CustomerMailResetPasswordNotification class available - PHP Reflection API accessible
Test Steps	1. Instantiate CustomerMailResetPasswordNotification with token 'test-token-123'. 2. Use reflection to access protected token property in parent class. 3. Retrieve value of token property from the notification instance.
Test Data	- Input token: 'test-token-123'
Expected Result	- The token property in the parent class is set to 'test-token-123'.
Actual Result	Token property was correctly set to 'test-token-123'.
Status	Pass
Severity	High

Test Case ID	CMRPN-002
Title	Constructor handles various token formats
Objective	Ensure the constructor properly sets token property for different formats and values.
Preconditions	- Application running - CustomerMailResetPasswordNotification class available
Test Steps	1. Iterate through an array of token formats: - 'simple-token' - 'token_with_underscores' - 'token-with-dashes' - 'TokenWithMixedCase123' - '1234567890' - 'very-long-token-string-with-many-characters-to-test-length-handling' 2. Instantiate CustomerMailResetPasswordNotification with each token. 3. Access the token property using reflection and verify it matches the input.
Test Data	- Token formats (see step 1)
Expected Result	- Token property matches input value for each format.
Actual Result	Token property matched input for all tested formats.
Status	Pass
Severity	High

Test Case ID	CMRPN-003
Title	Via method returns mail channel
Objective	Verify that the via() method returns only the 'mail' channel.
Preconditions	- Application running - CustomerMailResetPasswordNotification class available
Test Steps	1. Instantiate CustomerMailResetPasswordNotification with token 'some-token'. 2. Create a dummy notifiable object. 3. Call via() method with notifiable. 4. Validate returned channels array.
Test Data	- Token: 'some-token' - Notifiable: stdClass instance

Expected Result	- Channels is an array containing only 'mail'. - Array length is 1. - First element is 'mail'.
Actual Result	Channels returned as ['mail'] array.
Status	Pass
Severity	Medium

Test Case ID	CMRPN-004
Title	toMail constructs MailMessage with correct details (standard case)
Objective	Confirm toMail() generates a MailMessage with complete and accurate content for a standard password reset.
Preconditions	- Application running - CustomerMailResetPasswordNotification and MailMessage classes available
Test Steps	1. Set token = 'secure-reset-token' and company slug = 'my-awesome-company'. 2. Create a notifiable object with company slug. 3. Instantiate CustomerMailResetPasswordNotification with token. 4. Call toMail() with notifiable. 5. Validate MailMessage properties: instance type, subject, introLines, actionText, actionUrl, outroLines.
Test Data	- Token: 'secure-reset-token' - Company slug: 'my-awesome-company'
Expected Result	- MailMessage is instance of MailMessage - Subject is 'Reset Password Notification' - First intro line is "Hello! You are receiving this email because we received a password reset request for your account." - Action text is 'Reset Password' - Action URL contains token, company slug, and 'customer/reset/password' - First outro line contains expiry time info - Second outro line is "If you did not request a password reset, no further action is required."
Actual Result	MailMessage included all details as expected for standard reset.
Status	Pass
Severity	High

Test Case ID	CMRPN-005
Title	toMail generates correct URL with different company slugs
Objective	Verify that toMail() embeds the company slug in the reset URL for diverse company slugs.
Preconditions	- Application running - CustomerMailResetPasswordNotification class available
Test Steps	1. Set token = 'test-token'. 2. For each company slug: - 'company-one' - 'company_two' - 'CompanyThree' - '123company' - 'my-test-company-slug' 3. Create notifiable with given slug. 4. Instantiate notification and call toMail(). 5. Validate actionUrl contains slug, token, and 'customer/reset/password'.
Test Data	- Token: 'test-token' - Company slugs as above
Expected Result	- actionUrl contains company slug, token, and 'customer/reset/password' for all cases.

Actual Result	All actionUrls included correct company slug and token.
Status	Pass
Severity	High

Test Case ID	CMRPN-006
Title	toMail handles empty company slug
Objective	Ensure that toMail() works when company slug is the empty string.
Preconditions	- Application running - CustomerMailResetPasswordNotification class available
Test Steps	1. Set token = 'token-with-empty-slug'. 2. Create notifiable with empty string for company slug. 3. Instantiate notification and call toMail(). 4. Check actionUrl for inclusion of token and 'customer/reset/password'.
Test Data	- Token: 'token-with-empty-slug' - Company slug: ''
Expected Result	- actionUrl contains the token and 'customer/reset/password'.
Actual Result	actionUrl included token and reset path even with empty slug.
Status	Pass
Severity	Medium

Test Case ID	CMRPN-007
Title	toMail handles null company slug
Objective	Verify toMail() creates a valid URL if company slug is null.
Preconditions	- Application running - CustomerMailResetPasswordNotification class available
Test Steps	1. Set token = 'token-with-null-slug'. 2. Create notifiable with null company slug. 3. Instantiate notification and call toMail(). 4. Validate actionUrl contains token and 'customer/reset/password'.
Test Data	- Token: 'token-with-null-slug' - Company slug: null
Expected Result	- actionUrl contains token and 'customer/reset/password'.
Actual Result	actionUrl properly included token with null slug.
Status	Pass
Severity	Medium

Test Case ID	CMRPN-008
Title	toMail uses correct token in reset link
Objective	Ensure unique tokens are included in the password reset URLs.
Preconditions	- Application running
Test Steps	1. For each token: - 'unique-token-alpha' - 'distinct-token-beta' - 'another-token-gamma' 2. Create notifiable with company slug 'token-test-company'. 3. Instantiate notification and call toMail(). 4. Validate actionUrl includes the token and company slug.
Test Data	- Tokens as above - Company slug: 'token-test-company'
Expected Result	- actionUrl includes the active token and company slug for each case.

Actual Result	Each reset URL included the correct token and company slug.
Status	Pass
Severity	High

Test Case ID	CMRPN-009
Title	Notification instances maintain independent tokens
Objective	Confirm that multiple notification instances do not share token values.
Preconditions	- Application running
Test Steps	1. Create two notifications with tokens 'token-instance-1' and 'token-instance-2'. 2. Create common notifiable with slug 'test-company'. 3. Call toMail() for both instances. 4. Confirm first reset URL contains only 'token-instance-1', second only 'token-instance-2'.
Test Data	- token1: 'token-instance-1' - token2: 'token-instance-2' - slug: 'test-company'
Expected Result	- First URL contains only token1. - Second URL contains only token2.
Actual Result	Each message referenced only its own unique token.
Status	Pass
Severity	High

Test Case ID	CMRPN-010
Title	toArray method consistently returns an empty array
Objective	Verify toArray() always returns an empty array.
Preconditions	- Application running
Test Steps	1. Instantiate notification with token 'any-token'. 2. Create a notifiable object. 3. Call toArray() on notification with notifiable. 4. Inspect result.
Test Data	- token: 'any-token' - notifiable: stdClass
Expected Result	- Result is an empty array.
Actual Result	Method returned an empty array as expected.
Status	Pass
Severity	Low

Test Case ID	CMRPN-011
Title	toArray returns empty array regardless of notifiable
Objective	Ensure different notifiable objects do not affect toArray() output.
Preconditions	- Application running
Test Steps	1. Instantiate notification with token 'test-token'. 2. Prepare array of notifiable objects: - stdClass - object with id and email - object with company slug 3. Call toArray() for each notifiable object. 4. Validate results.
Test Data	- token: 'test-token' - notifiables as listed above

Expected Result	- Result is always an empty array.
Actual Result	All test cases returned empty array.
Status	Pass
Severity	Low

Test Case ID	CMRPN-012
Title	toMail returns complete MailMessage with all required properties
Objective	Verify MailMessage includes all essential content.
Preconditions	- Application running
Test Steps	1. Instantiate notification with token 'complete-test-token' and slug 'complete-company'. 2. Create notifiable with company slug. 3. Call toMail(). 4. Assert non-null, non-empty: - subject - actionText - actionUrl - introLines - outroLines
Test Data	- token: 'complete-test-token' - company slug: 'complete-company'
Expected Result	- All MailMessage properties are set and not empty.
Actual Result	All MailMessage properties were populated.
Status	Pass
Severity	High

Test Case ID	CMRPN-013
Title	Notification extends ResetPassword parent class
Objective	Confirm correct class inheritance.
Preconditions	- Application running
Test Steps	1. Instantiate notification with token 'test-token'. 2. Check notification is instance of ResetPassword and Notification classes.
Test Data	- token: 'test-token'
Expected Result	- Instance inherits from both ResetPassword and Notification.
Actual Result	Inheritance confirmed for both parent classes.
Status	Pass
Severity	Medium

Test Case ID	CMRPN-014
Title	via method returns mail channel for different notifiable types
Objective	Evaluate via() method response to diverse notifiable objects.
Preconditions	- Application running
Test Steps	1. Instantiate notification with token 'test-token'. 2. For each notifiable: - stdClass - object with email - object with company slug 3. Call via(). 4. Confirm result is ['mail'].

Test Data	- token: 'test-token' - Notifiables as above
Expected Result	- via() returns array containing 'mail' for all notifiable types.
Actual Result	All cases returned ['mail'] as expected.
Status	Pass
Severity	Medium

Test Case ID	CMRPN-015
Title	toMail includes expiry time in outro lines
Objective	Validate expiry information is present in email outro.
Preconditions	- Application running
Test Steps	1. Instantiate notification with token 'expiry-test-token' and slug 'expiry-company'. 2. Create notifiable with company slug. 3. Call toMail(). 4. Check first outro line for 'expire', 'minutes', and a digit (duration).
Test Data	- token: 'expiry-test-token' - company slug: 'expiry-company'
Expected Result	- First outro line contains expiry duration (a number) and 'minutes'.
Actual Result	Expiry time present in the outro as expected.
Status	Pass
Severity	High

Test Case ID	CMRPN-016
Title	toMail generates URL with correct structure
Objective	Ensure that the reset URL matches expected application routing conventions.
Preconditions	- Application running
Test Steps	1. Instantiate notification with token 'url-structure-token' and slug 'url-company'. 2. Create notifiable with slug. 3. Call toMail(). 4. Check that actionUrl matches pattern '/customer/reset/password/' and ends with the token.
Test Data	- token: 'url-structure-token' - company slug: 'url-company'
Expected Result	- actionUrl matches '/customer/reset/password/' pattern and ends with the token.
Actual Result	URL structure adhered to routing conventions as required.
Status	Pass
Severity	High

Test Case ID	CMRPN-017
Title	toMail contains user-friendly intro message
Objective	Verify that the intro message in the email is understandable and user-oriented.
Preconditions	- Application running
Test Steps	1. Instantiate notification with token 'test-token'. 2. Create notifiable with company slug 'test'. 3. Call toMail(). 4. Inspect first intro line for 'Hello', 'password reset request', and 'account'.

Test Data	- token: 'test-token' - company slug: 'test'
Expected Result	- First intro line contains 'Hello', 'password reset request', and 'account'.
Actual Result	Intro message was user-friendly and contained required phrases.
Status	Pass
Severity	Medium

File: CustomerPolicy-Test.txt

Test Case ID	CP-001
Title	Verify viewAny method exists and returns boolean
Objective	Ensure the CustomerPolicy's viewAny method exists and returns a boolean value when provided with a User instance.
Preconditions	- Application running - Database seeded - User model available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new CustomerPolicy object. 3. Call the viewAny method on CustomerPolicy, passing the User instance. 4. Assert that the result is of boolean type.
Test Data	- User: New instance with no custom data. - Policy: New CustomerPolicy instance.
Expected Result	- The viewAny method exists and returns a boolean value (true or false).
Actual Result	- viewAny method returned a boolean value as expected.
Status	Pass
Severity	High

Test Case ID	CP-002
Title	Verify view method exists and returns boolean
Objective	Ensure CustomerPolicy's view method exists and returns a boolean when provided with User and Customer instances.
Preconditions	- Application running - Database seeded - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the view method on CustomerPolicy, passing User and Customer instances. 5. Assert that the result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- The view method exists and returns a boolean value (true or false).
Actual Result	- view method returned a boolean value as expected.
Status	Pass
Severity	High

Test Case ID	CP-003
Title	Verify create method exists and returns boolean
Objective	Ensure CustomerPolicy's create method exists and returns a boolean value when provided with a User instance.
Preconditions	- Application running - Database seeded - User model available - CustomerPolicy class available

Test Steps	1. Instantiate a new User object. 2. Instantiate a new CustomerPolicy object. 3. Call the create method on CustomerPolicy, passing the User instance. 4. Assert that the result is of boolean type.
Test Data	- User: New instance.
Expected Result	- The create method exists and returns a boolean value (true or false).
Actual Result	- create method returned a boolean value as expected.
Status	Pass
Severity	High

Test Case ID	CP-004
Title	Verify update method exists and returns boolean
Objective	Ensure CustomerPolicy's update method exists and returns a boolean when provided with User and Customer instances.
Preconditions	- Application running - Database seeded - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the update method on CustomerPolicy, passing User and Customer instances. 5. Assert that the result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- The update method exists and returns a boolean value (true or false).
Actual Result	- update method returned a boolean value as expected.
Status	Pass
Severity	High

Test Case ID	CP-005
Title	Verify delete method exists and returns boolean
Objective	Ensure CustomerPolicy's delete method exists and returns a boolean when provided with User and Customer instances.
Preconditions	- Application running - Database seeded - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the delete method on CustomerPolicy, passing User and Customer instances. 5. Assert that the result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- The delete method exists and returns a boolean value (true or false).
Actual Result	- delete method returned a boolean value as expected.
Status	Pass
Severity	High

Test Case ID	CP-006
Title	Verify restore method exists and returns boolean
Objective	Ensure CustomerPolicy's restore method exists and returns a boolean when provided with User and Customer instances.
Preconditions	- Application running - Database seeded - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the restore method on CustomerPolicy, passing User and Customer instances. 5. Assert that the result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- The restore method exists and returns a boolean value (true or false).
Actual Result	- restore method returned a boolean value as expected.
Status	Pass
Severity	High

Test Case ID	CP-007
Title	Verify forceDelete method exists and returns boolean
Objective	Ensure CustomerPolicy's forceDelete method exists and returns a boolean when provided with User and Customer instances.
Preconditions	- Application running - Database seeded - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the forceDelete method on CustomerPolicy, passing User and Customer instances. 5. Assert that the result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- The forceDelete method exists and returns a boolean value (true or false).
Actual Result	- forceDelete method returned a boolean value as expected.
Status	Pass
Severity	High

Test Case ID	CP-008
Title	Verify deleteMultiple method exists and returns boolean
Objective	Ensure CustomerPolicy's deleteMultiple method exists and returns a boolean when provided with a User instance.
Preconditions	- Application running - Database seeded - User model available - CustomerPolicy class available

Test Steps	1. Instantiate a new User object. 2. Instantiate a new CustomerPolicy object. 3. Call the deleteMultiple method on CustomerPolicy, passing the User instance. 4. Assert that the result is of boolean type.
Test Data	- User: New instance.
Expected Result	- The deleteMultiple method exists and returns a boolean value (true or false).
Actual Result	- deleteMultiple method returned a boolean value as expected.
Status	Pass
Severity	High

Test Case ID	CP-009
Title	Verify policy uses HandlesAuthorization trait
Objective	Ensure CustomerPolicy uses the HandlesAuthorization trait from Laravel for authorization handling.
Preconditions	- Application running - CustomerPolicy class available - \Illuminate\Auth\Access\HandlesAuthorization trait available
Test Steps	1. Instantiate a new CustomerPolicy object. 2. Retrieve the list of traits used by CustomerPolicy. 3. Assert that HandlesAuthorization trait is present in the list.
Test Data	- Policy: New CustomerPolicy instance.
Expected Result	- The CustomerPolicy class uses \Illuminate\Auth\Access\HandlesAuthorization trait.
Actual Result	- HandlesAuthorization trait was present in CustomerPolicy as expected.
Status	Pass
Severity	High

Test Case ID	CP-010
Title	Verify viewAny accepts User parameter correctly
Objective	Ensure that passing a User instance to viewAny does not cause any errors and returns a boolean value.
Preconditions	- Application running - User model available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new CustomerPolicy object. 3. Call the viewAny method with the User instance. 4. Assert that no error is thrown. 5. Assert result is of boolean type.
Test Data	- User: New instance.
Expected Result	- No error occurs passing User to viewAny method. - Method returns boolean value.
Actual Result	- Method accepted User and returned boolean as expected.
Status	Pass
Severity	High

Test Case ID	CP-011
Title	Verify view accepts User and Customer parameters correctly
Objective	Ensure that passing User and Customer instances to view does not cause errors and returns a boolean value.

Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the view method with User and Customer instances. 5. Assert that no error is thrown. 6. Assert result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- No error occurs passing User and Customer to view method. - Method returns boolean value.
Actual Result	- Method accepted parameters and returned boolean as expected.
Status	Pass
Severity	High

Test Case ID	CP-012
Title	Verify create accepts User parameter correctly
Objective	Ensure that passing a User instance to create does not cause any errors and returns a boolean value.
Preconditions	- Application running - User model available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new CustomerPolicy object. 3. Call the create method with the User instance. 4. Assert that no error is thrown. 5. Assert result is of boolean type.
Test Data	- User: New instance.
Expected Result	- No error occurs passing User to create method. - Method returns boolean value.
Actual Result	- Method accepted User and returned boolean as expected.
Status	Pass
Severity	High

Test Case ID	CP-013
Title	Verify update accepts User and Customer parameters correctly
Objective	Ensure that passing User and Customer instances to update does not cause errors and returns a boolean value.
Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the update method with User and Customer instances. 5. Assert that no error is thrown. 6. Assert result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- No error occurs passing User and Customer to update method. - Method returns boolean value.

Actual Result	- Method accepted parameters and returned boolean as expected.
Status	Pass
Severity	High

Test Case ID	CP-014
Title	Verify delete accepts User and Customer parameters correctly
Objective	Ensure that passing User and Customer instances to delete does not cause errors and returns a boolean value.
Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the delete method with User and Customer instances. 5. Assert that no error is thrown. 6. Assert result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- No error occurs passing User and Customer to delete method. - Method returns boolean value.
Actual Result	- Method accepted parameters and returned boolean as expected.
Status	Pass
Severity	High

Test Case ID	CP-015
Title	Verify restore accepts User and Customer parameters correctly
Objective	Ensure that passing User and Customer instances to restore does not cause errors and returns a boolean value.
Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the restore method with User and Customer instances. 5. Assert that no error is thrown. 6. Assert result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- No error occurs passing User and Customer to restore method. - Method returns boolean value.
Actual Result	- Method accepted parameters and returned boolean as expected.
Status	Pass
Severity	High

Test Case ID	CP-016
Title	Verify forceDelete accepts User and Customer parameters correctly
Objective	Ensure that passing User and Customer instances to forceDelete does not cause errors and returns a boolean value.

Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new Customer object. 3. Instantiate a new CustomerPolicy object. 4. Call the forceDelete method with User and Customer instances. 5. Assert that no error is thrown. 6. Assert result is of boolean type.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- No error occurs passing User and Customer to forceDelete method. - Method returns boolean value.
Actual Result	- Method accepted parameters and returned boolean as expected.
Status	Pass
Severity	High

Test Case ID	CP-017
Title	Verify deleteMultiple accepts User parameter correctly
Objective	Ensure that passing a User instance to deleteMultiple does not cause any errors and returns a boolean value.
Preconditions	- Application running - User model available - CustomerPolicy class available
Test Steps	1. Instantiate a new User object. 2. Instantiate a new CustomerPolicy object. 3. Call the deleteMultiple method with the User instance. 4. Assert that no error is thrown. 5. Assert result is of boolean type.
Test Data	- User: New instance.
Expected Result	- No error occurs passing User to deleteMultiple method. - Method returns boolean value.
Actual Result	- Method accepted User and returned boolean as expected.
Status	Pass
Severity	High

Test Case ID	CP-018
Title	Verify all policy methods return strictly boolean values
Objective	Ensure all permission methods in CustomerPolicy return strictly boolean values for given input.
Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a User object. 2. Instantiate a Customer object. 3. Instantiate a CustomerPolicy object. 4. Call all policy methods: viewAny, view, create, update, delete, restore, forceDelete, deleteMultiple. 5. Assert each returns a boolean value.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- Each CustomerPolicy method returns a boolean value (true or false).
Actual Result	- Each method returned boolean value as expected.

Status	Pass
Severity	High

Test Case ID	CP-019
Title	Verify policy methods can be called multiple times without errors
Objective	Confirm CustomerPolicy methods are safe to call repeatedly and always return correct types.
Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a User object. 2. Instantiate a Customer object. 3. Instantiate a CustomerPolicy object. 4. For 3 iterations, call all CustomerPolicy methods (viewAny, view, create, update, delete, restore, forceDelete, deleteMultiple). 5. Assert no error or exception is thrown during repeated calls.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- Methods can be called repeatedly; no errors or exceptions occur.
Actual Result	- No errors occurred on repeated calls; functionality preserved.
Status	Pass
Severity	High

Test Case ID	CP-020
Title	Verify CustomerPolicy can be instantiated without errors
Objective	Ensure that a CustomerPolicy instance can be created without exceptions or errors.
Preconditions	- Application running - CustomerPolicy class available
Test Steps	1. Instantiate a new CustomerPolicy object. 2. Assert that the instantiated object is of type CustomerPolicy.
Test Data	- Policy: New instance.
Expected Result	- CustomerPolicy object is created without exception and is of class CustomerPolicy.
Actual Result	- CustomerPolicy instantiated and class verified as expected.
Status	Pass
Severity	High

Test Case ID	CP-021
Title	Verify policy methods handle different User instances
Objective	Confirm that policy methods handle multiple User instances and return correct types.
Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate two User objects (User1, User2). 2. Instantiate a Customer object. 3. Instantiate a CustomerPolicy object. 4. Call the view method with User1 and Customer, and with User2 and Customer. 5. Assert both results are boolean values.

Test Data	- User1: New instance. - User2: New instance. - Customer: New instance.
Expected Result	- view method returns boolean for both User1 and User2.
Actual Result	- view method returned boolean for both users as expected.
Status	Pass
Severity	High

Test Case ID	CP-022
Title	Verify policy methods handle different Customer instances
Objective	Confirm that policy methods handle multiple Customer instances and return correct types.
Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a User object. 2. Instantiate two Customer objects (Customer1, Customer2). 3. Instantiate a CustomerPolicy object. 4. Call the view method with User and Customer1, and with User and Customer2. 5. Assert both results are boolean values.
Test Data	- User: New instance. - Customer1: New instance. - Customer2: New instance.
Expected Result	- view method returns boolean for both Customer1 and Customer2.
Actual Result	- view method returned boolean for both customers as expected.
Status	Pass
Severity	High

Test Case ID	CP-023
Title	Verify class-level permission methods work correctly
Objective	Ensure class-level CustomerPolicy methods (viewAny, create, deleteMultiple) function and return boolean values.
Preconditions	- Application running - User model available - CustomerPolicy class available
Test Steps	1. Instantiate a User object. 2. Instantiate a CustomerPolicy object. 3. Call viewAny, create, and deleteMultiple methods with User instance. 4. Assert each returns a boolean value.
Test Data	- User: New instance.
Expected Result	- All class-level methods return boolean values.
Actual Result	- All class-level methods returned boolean values as expected.
Status	Pass
Severity	High

Test Case ID	CP-024
Title	Verify instance-level permission methods work correctly
Objective	Ensure instance-level CustomerPolicy methods (view, update, delete, restore, forceDelete) function and return boolean values.

Preconditions	- Application running - User and Customer models available - CustomerPolicy class available
Test Steps	1. Instantiate a User object. 2. Instantiate a Customer object. 3. Instantiate a CustomerPolicy object. 4. Call view, update, delete, restore, and forceDelete methods with User and Customer instances. 5. Assert each returns a boolean value.
Test Data	- User: New instance. - Customer: New instance.
Expected Result	- All instance-level methods return boolean values.
Actual Result	- All instance-level methods returned boolean values as expected.
Status	Pass
Severity	High

Test Case ID	CP-025
Title	Verify policy has all required authorization methods
Objective	Ensure CustomerPolicy implements all necessary method signatures for authorization (viewAny, view, create, update, delete, restore, forceDelete, deleteMultiple).
Preconditions	- Application running - CustomerPolicy class available
Test Steps	1. Instantiate a CustomerPolicy object. 2. Assert method_exists returns true for: viewAny, view, create, update, delete, restore, forceDelete, deleteMultiple.
Test Data	- Policy: New instance.
Expected Result	- All authorization methods exist in CustomerPolicy.
Actual Result	- All required methods found in CustomerPolicy as expected.
Status	Pass
Severity	High

Test Case ID	CP-026
Title	Verify policy methods have correct number of parameters
Objective	Ensure CustomerPolicy methods have the correct method signatures and parameter counts.
Preconditions	- Application running - CustomerPolicy class available
Test Steps	1. Use PHP ReflectionClass to inspect CustomerPolicy. 2. Assert viewAny has 1 parameter. 3. Assert view has 2 parameters. 4. Assert create has 1 parameter. 5. Assert update has 2 parameters. 6. Assert delete has 2 parameters. 7. Assert restore has 2 parameters. 8. Assert forceDelete has 2 parameters. 9. Assert deleteMultiple has 1 parameter.
Test Data	- ReflectionClass on CustomerPolicy.
Expected Result	- Parameter count for each method matches expectations.
Actual Result	- Parameter counts matched for all methods.
Status	Pass
Severity	High

Test Case ID	CP-027
Title	Verify all policy methods are public
Objective	Ensure all CustomerPolicy permission methods are public and accessible.
Preconditions	- Application running - CustomerPolicy class available
Test Steps	1. Use PHP ReflectionClass to inspect CustomerPolicy. 2. Assert viewAny is public. 3. Assert view is public. 4. Assert create is public. 5. Assert update is public. 6. Assert delete is public. 7. Assert restore is public. 8. Assert forceDelete is public. 9. Assert deleteMultiple is public.
Test Data	- ReflectionClass on CustomerPolicy.
Expected Result	- All listed methods are public.
Actual Result	- All CustomerPolicy methods found to be public.
Status	Pass
Severity	High

File: CustomerPortalMiddleware-Test.txt

Test Case ID	CPM-001
Title	Error is thrown when accessing enable_portal on null user
Objective	Verify that accessing the customer portal when no authenticated user exists throws an appropriate error and prevents access.
Preconditions	- Application running - Database seeded - No customer user is authenticated - CustomerPortalMiddleware is registered
Test Steps	1. Mock the application authentication guard to return null for the user. 2. Simulate a GET request to the '/portal' endpoint. 3. Pass the request to the CustomerPortalMiddleware. 4. Observe if an ErrorException is thrown when accessing 'enable_portal' on a null user.
Test Data	- Mocked user: null - Endpoint: GET /portal
Expected Result	- An ErrorException is thrown. - Exception message matches "Attempt to read property 'enable_portal' on null". - The next middleware is NOT called.
Actual Result	An ErrorException was thrown with message "Attempt to read property 'enable_portal' on null". The next middleware was not called.
Status	Pass
Severity	High

Test Case ID	CPM-002
Title	Logout and unauthorized response when customer portal is disabled
Objective	Verify that if the authenticated user has customer portal disabled, the user is logged out and an unauthorized response is returned.
Preconditions	- Application running - Database seeded - Authenticated customer user with enable_portal set to false - CustomerPortalMiddleware is registered
Test Steps	1. Mock the authentication guard to return a user object with 'enable_portal' set to false. 2. Mock guard expects logout to be called once. 3. Simulate a GET request to the '/portal' endpoint. 4. Pass the request to the CustomerPortalMiddleware. 5. Observe the response returned by the middleware. 6. Verify that the next middleware is NOT called.
Test Data	- Mocked user: { enable_portal: false } - Endpoint: GET /portal
Expected Result	- Authentication guard's logout method is called once. - Middleware returns a Response object. - Response status code is 401 (Unauthorized). - Response content is "Unauthorized." - The next middleware is NOT called.
Actual Result	Logout was called, a 401 Unauthorized response with the correct content was returned, and the next middleware was not called.
Status	Pass
Severity	High

Test Case ID	CPM-003

Title	Proceeds to next middleware when customer portal is enabled
Objective	Verify that if the authenticated user has customer portal enabled, the middleware passes the request to the next middleware, allowing portal access.
Preconditions	- Application running - Database seeded - Authenticated customer user with enable_portal set to true - CustomerPortalMiddleware is registered
Test Steps	1. Mock the authentication guard to return a user object with 'enable_portal' set to true. 2. Ensure authentication guard does NOT call logout. 3. Simulate a GET request to the '/portal' endpoint. 4. Pass the request to the CustomerPortalMiddleware along with a next middleware closure that asserts correct behavior. 5. Observe the response returned by the middleware. 6. Verify that the next middleware IS called and receives the correct request.
Test Data	- Mocked user: { enable_portal: true } - Endpoint: GET /portal - Expected next middleware response: Response("Authorized and proceeded", 200)
Expected Result	- Middleware returns a Response object. - Exact response from next middleware ("Authorized and proceeded", 200) is returned. - The next middleware IS called. - Authentication guard's logout method is NOT called.
Actual Result	The middleware returned the expected "Authorized and proceeded" response, the next middleware was called, and logout was not called.
Status	Pass
Severity	High

File: CustomerProfileRequest-Test.txt

Test Case ID	CPR-001
Title	Authorize method returns true
Objective	Verify that the CustomerProfileRequest authorize() method always returns true, allowing requests to proceed.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Call the authorize() method on the request. 3. Check if the result is true.
Test Data	- Method: authorize() - No input data required
Expected Result	- The authorize() method returns true.
Actual Result	- The authorize() method returns true as expected.
Status	Pass
Severity	High

Test Case ID	CPR-002
Title	Rules method returns correct validation rules structure
Objective	Verify that the rules() method returns an array containing all expected keys for profile data validation.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Call the rules() method. 3. Check that the returned value is an array. 4. Verify that all expected keys are present in the rules array.
Test Data	- Method: rules() - Expected keys: 'name', 'password', 'email', billing.*, shipping.*, 'customer_avatar'
Expected Result	- rules() returns an array. - All specified keys exist in the array.
Actual Result	- The rules() method returns an array containing all specified keys.
Status	Pass
Severity	High

Test Case ID	CPR-003
Title	Name field has nullable validation
Objective	Verify that the 'name' field includes the 'nullable' validation rule.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Call the rules() method. 3. Check that the 'name' field in the rules array contains 'nullable'.
Test Data	- Field: 'name' - Rule: 'nullable'
Expected Result	- 'name' validation rules contains 'nullable'.
Actual Result	- 'name' field contains 'nullable' as expected.
Status	Pass

Severity	High

Test Case ID	CPR-004
Title	Password field has nullable and min:8 validation
Objective	Verify that the 'password' field contains both 'nullable' and 'min:8' validation rules.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Call the rules() method. 3. Check that the 'password' field contains both 'nullable' and 'min:8'.
Test Data	- Field: 'password' - Rules: 'nullable', 'min:8'
Expected Result	- 'password' field contains both 'nullable' and 'min:8'.
Actual Result	- 'password' field contains both rules as expected.
Status	Pass
Severity	High

Test Case ID	CPR-005
Title	Email field has nullable and email validation
Objective	Ensure that the 'email' field contains 'nullable' and 'email' validation rules.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Call the rules() method. 3. Verify that the 'email' field contains both 'nullable' and 'email'.
Test Data	- Field: 'email' - Rules: 'nullable', 'email'
Expected Result	- 'email' field contains 'nullable' and 'email' rules.
Actual Result	- 'email' field contains both rules as expected.
Status	Pass
Severity	High

Test Case ID	CPR-006
Title	Customer_avatar has correct validation rules
Objective	Verify that the 'customer_avatar' field contains 'nullable', 'file', 'mimes:gif,jpg,png', and 'max:20000' validation rules.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Call the rules() method. 3. Check that 'customer_avatar' field contains all required rules.
Test Data	- Field: 'customer_avatar' - Rules: 'nullable', 'file', 'mimes:gif,jpg,png', 'max:20000'
Expected Result	- 'customer_avatar' contains all specified rules.
Actual Result	- All required rules are present for 'customer_avatar'.
Status	Pass
Severity	High

Test Case ID	CPR-007
Title	All billing fields have nullable validation
Objective	Ensure that all billing-related fields include the 'nullable' validation rule.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Call the rules() method. 3. For each billing.* field, check for the 'nullable' rule.
Test Data	- Fields: 'billing.name', 'billing.address_street_1', 'billing.address_street_2', 'billing.city', 'billing.state', 'billing.country_id', 'billing.zip', 'billing.phone', 'billing.fax' - Rule: 'nullable'
Expected Result	- Each billing field contains 'nullable'.
Actual Result	- All billing fields contain 'nullable'.
Status	Pass
Severity	High

Test Case ID	CPR-008
Title	All shipping fields have nullable validation
Objective	Verify that all shipping-related fields include the 'nullable' validation rule.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Call the rules() method. 3. For each shipping.* field, check for the 'nullable' rule.
Test Data	- Fields: 'shipping.name', 'shipping.address_street_1', 'shipping.address_street_2', 'shipping.city', 'shipping.state', 'shipping.country_id', 'shipping.zip', 'shipping.phone', 'shipping.fax' - Rule: 'nullable'
Expected Result	- Each shipping field contains 'nullable'.
Actual Result	- All shipping fields contain 'nullable'.
Status	Pass
Severity	High

Test Case ID	CPR-009
Title	Validation passes with valid complete data
Objective	Verify that validation passes with a fully completed and correct data set.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Prepare complete valid profile data for all fields. 3. Remove potential unique validation on email for simplicity. 4. Use Validator::make() to validate data against rules. 5. Verify validation passes.
Test Data	- 'name': 'John Doe' - 'email': 'john@example.com' - 'password': 'password123' - 'billing': {...valid fields...} - 'shipping': {...valid fields...}
Expected Result	- Validator passes with the provided data.
Actual Result	- Validation passes successfully.

Status	Pass
Severity	High

Test Case ID	CPR-010
Title	Validation passes with minimal data since all fields are nullable
Objective	Confirm validation passes when no data is provided, leveraging 'nullable' on all fields.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Prepare empty data array. 3. Remove potential unique validation on email for simplicity. 4. Use Validator::make() to validate data against rules. 5. Check validation passes.
Test Data	- Empty data array
Expected Result	- Validator passes successfully (no required field errors).
Actual Result	- Validation passes as expected.
Status	Pass
Severity	High

Test Case ID	CPR-011
Title	Validation fails with invalid email format
Objective	Verify that validation fails when an incorrectly formatted email is provided.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Prepare profile data with 'email':'not-an-email'. 3. Remove potential unique validation on email for simplicity. 4. Use Validator::make() to validate data against rules. 5. Check that validation fails. 6. Ensure error is present for 'email' field.
Test Data	- 'email': 'not-an-email'
Expected Result	- Validation fails. - Validator errors contain 'email'.
Actual Result	- Validation fails and error for 'email' field exists.
Status	Pass
Severity	High

Test Case ID	CPR-012
Title	Validation fails with password shorter than 8 characters
Objective	Verify that validation fails when the password provided is shorter than 8 characters.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Prepare profile data with 'password': 'short'. 3. Use Validator::make() to validate data against rules. 4. Check that validation fails. 5. Ensure error is present for 'password' field.
Test Data	- 'password': 'short' (length 5)

Expected Result	- Validation fails. - Validator errors contain 'password'.
Actual Result	- Validation fails and error for 'password' field is present.
Status	Pass
Severity	High

Test Case ID	CPR-013
Title	getShippingAddress returns correct array with shipping data and type
Objective	Verify that getShippingAddress returns an array populated with provided shipping data and the correct type.
Preconditions	- Application running - CustomerProfileRequest class loaded - Address class loaded
Test Steps	1. Prepare shipping data array with sample data. 2. Create CustomerProfileRequest with shipping data. 3. Call getShippingAddress() method. 4. Verify result is an array. 5. Confirm array has 'type', matches Address::SHIPPING_TYPE. 6. Confirm name, address_street_1, city, and country_id are present and match input data.
Test Data	- Shipping data: 'name': 'Shipping Name', 'address_street_1': '123 Shipping St', 'city': 'Shipping City', 'country_id': 1
Expected Result	- Returned array contains all provided shipping values and 'type' set to Address::SHIPPING_TYPE.
Actual Result	- Returned array contains correct values and 'type' as expected.
Status	Pass
Severity	Medium

Test Case ID	CPR-014
Title	getShippingAddress returns array with type when shipping data is empty
Objective	Confirm getShippingAddress returns array with 'type'=>Address::SHIPPING_TYPE when no shipping fields are provided.
Preconditions	- Application running - CustomerProfileRequest class loaded - Address class loaded
Test Steps	1. Create CustomerProfileRequest with empty shipping data array. 2. Call getShippingAddress(). 3. Verify that returned value is an array. 4. Confirm result has 'type' key set to Address::SHIPPING_TYPE.
Test Data	- 'shipping': []
Expected Result	- Array returned contains 'type' = Address::SHIPPING_TYPE.
Actual Result	- Array returned with 'type' set as expected.
Status	Pass
Severity	Medium

Test Case ID	CPR-015
Title	getShippingAddress returns array with type when shipping data is null
Objective	Verify getShippingAddress returns array with 'type'=>Address::SHIPPING_TYPE when shipping data is missing.

Preconditions	- Application running - CustomerProfileRequest class loaded - Address class loaded
Test Steps	1. Create CustomerProfileRequest with no 'shipping' field. 2. Call getShippingAddress(). 3. Verify that returned value is an array. 4. Confirm 'type' key is present and matches Address::SHIPPING_TYPE.
Test Data	- No 'shipping' data in request
Expected Result	- Array returned contains 'type' = Address::SHIPPING_TYPE.
Actual Result	- Array returned with 'type' set as expected.
Status	Pass
Severity	Medium

Test Case ID	CPR-016
Title	getBillingAddress returns correct array with billing data and type
Objective	Verify that getBillingAddress returns an array populated with provided billing data and correct type.
Preconditions	- Application running - CustomerProfileRequest class loaded - Address class loaded
Test Steps	1. Prepare billing data array with sample data. 2. Create CustomerProfileRequest with billing data. 3. Call getBillingAddress() method. 4. Verify result is an array. 5. Confirm array has 'type', matches Address::BILLING_TYPE. 6. Confirm name, address_street_1, city, and country_id are present and match input data.
Test Data	- Billing data: 'name': 'Billing Name', 'address_street_1': '456 Billing Ave', 'city': 'Billing City', 'country_id': 2
Expected Result	- Returned array contains all provided billing values and 'type' set to Address::BILLING_TYPE.
Actual Result	- Returned array contains correct values and 'type' as expected.
Status	Pass
Severity	Medium

Test Case ID	CPR-017
Title	getBillingAddress returns array with type when billing data is empty
Objective	Confirm getBillingAddress returns array with 'type'=>Address::BILLING_TYPE when no billing fields are provided.
Preconditions	- Application running - CustomerProfileRequest class loaded - Address class loaded
Test Steps	1. Create CustomerProfileRequest with empty billing data array. 2. Call getBillingAddress(). 3. Verify that returned value is an array. 4. Confirm result has 'type' key set to Address::BILLING_TYPE.
Test Data	- 'billing': []
Expected Result	- Array returned contains 'type' = Address::BILLING_TYPE.
Actual Result	- Array returned with 'type' as expected.
Status	Pass
Severity	Medium

Test Case ID	CPR-018
Title	getBillingAddress returns array with type when billing data is null
Objective	Verify getBillingAddress returns array with 'type'=>Address::BILLING_TYPE when billing data is missing.
Preconditions	- Application running - CustomerProfileRequest class loaded - Address class loaded
Test Steps	1. Create CustomerProfileRequest with no 'billing' field. 2. Call getBillingAddress(). 3. Verify that returned value is an array. 4. Confirm 'type' key is present and matches Address::BILLING_TYPE.
Test Data	- No 'billing' data in request
Expected Result	- Array returned contains 'type' = Address::BILLING_TYPE.
Actual Result	- Array returned with 'type' as expected.
Status	Pass
Severity	Medium

Test Case ID	CPR-019
Title	Both address methods work correctly with complete data
Objective	Confirm that getBillingAddress and getShippingAddress correctly map and return their respective data from input.
Preconditions	- Application running - CustomerProfileRequest class loaded - Address class loaded
Test Steps	1. Prepare data with full billing and shipping sections. 2. Create CustomerProfileRequest with data. 3. Call getBillingAddress() and getShippingAddress(). 4. Verify each returns correct type and name.
Test Data	- billing: { 'name': 'Bill Name', 'address_street_1': '111 Bill St', 'city': 'Bill City' } - shipping: { 'name': 'Ship Name', 'address_street_1': '222 Ship St', 'city': 'Ship City' }
Expected Result	- billing['type'] = Address::BILLING_TYPE; billing['name'] = 'Bill Name' - shipping['type'] = Address::SHIPPING_TYPE; shipping['name'] = 'Ship Name'
Actual Result	- Both methods return expected values.
Status	Pass
Severity	Medium

Test Case ID	CPR-020
Title	Address type constants are correctly used
Objective	Confirm that type constants for billing and shipping addresses are set correctly and are distinct.
Preconditions	- Application running - CustomerProfileRequest class loaded - Address class loaded
Test Steps	1. Create CustomerProfileRequest with no billing or shipping data. 2. Call getBillingAddress() and getShippingAddress(). 3. Verify billing['type'] is Address::BILLING_TYPE. 4. Verify shipping['type'] is Address::SHIPPING_TYPE. 5. Verify the two types are not equal.
Test Data	- Empty data set.

Expected Result	- billing['type'] == Address::BILLING_TYPE - shipping['type'] == Address::SHIPPING_TYPE - billing['type'] != shipping['type']
Actual Result	- Types assigned and compared as expected.
Status	Pass
Severity	Medium

Test Case ID	CPR-021
Title	Address methods return arrays not collections
Objective	Verify that getBillingAddress and getShippingAddress both return arrays, not collections or other types.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Create CustomerProfileRequest with billing and shipping data containing only 'name' fields. 2. Call getBillingAddress() and getShippingAddress(). 3. Verify both methods return arrays.
Test Data	- 'billing': {'name': 'Test'} - 'shipping': {'name': 'Test'}
Expected Result	- Both methods return arrays.
Actual Result	- getBillingAddress and getShippingAddress returned arrays.
Status	Pass
Severity	Low

Test Case ID	CPR-022
Title	Validation passes with only name provided
Objective	Confirm that validation passes when only the 'name' field is provided.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Prepare data with only 'name': 'John Doe'. 3. Remove potential unique validation on email for simplicity. 4. Validate using Validator::make(). 5. Check result.
Test Data	- data: { 'name': 'John Doe' }
Expected Result	- Validator passes.
Actual Result	- Validation passes as expected.
Status	Pass
Severity	High

Test Case ID	CPR-023
Title	Validation passes with only valid email provided
Objective	Confirm that validation passes when only a valid 'email' is provided.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Prepare data with only 'email': 'test@example.com'. 3. Remove potential unique validation on email for simplicity. 4. Validate using Validator::make(). 5. Check result.

Test Data	- data: { 'email': 'test@example.com' }
Expected Result	- Validator passes.
Actual Result	- Validation passes as expected.
Status	Pass
Severity	High

Test Case ID	CPR-024
Title	Validation passes with valid password provided
Objective	Confirm that validation passes using only the 'password' field with a valid value.
Preconditions	- Application running - CustomerProfileRequest class loaded
Test Steps	1. Instantiate a CustomerProfileRequest object. 2. Prepare data with only 'password': 'validpassword123'. 3. Validate using Validator::make(). 4. Check result.
Test Data	- data: { 'password': 'validpassword123' }
Expected Result	- Validator passes.
Actual Result	- Validation passes as expected.
Status	Pass
Severity	High

File: CustomerRequest-Test.txt

Test Case ID	CR-001
Title	Authorize method always returns true
Objective	Verify that the authorize() method of CustomerRequest always returns true, ensuring the request is allowed to proceed.
Preconditions	- Application running - CustomerRequest class available - Database seeded
Test Steps	1. Instantiate a CustomerRequest object. 2. Call the authorize() method. 3. Check the return value.
Test Data	- Instantiated CustomerRequest object.
Expected Result	- authorize() returns true.
Actual Result	authorize() returned true as expected.
Status	Pass
Severity	High

Test Case ID	CR-002
Title	Rules method returns validation rules with all required fields
Objective	Ensure that the rules() method returns an array including all necessary validation rule keys for customer creation.
Preconditions	- Application running - CustomerRequest class available - Database seeded
Test Steps	1. Create a CustomerRequest with POST method. 2. Call the rules() method. 3. Check that the result is an array and contains all required keys.
Test Data	- POST request to "/test" - No input fields
Expected Result	- rules() returns an array. - Array has keys: name, email, password, phone, company_name, contact_name, website, prefix, enable_portal, currency_id.
Actual Result	All required keys were present in the rules array.
Status	Pass
Severity	High

Test Case ID	CR-003
Title	Rules method includes all billing address fields
Objective	Verify that the rules() method returns all billing address related keys.
Preconditions	- Application running - CustomerRequest class available - Database seeded
Test Steps	1. Create a CustomerRequest with POST method. 2. Call the rules() method. 3. Check for keys related to billing address.
Test Data	- POST request to "/test" - No input fields
Expected Result	- Array keys include: billing.name, billing.address_street_1, billing.address_street_2, billing.city, billing.state, billing.country_id, billing.zip, billing.phone, billing.fax.

Actual Result	All billing address fields were included as required keys.
Status	Pass
Severity	High

Test Case ID	CR-004
Title	Rules method includes all shipping address fields
Objective	Confirm that the rules() method returns all shipping address related keys.
Preconditions	- Application running - CustomerRequest class available - Database seeded
Test Steps	1. Create a CustomerRequest with POST method. 2. Call the rules() method. 3. Check for keys related to shipping address.
Test Data	- POST request to "/test" - No input fields
Expected Result	- Array keys include: shipping.name, shipping.address_street_1, shipping.address_street_2, shipping.city, shipping.state, shipping.country_id, shipping.zip, shipping.phone, shipping.fax.
Actual Result	All shipping address fields were present as array keys.
Status	Pass
Severity	High

Test Case ID	CR-005
Title	Name field is required in validation rules
Objective	Ensure the validation rules for the "name" field include the "required" rule.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Create a CustomerRequest with POST method. 2. Call the rules() method. 3. Inspect the rules for the "name" field.
Test Data	- POST request to "/test" - No input fields
Expected Result	- rules['name'] contains 'required'.
Actual Result	'rules["name"]' included 'required' as expected.
Status	Pass
Severity	High

Test Case ID	CR-006
Title	Email field has email and nullable validation
Objective	Ensure the "email" field validation rules include 'email' and 'nullable'.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Create a CustomerRequest with POST method. 2. Call the rules() method. 3. Inspect the rules for the "email" field.
Test Data	- POST request to "/test" - No input fields
Expected Result	- rules['email'] contains 'email'. - rules['email'] contains 'nullable'.
Actual Result	Both 'email' and 'nullable' present for 'email' field.

Status	Pass
Severity	High

Test Case ID	CR-007
Title	Enable_portal field has boolean validation
Objective	Ensure the "enable_portal" field's validation rules contain 'boolean'.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Create CustomerRequest with POST method. 2. Call the rules() method. 3. Inspect the rules for "enable_portal".
Test Data	- POST request to "/test" - No input fields
Expected Result	- rules['enable_portal'] contains 'boolean'.
Actual Result	The 'enable_portal' field validation contained 'boolean'.
Status	Pass
Severity	High

Test Case ID	CR-008
Title	Validation passes with complete valid data
Objective	Verify that passing all required and additional valid fields succeeds validation.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare complete valid input data for a customer. 2. Create CustomerRequest with POST method and the provided data. 3. Obtain rules and modify 'email' rule for testing. 4. Validate data using the rules. 5. Inspect if validation passes.
Test Data	- name: John Doe - email: john@example.com - password: secret123 - phone: 123-456-7890 - company_name: Acme Inc. - contact_name: Jane Smith - website: acme.com - enable_portal: true - currency_id: 1 - prefix: CUST- - billing: (name, address_street_1, city, country_id) - shipping: (name, address_street_1, city, country_id)
Expected Result	- Validation passes successfully.
Actual Result	Validation passed with full valid data.
Status	Pass
Severity	High

Test Case ID	CR-009
Title	Validation fails when name is missing
Objective	Ensure validation fails when required "name" field is missing in input.
Preconditions	- Application running - CustomerRequest class available

Test Steps	1. Prepare input data with only "email" provided. 2. Create CustomerRequest with POST method and the data. 3. Obtain and modify rules. 4. Validate data. 5. Check if validation fails and error exists for "name".
Test Data	- email: test@example.com - Missing: name
Expected Result	- Validation fails. - Error exists for 'name' field.
Actual Result	Validation failed and error reported for 'name' field as expected.
Status	Pass
Severity	High

Test Case ID	CR-010
Title	Validation fails with invalid email format
Objective	Ensure validation fails if the "email" field does not contain a valid email address.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare input data with valid name and invalid email. 2. Create CustomerRequest with POST method and the data. 3. Obtain and modify rules. 4. Validate data. 5. Check if validation fails and error exists for "email".
Test Data	- name: John Doe - email: not-an-email
Expected Result	- Validation fails. - Error exists for 'email' field.
Actual Result	Validation failed and error was reported for 'email' field.
Status	Pass
Severity	High

Test Case ID	CR-011
Title	getShippingAddress returns correct data with type
Objective	Verify getShippingAddress() returns an array with correct keys and the shipping type.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare shipping data. 2. Create CustomerRequest and assign shipping data. 3. Call getShippingAddress(). 4. Verify output includes 'type', 'name', 'address_street_1', and 'city', and correct values.
Test Data	- shipping: name: Shipping Name, address_street_1: 123 Shipping St, city: Ship City
Expected Result	- Output is array. - Includes 'type' key equal to Address::SHIPPING_TYPE. - Includes name, address_street_1, city keys and correct values.
Actual Result	Array contained all expected keys with correct values and type.
Status	Pass
Severity	Medium

Test Case ID	CR-012
Title	getShippingAddress returns only type when shipping data is empty
Objective	Ensure getShippingAddress() outputs only the 'type' key when no shipping data is provided.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Create CustomerRequest with empty shipping data. 2. Call getShippingAddress(). 3. Verify that output contains only the 'type' key.
Test Data	- shipping: []
Expected Result	- Output includes 'type' key equal to Address::SHIPPING_TYPE, no other data.
Actual Result	Returned array contained only the 'type' key as expected.
Status	Pass
Severity	Medium

Test Case ID	CR-013
Title	getBillingAddress returns correct data with type
Objective	Verify getBillingAddress() returns array with correct billing fields and address type.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare billing data. 2. Create CustomerRequest and assign billing data. 3. Call getBillingAddress(). 4. Verify output includes 'type', 'name', 'address_street_1', 'state' and correct values.
Test Data	- billing: name: Billing Name, address_street_1: 456 Billing Ave, state: Bill State
Expected Result	- Output is array. - Includes 'type' key equal to Address::BILLING_TYPE. - Includes name, address_street_1, state keys and correct values.
Actual Result	Array contained all expected keys with correct values and type.
Status	Pass
Severity	Medium

Test Case ID	CR-014
Title	getBillingAddress returns only type when billing data is empty
Objective	Ensure getBillingAddress() outputs only the 'type' key when no billing data is provided.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Create CustomerRequest with empty billing data. 2. Call getBillingAddress(). 3. Verify that output contains only the 'type' key.
Test Data	- billing: []
Expected Result	- Output includes 'type' key equal to Address::BILLING_TYPE, no other data.
Actual Result	Returned array contained only the 'type' key as expected.
Status	Pass
Severity	Medium

Test Case ID	CR-015
Title	hasAddress returns only non-null values from array
Objective	Ensure hasAddress() filters out any null values and only includes non-null fields.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare address data with some null/empty fields. 2. Create CustomerRequest and call hasAddress() with data. 3. Verify result contains only keys with non-null values.
Test Data	- address: name: Test Name, street: 123 Main St, city: null, state: '', zip: 12345, country_id: null, phone: false, fax: 0
Expected Result	- Result array only contains name, street, zip, phone, fax (not city or country_id).
Actual Result	Only non-null fields included; null fields omitted as expected.
Status	Pass
Severity	Medium

Test Case ID	CR-016
Title	hasAddress returns empty array if all values are null
Objective	Ensure hasAddress() returns empty array when all input values are null.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare address data where all values are null. 2. Create CustomerRequest and call hasAddress(). 3. Confirm result is an empty array.
Test Data	- address: name: null, street: null, city: null
Expected Result	- Output is an empty array.
Actual Result	Returned empty array as expected.
Status	Pass
Severity	Medium

Test Case ID	CR-017
Title	hasAddress returns same array if no values are null
Objective	Ensure hasAddress() returns the exact input array if all values are non-null.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare address data with all non-null values. 2. Create CustomerRequest and call hasAddress(). 3. Verify returned array matches input.
Test Data	- address: name: Value, street: Another Value, city: Some City
Expected Result	- Output matches the input array.
Actual Result	Returned array matched input array exactly.
Status	Pass
Severity	Medium

Test Case ID	CR-018
Title	hasAddress handles empty input array

Objective	Ensure hasAddress() returns empty array if the input is empty.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare empty address data array. 2. Create CustomerRequest and call hasAddress(). 3. Confirm output is empty array.
Test Data	- address: []
Expected Result	- Output is an empty array.
Actual Result	Returned empty array as expected.
Status	Pass
Severity	Low

Test Case ID	CR-019
Title	Both address methods work correctly with complete data
Objective	Verify that getBillingAddress and getShippingAddress handle complete address data and set proper type and name fields.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare complete billing and shipping data. 2. Create CustomerRequest and assign data. 3. Call getBillingAddress() and check 'type' and 'name'. 4. Call getShippingAddress() and check 'type' and 'name'.
Test Data	- billing: name: Bill Name, address_street_1: 111 Bill St, city: Bill City - shipping: name: Ship Name, address_street_1: 222 Ship St, city: Ship City
Expected Result	- billing['type'] is Address::BILLING_TYPE, billing['name'] is Bill Name - shipping['type'] is Address::SHIPPING_TYPE, shipping['name'] is Ship Name
Actual Result	Both address arrays returned with correct type and name fields.
Status	Pass
Severity	Medium

Test Case ID	CR-020
Title	Address type constants are correctly used and different
Objective	Confirm that address type constants for billing and shipping are not equal and correctly assigned.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Create CustomerRequest with empty data. 2. Call getBillingAddress() and getShippingAddress(). 3. Confirm billing address type equals Address::BILLING_TYPE. 4. Confirm shipping address type equals Address::SHIPPING_TYPE. 5. Confirm billing and shipping types are not equal.
Test Data	- No input data
Expected Result	- billing['type'] is Address::BILLING_TYPE - shipping['type'] is Address::SHIPPING_TYPE - billing['type'] != shipping['type']
Actual Result	Types assigned correctly and were different as expected.
Status	Pass
Severity	Medium

Test Case ID	CR-021

Title	Validation passes with only required name field
Objective	Verify validation passes with just the required "name" field provided.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Prepare input with 'name' field only. 2. Create CustomerRequest with POST method and data. 3. Obtain and modify rules. 4. Validate data. 5. Confirm validation passes.
Test Data	- name: John Doe
Expected Result	- Validation passes.
Actual Result	Validation passed with only the name field provided.
Status	Pass
Severity	High

Test Case ID	CR-022
Title	CustomerRequest extends FormRequest
Objective	Confirm that CustomerRequest class extends FormRequest for Laravel's request validation.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Instantiate CustomerRequest. 2. Verify its type is an instance of FormRequest.
Test Data	- CustomerRequest object
Expected Result	- Object is instance of \Illuminate\Foundation\Http\FormRequest.
Actual Result	CustomerRequest is correctly an instance of FormRequest.
Status	Pass
Severity	High

Test Case ID	CR-023
Title	CustomerRequest has all required methods
Objective	Ensure CustomerRequest contains all necessary methods for handling customer requests.
Preconditions	- Application running - CustomerRequest class available
Test Steps	1. Instantiate CustomerRequest. 2. Check for method existence: authorize, rules, getCustomerPayload, getShippingAddress, getBillingAddress, hasAddress.
Test Data	- CustomerRequest object
Expected Result	- All listed methods exist in CustomerRequest.
Actual Result	All required methods were present.
Status	Pass
Severity	High

File: CustomersController-Test.txt

Test Case ID	CC-001
Title	Controller Instantiation
Objective	Verify that the CustomersController class can be instantiated without error.
Preconditions	- Application running - Source code compiled and accessible - All dependencies available
Test Steps	1. Attempt to instantiate the CustomersController class. 2. Check the resulting object type.
Test Data	- Class: CustomersController
Expected Result	- An instance of CustomersController is created successfully. - The instance is of type CustomersController.
Actual Result	An instance of CustomersController was created and validated as CustomersController.
Status	Pass
Severity	High

Test Case ID	CC-002
Title	Controller Inheritance from Base Controller
Objective	Verify that CustomersController extends the base Controller class.
Preconditions	- Application running - Source code compiled and accessible - All dependencies available
Test Steps	1. Instantiate an object of CustomersController. 2. Check if the object is also an instance of the base Controller class.
Test Data	- Class: CustomersController
Expected Result	- CustomersController is an instance of base Controller.
Actual Result	Object was confirmed as an instance of Controller.
Status	Pass
Severity	High

Test Case ID	CC-003
Title	Controller Has Index Method
Objective	Verify that CustomersController class contains the index method.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Instantiate an object of CustomersController. 2. Check if the method 'index' exists on the object.
Test Data	- Method: index
Expected Result	- 'index' method exists in CustomersController.
Actual Result	'index' method confirmed to exist.
Status	Pass
Severity	High

Test Case ID	CC-004
Title	Controller Has Store Method
Objective	Verify that CustomersController contains a method named 'store'.

Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Instantiate an object of CustomersController. 2. Check if the method 'store' exists on the object.
Test Data	- Method: store
Expected Result	- 'store' method exists in CustomersController.
Actual Result	'store' method confirmed to exist.
Status	Pass
Severity	High

Test Case ID	CC-005
Title	Controller Has Show Method
Objective	Verify that CustomersController contains a method named 'show'.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Instantiate an object of CustomersController. 2. Check if the method 'show' exists on the object.
Test Data	- Method: show
Expected Result	- 'show' method exists in CustomersController.
Actual Result	'show' method confirmed to exist.
Status	Pass
Severity	High

Test Case ID	CC-006
Title	Controller Has Update Method
Objective	Verify that CustomersController contains a method named 'update'.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Instantiate an object of CustomersController. 2. Check if the method 'update' exists on the object.
Test Data	- Method: update
Expected Result	- 'update' method exists in CustomersController.
Actual Result	'update' method confirmed to exist.
Status	Pass
Severity	High

Test Case ID	CC-007
Title	Controller Has Delete Method
Objective	Verify that CustomersController contains a method named 'delete'.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Instantiate an object of CustomersController. 2. Check if the method 'delete' exists on the object.
Test Data	- Method: delete
Expected Result	- 'delete' method exists in CustomersController.
Actual Result	'delete' method confirmed to exist.
Status	Pass

Severity	High

Test Case ID	CC-008
Title	Controller Has All CRUD Methods
Objective	Verify that the CustomersController contains all CRUD methods: index, store, show, update, delete.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Instantiate an object of CustomersController. 2. Check existence of 'index', 'store', 'show', 'update', and 'delete' methods.
Test Data	- Methods: index, store, show, update, delete
Expected Result	- All five methods exist in CustomersController.
Actual Result	All five methods confirmed to exist.
Status	Pass
Severity	High

Test Case ID	CC-009
Title	Index Method Signature
Objective	Verify that the index method accepts one parameter and is public.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to get the 'index' method of CustomersController. 2. Check number of parameters. 3. Check method visibility.
Test Data	- Method: index
Expected Result	- Method has 1 parameter. - Method is public.
Actual Result	Method has 1 parameter and is public.
Status	Pass
Severity	High

Test Case ID	CC-010
Title	Store Method Signature
Objective	Verify that the store method accepts one parameter and is public.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access 'store' method. 2. Check number of parameters. 3. Check visibility.
Test Data	- Method: store
Expected Result	- Method has 1 parameter. - Method is public.
Actual Result	Method has 1 parameter and is public.
Status	Pass
Severity	High

Test Case ID	CC-011
Title	Show Method Signature

Objective	Verify that the show method accepts one parameter and is public.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access 'show' method. 2. Check number of parameters. 3. Check visibility.
Test Data	- Method: show
Expected Result	- Method has 1 parameter. - Method is public.
Actual Result	Method has 1 parameter and is public.
Status	Pass
Severity	High

Test Case ID	CC-012
Title	Update Method Signature
Objective	Verify that the update method accepts two parameters and is public.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access 'update' method. 2. Check number of parameters. 3. Check visibility.
Test Data	- Method: update
Expected Result	- Method has 2 parameters. - Method is public.
Actual Result	Method has 2 parameters and is public.
Status	Pass
Severity	High

Test Case ID	CC-013
Title	Delete Method Signature
Objective	Verify that the delete method accepts one parameter and is public.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access 'delete' method. 2. Check number of parameters. 3. Check visibility.
Test Data	- Method: delete
Expected Result	- Method has 1 parameter. - Method is public.
Actual Result	Method has 1 parameter and is public.
Status	Pass
Severity	High

Test Case ID	CC-014
Title	All CRUD Methods Are Public
Objective	Verify that all CRUD methods in CustomersController are public.
Preconditions	- Application running - Source code compiled and accessible

Test Steps	1. Use PHP reflection to access 'index', 'store', 'show', 'update', 'delete' methods. 2. Check visibility for all methods.
Test Data	- Methods: index, store, show, update, delete
Expected Result	- All methods are public.
Actual Result	All methods are public.
Status	Pass
Severity	High

Test Case ID	CC-015
Title	Controller Namespace Validation
Objective	Verify that CustomersController is in the correct namespace.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to check class namespace.
Test Data	- Expected Namespace: Crater\Http\Controllers\V1\Admin\Customer
Expected Result	- CustomersController resides in 'Crater\Http\Controllers\V1\Admin\Customer'.
Actual Result	Namespace confirmed as 'Crater\Http\Controllers\V1\Admin\Customer'.
Status	Pass
Severity	Medium

Test Case ID	CC-016
Title	Controller Class Name Verification
Objective	Verify that CustomersController class name is correct.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to check class short name.
Test Data	- Expected Class Name: CustomersController
Expected Result	- Class name is 'CustomersController'.
Actual Result	Class name is 'CustomersController'.
Status	Pass
Severity	Low

Test Case ID	CC-017
Title	Controller Is Not Abstract
Objective	Verify that CustomersController is not abstract.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to check if CustomersController is abstract.
Test Data	- Class: CustomersController
Expected Result	- CustomersController is not abstract.
Actual Result	Controller is not abstract.
Status	Pass
Severity	Medium

Test Case ID	CC-018

Title	Controller Is Not an Interface
Objective	Verify that CustomersController is not an interface.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to check if CustomersController is an interface.
Test Data	- Class: CustomersController
Expected Result	- CustomersController is not an interface.
Actual Result	Controller is not an interface.
Status	Pass
Severity	Medium

Test Case ID	CC-019
Title	Controller Is Not a Trait
Objective	Verify that CustomersController is not a trait.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to check if CustomersController is a trait.
Test Data	- Class: CustomersController
Expected Result	- CustomersController is not a trait.
Actual Result	Controller is not a trait.
Status	Pass
Severity	Medium

Test Case ID	CC-020
Title	Controller Is Instantiable
Objective	Verify that CustomersController can be instantiated via reflection.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to check if CustomersController is instantiable.
Test Data	- Class: CustomersController
Expected Result	- Controller is instantiable.
Actual Result	Controller is instantiable.
Status	Pass
Severity	High

Test Case ID	CC-021
Title	Multiple Controller Instances Creation
Objective	Verify that multiple CustomersController instances can be created and are distinct.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Instantiate two CustomersController objects. 2. Check each instance type. 3. Confirm that instances are not same object.
Test Data	- Instances: controller1, controller2
Expected Result	- Both instances are CustomersController. - Instances are not the same (distinct).

Actual Result	Both instances created, distinct.
Status	Pass
Severity	High

Test Case ID	CC-022
Title	Index Method Parameter Type
Objective	Verify that index method accepts a Request parameter named 'request'.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access parameters of 'index' method. 2. Check parameter name and count.
Test Data	- Method: index - Parameter: request
Expected Result	- Method has 1 parameter named 'request'.
Actual Result	Parameter name and count confirmed.
Status	Pass
Severity	High

Test Case ID	CC-023
Title	Store Method Parameter Type
Objective	Verify store method accepts CustomerRequest parameter named 'request'.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access parameters of 'store' method. 2. Check parameter name and count.
Test Data	- Method: store - Parameter: request
Expected Result	- Method has 1 parameter named 'request'.
Actual Result	Parameter name and count confirmed.
Status	Pass
Severity	High

Test Case ID	CC-024
Title	Show Method Parameter Type
Objective	Verify show method accepts Customer parameter named 'customer'.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access parameters of 'show' method. 2. Check parameter name and count.
Test Data	- Method: show - Parameter: customer
Expected Result	- Method has 1 parameter named 'customer'.
Actual Result	Parameter name and count confirmed.
Status	Pass
Severity	High

Test Case ID	CC-025

Title	Update Method Parameter Types
Objective	Verify update method accepts CustomerRequest and Customer parameters named 'request' and 'customer'.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access parameters of 'update' method. 2. Check number and names of parameters.
Test Data	- Method: update - Parameters: request, customer
Expected Result	- Method has 2 parameters: 'request' and 'customer'.
Actual Result	Parameters confirmed.
Status	Pass
Severity	High

Test Case ID	CC-026
Title	Delete Method Parameter Type
Objective	Verify delete method accepts DeleteCustomersRequest parameter named 'request'.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Use PHP reflection to access parameters of 'delete' method. 2. Check parameter name and count.
Test Data	- Method: delete - Parameter: request
Expected Result	- Method has 1 parameter named 'request'.
Actual Result	Parameter name and count confirmed.
Status	Pass
Severity	High

Test Case ID	CC-027
Title	Methods Are Callable
Objective	Verify all CRUD methods are callable on CustomersController instance.
Preconditions	- Application running - Source code compiled and accessible
Test Steps	1. Instantiate CustomersController. 2. Use is_callable to verify all methods: index, store, show, update, delete.
Test Data	- Methods: index, store, show, update, delete
Expected Result	- All methods are callable.
Actual Result	All methods callable.
Status	Pass
Severity	High

Test Case ID	CC-028
Title	Controller Constructor Has No Required Parameters
Objective	Verify CustomersController can be constructed without required arguments.
Preconditions	- Application running - Source code compiled and accessible

Test Steps	1. Use reflection to inspect constructor parameters. 2. Ensure no required parameters, or that constructor is absent.
Test Data	- Constructor parameters
Expected Result	- No required parameters for constructor, or constructor does not exist.
Actual Result	No required parameters for constructor.
Status	Pass
Severity	High

File: DashboardController-Test.txt

Test Case ID	DC-001
Title	Instantiation of DashboardController
Objective	Verify that the DashboardController can be instantiated successfully.
Preconditions	- Application running - Autoload configuration includes DashboardController - Database seeded (if required by controller)
Test Steps	1. Attempt to instantiate the DashboardController class. 2. Check if the object is of type DashboardController.
Test Data	- Class: DashboardController
Expected Result	- The created object is an instance of DashboardController.
Actual Result	- The object is correctly instantiated as DashboardController.
Status	Pass
Severity	High

Test Case ID	DC-002
Title	DashboardController Extends Controller
Objective	Verify that DashboardController extends the base Controller class.
Preconditions	- Application running - Controller class available
Test Steps	1. Instantiate the DashboardController. 2. Confirm the object is also an instance of Controller.
Test Data	- Class: DashboardController - Base class: Controller
Expected Result	- The object is an instance of Controller.
Actual Result	- The DashboardController object extends Controller.
Status	Pass
Severity	High

Test Case ID	DC-003
Title	DashboardController is Invokable
Objective	Confirm DashboardController is invokable (callable via __invoke).
Preconditions	- Application running
Test Steps	1. Instantiate the DashboardController. 2. Verify that the object is callable.
Test Data	- Class: DashboardController
Expected Result	- The DashboardController instance evaluates as callable.
Actual Result	- The DashboardController instance is callable.
Status	Pass
Severity	Medium

Test Case ID	DC-004
Title	DashboardController Has __invoke Method
Objective	Ensure DashboardController class contains __invoke method.
Preconditions	- Application running

Test Steps	1. Instantiate DashboardController. 2. Check for existence of __invoke method.
Test Data	- Class: DashboardController
Expected Result	- __invoke method exists in DashboardController.
Actual Result	- __invoke method is present.
Status	Pass
Severity	Medium

Test Case ID	DC-005
Title	Public Visibility of __invoke Method
Objective	Verify visibility of the __invoke method in DashboardController is public.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Retrieve __invoke method. 3. Assert method is public.
Test Data	- Class: DashboardController
Expected Result	- __invoke method is set as public.
Actual Result	- __invoke method is public.
Status	Pass
Severity	High

Test Case ID	DC-006
Title	__invoke Method Accepts Request Parameter
Objective	Verify that the __invoke method of DashboardController accepts a Request parameter.
Preconditions	- Application running - Illuminate\Http\Request class available
Test Steps	1. Use ReflectionClass to get __invoke method from DashboardController. 2. Retrieve parameters of __invoke method. 3. Verify there is exactly 1 parameter named 'request'.
Test Data	- Method: __invoke - Expected parameter: request
Expected Result	- __invoke method accepts one parameter named 'request'.
Actual Result	- __invoke method has a request parameter.
Status	Pass
Severity	High

Test Case ID	DC-007
Title	DashboardController Namespace Validation
Objective	Ensure DashboardController resides in the expected namespace.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to obtain namespace of DashboardController. 2. Check that the namespace matches 'Crater\Http\Controllers\V1\Customer\General'.
Test Data	- Expected namespace: Crater\Http\Controllers\V1\Customer\General
Expected Result	- Namespace of DashboardController is correct.
Actual Result	- Namespace verified as correct.
Status	Pass

Severity	Medium

Test Case ID	DC-008
Title	DashboardController Class Name Validation
Objective	Verify DashboardController's class name is correct.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Retrieve short name of class. 3. Assert it equals 'DashboardController'.
Test Data	- Expected short name: DashboardController
Expected Result	- Short name is 'DashboardController'.
Actual Result	- Class name is correct.
Status	Pass
Severity	Medium

Test Case ID	DC-009
Title	DashboardController is Not Abstract
Objective	Ensure DashboardController is not an abstract class.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Check isAbstract property.
Test Data	- Class: DashboardController
Expected Result	- isAbstract returns false.
Actual Result	- Class is not abstract.
Status	Pass
Severity	Medium

Test Case ID	DC-010
Title	DashboardController is Not an Interface
Objective	Verify that DashboardController is not an interface.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Check isInterface property.
Test Data	- Class: DashboardController
Expected Result	- isInterface returns false.
Actual Result	- Class is not an interface.
Status	Pass
Severity	Low

Test Case ID	DC-011
Title	DashboardController is Not a Trait
Objective	Validate DashboardController is not defined as a trait.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Check isTrait property.
Test Data	- Class: DashboardController

Expected Result	- isTrait returns false.
Actual Result	- Class is not a trait.
Status	Pass
Severity	Low

Test Case ID	DC-012
Title	DashboardController is Instantiable
Objective	Confirm DashboardController class can be instantiated.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Check isInstantiable property.
Test Data	- Class: DashboardController
Expected Result	- isInstantiable returns true.
Actual Result	- Class is instantiable.
Status	Pass
Severity	High

Test Case ID	DC-013
Title	Multiple DashboardController Instances Creation
Objective	Validate that multiple DashboardController instances can be created and are unique.
Preconditions	- Application running
Test Steps	1. Instantiate DashboardController as $controller1. 2. Instantiate DashboardController as $controller2. 3. Check both are instances of DashboardController. 4. Assert $controller1 is not $controller2 (not the same object).
Test Data	- Objects: $controller1, $controller2
Expected Result	- Both objects are DashboardController and not same instance.
Actual Result	- Two separate instances created as expected.
Status	Pass
Severity	Medium

Test Case ID	DC-014
Title	DashboardController Constructor Parameters Validation
Objective	Confirm DashboardController constructor has no required parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Retrieve constructor using getConstructor(). 3. Filter parameters to check required ones. 4. Assert no required constructor parameters.
Test Data	- Constructor parameters
Expected Result	- No required constructor parameters.
Actual Result	- Constructor validated to have no required parameters.
Status	Pass
Severity	Medium

Test Case ID	DC-015

Title	__invoke Method Signature Validation
Objective	Confirm signature of the __invoke method is correct.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get DashboardController. 2. Get __invoke method. 3. Check that it is public, has one parameter, and is not static.
Test Data	- __invoke method details
Expected Result	- __invoke method is public. - __invoke method has one parameter. - __invoke method is not static.
Actual Result	- Signature of __invoke is correct.
Status	Pass
Severity	High

Test Case ID	DC-016
Title	DashboardController Class Loaded Check
Objective	Validate that DashboardController class is loaded and available.
Preconditions	- Application running
Test Steps	1. Check using class_exists for DashboardController.
Test Data	- Class: DashboardController
Expected Result	- class_exists returns true.
Actual Result	- Class is loaded as expected.
Status	Pass
Severity	High

Test Case ID	DC-017
Title	DashboardController Import Statements Validation
Objective	Confirm DashboardController includes all required class import statements.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get file location of DashboardController. 2. Read file contents. 3. Check for imports of: - Controller - Estimate - Invoice - Payment - Request - Auth
Test Data	- File contents
Expected Result	- All required import statements are present in DashboardController.
Actual Result	- All import statements found as expected.
Status	Pass
Severity	High

Test Case ID	DC-018
Title	DashboardController Public Methods Count
Objective	Verify DashboardController contains at least one public method (not inherited).
Preconditions	- Application running

Test Steps	1. Use ReflectionClass on DashboardController. 2. Obtain list of public methods. 3. Filter out inherited methods. 4. Count own public methods.
Test Data	- Public methods list
Expected Result	- At least one public method in DashboardController.
Actual Result	- Controller has at least one own public method.
Status	Pass
Severity	Medium

Test Case ID	DC-019
Title	DashboardController Is Not Final
Objective	Verify that DashboardController is not declared as final and can be extended.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Check isFinal property.
Test Data	- Class: DashboardController
Expected Result	- isFinal returns false.
Actual Result	- Class can be extended (not final).
Status	Pass
Severity	Medium

Test Case ID	DC-020
Title	DashboardController Parent Class Validation
Objective	Verify DashboardController’s parent class is Controller.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Retrieve parent class object. 3. Check parent class name.
Test Data	- Parent class: Controller
Expected Result	- Parent class is Controller.
Actual Result	- Controller is parent as expected.
Status	Pass
Severity	High

Test Case ID	DC-021
Title	DashboardController Implements No Interfaces Directly
Objective	Validate DashboardController structure does not directly implement interfaces.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Verify isInstantiable property. 3. Check for existence of __invoke method. 4. Confirm class does not directly implement any interfaces.
Test Data	- Class: DashboardController
Expected Result	- Class is instantiable and has __invoke method.
Actual Result	- Class structure validated (no direct interfaces).
Status	Pass

Severity	Medium

Test Case ID	DC-022
Title	DashboardController Can Be Used in Type Hints
Objective	Ensure DashboardController can be type-hinted in functions.
Preconditions	- Application running
Test Steps	1. Create a test function accepting DashboardController argument. 2. Pass instance to function. 3. Assert returned value matches input instance.
Test Data	- Instance of DashboardController
Expected Result	- Function accepts and returns DashboardController instance as expected.
Actual Result	- Type hint usage confirmed.
Status	Pass
Severity	Medium

Test Case ID	DC-023
Title	__invoke Method Correct Visibility
Objective	Verify that __invoke method is public, not protected or private.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get DashboardController. 2. Retrieve __invoke method. 3. Verify method is public. 4. Assert method is not protected. 5. Assert method is not private.
Test Data	- Method: __invoke
Expected Result	- __invoke method is public. - __invoke method is not protected. - __invoke method is not private.
Actual Result	- Method visibility is correct.
Status	Pass
Severity	High

Test Case ID	DC-024
Title	DashboardController Namespace Depth Validation
Objective	Confirm that DashboardController namespace is at the expected depth.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on DashboardController. 2. Split namespace into parts. 3. Assert all expected segments are present: - Crater - Http - Controllers - V1 - Customer - General
Test Data	- Namespace string
Expected Result	- All expected namespace segments are present.
Actual Result	- Namespace depth is correct.
Status	Pass
Severity	Low

Test Case ID	DC-025
Title	DashboardController Cloneable Verification
Objective	Ensure DashboardController instances can be cloned.
Preconditions	- Application running
Test Steps	1. Instantiate DashboardController. 2. Clone the instance. 3. Check clone is instance of DashboardController. 4. Assert clone is not same as original.
Test Data	- Instance and its clone
Expected Result	- Clone is a DashboardController object and a distinct instance.
Actual Result	- Clone operation successful.
Status	Pass
Severity	Low

File: DatabaseEnvironmentRequest-Test.txt

Test Case ID	DER-001
Title	Authorize method always returns true
Objective	Verify that the authorize() method of DatabaseEnvironmentRequest always returns true, ensuring user authorization by default.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\DatabaseEnvironmentRequest class available
Test Steps	1. Instantiate an object of DatabaseEnvironmentRequest. 2. Invoke the authorize() method on the instance. 3. Assert that the returned value is true.
Test Data	- Input: None - Expected Value: authorize() returns true
Expected Result	authorize() returns true
Actual Result	authorize() returns true
Status	Pass
Severity	High

Test Case ID	DER-002
Title	Rules method returns sqlite-specific rules
Objective	Verify that rules() returns the correct validation rules when database_connection is set to 'sqlite'.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\DatabaseEnvironmentRequest class available
Test Steps	1. Mock the DatabaseEnvironmentRequest class as a partial mock. 2. Set the get('database_connection') method to return 'sqlite'. 3. Call the rules() method. 4. Assert that the returned array matches the expected sqlite rules.
Test Data	- Input: database_connection = 'sqlite' - Expected Value: ['app_url' => ['required', 'url'], 'database_connection' => ['required', 'string'], 'database_name' => ['required', 'string'],]
Expected Result	rules() returns array for sqlite: - app_url: required, url - database_connection: required, string - database_name: required, string
Actual Result	rules() returns array for sqlite as expected
Status	Pass
Severity	High

Test Case ID	DER-003
Title	Rules method returns default rules for mysql connection
Objective	Verify that rules() returns the correct default validation rules when database_connection is set to 'mysql'.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\DatabaseEnvironmentRequest class available

Test Steps	1. Mock the DatabaseEnvironmentRequest class as a partial mock. 2. Set the get('database_connection') method to return 'mysql'. 3. Call the rules() method. 4. Assert that the returned array matches the expected mysql rules.
Test Data	- Input: database_connection = 'mysql' - Expected Value: ['app_url' => ['required', 'url'], 'database_connection' => ['required', 'string'], 'database_hostname' => ['required', 'string'], 'database_port' => ['required', 'numeric'], 'database_name' => ['required', 'string'], 'database_username' => ['required', 'string'],]
Expected Result	rules() returns array for mysql: - app_url: required, url - database_connection: required, string - database_hostname: required, string - database_port: required, numeric - database_name: required, string - database_username: required, string
Actual Result	rules() returns array for mysql as expected
Status	Pass
Severity	High

Test Case ID	DER-004
Title	Rules method returns default rules for pgsql connection
Objective	Verify that rules() returns the correct default validation rules when database_connection is set to 'pgsql'.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\DatabaseEnvironmentRequest class available
Test Steps	1. Mock the DatabaseEnvironmentRequest class as a partial mock. 2. Set the get('database_connection') method to return 'pgsql'. 3. Call the rules() method. 4. Assert that the returned array matches the expected pgsql rules.
Test Data	- Input: database_connection = 'pgsql' - Expected Value: ['app_url' => ['required', 'url'], 'database_connection' => ['required', 'string'], 'database_hostname' => ['required', 'string'], 'database_port' => ['required', 'numeric'], 'database_name' => ['required', 'string'], 'database_username' => ['required', 'string'],]
Expected Result	rules() returns array for pgsql: - app_url: required, url - database_connection: required, string - database_hostname: required, string - database_port: required, numeric - database_name: required, string - database_username: required, string
Actual Result	rules() returns array for pgsql as expected
Status	Pass
Severity	High

Test Case ID	DER-005
Title	Rules method returns default rules for unknown database connection

Objective	Verify that rules() returns the default validation rules when database_connection is set to an unknown string.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\DatabaseEnvironmentRequest class available
Test Steps	1. Mock the DatabaseEnvironmentRequest class as a partial mock. 2. Set the get('database_connection') method to return 'unknown_db_type'. 3. Call the rules() method. 4. Assert that the returned array matches the expected default rules.
Test Data	- Input: database_connection = 'unknown_db_type' - Expected Value: ['app_url' => ['required', 'url'], 'database_connection' => ['required', 'string'], 'database_hostname' => ['required', 'string'], 'database_port' => ['required', 'numeric'], 'database_name' => ['required', 'string'], 'database_username' => ['required', 'string'],]
Expected Result	rules() returns default array for unknown database type: - app_url: required, url - database_connection: required, string - database_hostname: required, string - database_port: required, numeric - database_name: required, string - database_username: required, string
Actual Result	rules() returns default array for unknown database type as expected
Status	Pass
Severity	High

Test Case ID	DER-006
Title	Rules method returns default rules when connection is null
Objective	Verify that rules() returns the default validation rules when database_connection is null.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\DatabaseEnvironmentRequest class available
Test Steps	1. Mock the DatabaseEnvironmentRequest class as a partial mock. 2. Set the get('database_connection') method to return null. 3. Call the rules() method. 4. Assert that the returned array matches the expected default rules.
Test Data	- Input: database_connection = null - Expected Value: ['app_url' => ['required', 'url'], 'database_connection' => ['required', 'string'], 'database_hostname' => ['required', 'string'], 'database_port' => ['required', 'numeric'], 'database_name' => ['required', 'string'], 'database_username' => ['required', 'string'],]
Expected Result	rules() returns default array for null database connection: - app_url: required, url - database_connection: required, string - database_hostname: required, string - database_port: required, numeric - database_name: required, string - database_username: required, string
Actual Result	rules() returns default array for null database connection as expected
Status	Pass

Severity	High

Test Case ID	DER-007
Title	Rules method returns default rules when connection is empty string
Objective	Verify that rules() returns the default validation rules when database_connection is an empty string.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\DatabaseEnvironmentRequest class available
Test Steps	1. Mock the DatabaseEnvironmentRequest class as a partial mock. 2. Set the get('database_connection') method to return '' (empty string). 3. Call the rules() method. 4. Assert that the returned array matches the expected default rules.
Test Data	- Input: database_connection = '' - Expected Value: ['app_url' => ['required', 'url'], 'database_connection' => ['required', 'string'], 'database_hostname' => ['required', 'string'], 'database_port' => ['required', 'numeric'], 'database_name' => ['required', 'string'], 'database_username' => ['required', 'string'],]
Expected Result	rules() returns default array for empty string database connection: - app_url: required, url - database_connection: required, string - database_hostname: required, string - database_port: required, numeric - database_name: required, string - database_username: required, string
Actual Result	rules() returns default array for empty string database connection as expected
Status	Pass
Severity	High

File: DeleteExpensesRequest-Test.txt

Test Case ID	DER-001
Title	Authorize method returns true
Objective	Verify that the authorize() method of DeleteExpensesRequest returns true, allowing request processing.
Preconditions	- Application running - DeleteExpensesRequest class available - Database seeded (not required for this test)
Test Steps	1. Instantiate the DeleteExpensesRequest class. 2. Call the authorize() method.
Test Data	- No input data required. - Expected value: true
Expected Result	authorize() method returns true.
Actual Result	authorize() method returned true.
Status	Pass
Severity	High

Test Case ID	DER-002
Title	Rules method returns correct validation rules structure
Objective	Verify that the rules() method returns an array containing both 'ids' and 'ids.*' keys.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate the DeleteExpensesRequest class. 2. Call the rules() method. 3. Check if the returned array contains the 'ids' and 'ids.*' keys.
Test Data	- No input data required. - Expected keys: 'ids', 'ids.*'
Expected Result	1. rules() returns an array. 2. rules() array contains 'ids' and 'ids.*' keys.
Actual Result	rules() returned an array with the correct keys.
Status	Pass
Severity	High

Test Case ID	DER-003
Title	ids field has required validation
Objective	Validate that the 'ids' field in rules has a 'required' validation rule.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate the DeleteExpensesRequest class. 2. Call the rules() method. 3. Verify that the 'ids' field contains 'required' rule.
Test Data	- No input data required. - Expected value: 'required' in rules['ids']
Expected Result	The 'ids' rules array contains 'required'.
Actual Result	The 'ids' field contains 'required' rule.
Status	Pass
Severity	High

Test Case ID	DER-004
Title	ids.* field has required validation
Objective	Validate that the 'ids.*' field in rules has a 'required' validation rule.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate the DeleteExpensesRequest class. 2. Call the rules() method. 3. Verify that the 'ids.*' field contains 'required' rule.
Test Data	- No input data required. - Expected value: 'required' in rules['ids.*']
Expected Result	The 'ids.*' rules array contains 'required'.
Actual Result	The 'ids.*' field contains 'required' rule.
Status	Pass
Severity	High

Test Case ID	DER-005
Title	ids.* field has exists rule for expenses table
Objective	Validate that the 'ids.*' field includes an Exists rule for 'expenses' table and 'id' column.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate the DeleteExpensesRequest class. 2. Call the rules() method. 3. Identify the Exists rule in rules['ids.*']. 4. Using reflection, check the Exists rule's table is 'expenses' and column is 'id'.
Test Data	- No input data required. - Expected: Exists rule for 'expenses', 'id'
Expected Result	1. Exists rule is present on 'ids.*'. 2. Table property is 'expenses'. 3. Column property is 'id'.
Actual Result	Exists rule for 'ids.*' validated with correct table and column.
Status	Pass
Severity	High

Test Case ID	DER-006
Title	Validation passes with valid expense IDs array
Objective	Validate that the request passes with a valid array of expense IDs.
Preconditions	- Application running - Validator available - Database seeded (IDs assumed valid)
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Prepare data: ['ids' => [1, 2, 3]]. 3. Validate data using required rules. 4. Check if validation passes.
Test Data	- Input: ['ids' => [1, 2, 3]] - Expected: validator->passes() == true
Expected Result	Validation passes for valid array of expense IDs.
Actual Result	Validation passed successfully.
Status	Pass
Severity	High

Test Case ID	DER-007
Title	Validation fails when ids field is missing
Objective	Validate that the request fails validation when 'ids' is missing.
Preconditions	- Application running - Validator available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Prepare data: [] (no 'ids' field). 3. Validate data using required rules. 4. Check if validation fails and errors contain 'ids'.
Test Data	- Input: [] - Expected: validator->fails() == true; errors has 'ids'
Expected Result	Validation fails and errors include 'ids' field.
Actual Result	Validation failed, 'ids' error present.
Status	Pass
Severity	High

Test Case ID	DER-008
Title	Validation fails when ids array is empty
Objective	Validate that the request fails when 'ids' is provided as an empty array.
Preconditions	- Application running - Validator available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Prepare data: ['ids' => []]. 3. Validate data using required rules. 4. Check if validation fails.
Test Data	- Input: ['ids' => []] - Expected: validator->fails() == true
Expected Result	Validation fails for empty 'ids' array.
Actual Result	Validation failed as expected.
Status	Pass
Severity	High

Test Case ID	DER-009
Title	Validation fails when ids is null
Objective	Verify that validation fails when the 'ids' field is null.
Preconditions	- Application running - Validator available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Prepare data: ['ids' => null]. 3. Validate data using required rules. 4. Check if validation fails.
Test Data	- Input: ['ids' => null] - Expected: validator->fails() == true
Expected Result	Validation fails for null 'ids'.
Actual Result	Validation failed as expected.
Status	Pass
Severity	High

Test Case ID	DER-010

Title	DeleteExpensesRequest extends FormRequest
Objective	Verify that DeleteExpensesRequest is a subclass of FormRequest.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Check if object is an instance of Illuminate\Foundation\Http\FormRequest.
Test Data	- No input data required. - Expected: instance of FormRequest
Expected Result	DeleteExpensesRequest is an instance of FormRequest.
Actual Result	Request is an instance of FormRequest.
Status	Pass
Severity	High

Test Case ID	DER-011
Title	DeleteExpensesRequest can be instantiated
Objective	Confirm that DeleteExpensesRequest can be instantiated successfully.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Check if object is an instance of DeleteExpensesRequest.
Test Data	- No input data required. - Expected: instance of DeleteExpensesRequest
Expected Result	Request object is an instance of DeleteExpensesRequest.
Actual Result	Request instantiated successfully.
Status	Pass
Severity	High

Test Case ID	DER-012
Title	DeleteExpensesRequest has authorize method
Objective	Verify that the DeleteExpensesRequest class defines an authorize method.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Check if the class has a method named 'authorize'.
Test Data	- No input data required. - Expected: method_exists returns true
Expected Result	authorize method exists.
Actual Result	authorize method found.
Status	Pass
Severity	High

Test Case ID	DER-013
Title	DeleteExpensesRequest has rules method
Objective	Verify that the DeleteExpensesRequest class defines a rules method.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Check if the class has a method named 'rules'.

Test Data	- No input data required. - Expected: method_exists returns true
Expected Result	rules method exists.
Actual Result	rules method found.
Status	Pass
Severity	High

Test Case ID	DER-014
Title	Rules method returns an array
Objective	Confirm that the rules() method returns an array.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Call the rules() method. 3. Confirm the returned value is an array.
Test Data	- No input data required. - Expected: type is array
Expected Result	rules() returns an array.
Actual Result	rules() returned an array.
Status	Pass
Severity	High

Test Case ID	DER-015
Title	Validation passes with single expense ID
Objective	Validate that request passes with a single expense ID in array form.
Preconditions	- Application running - Validator available - Database seeded (ID assumed valid)
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Prepare data: ['ids' => [1]]. 3. Validate data with rules. 4. Check if validation passes.
Test Data	- Input: ['ids' => [1]] - Expected: validator->passes() == true
Expected Result	Validation passes for single expense ID.
Actual Result	Validation passed successfully.
Status	Pass
Severity	High

Test Case ID	DER-016
Title	Validation passes with multiple expense IDs
Objective	Validate that request passes with multiple expense IDs in array form.
Preconditions	- Application running - Validator available - Database seeded (IDs assumed valid)
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Prepare data: ['ids' => [1, 2, 3, 4, 5]]. 3. Validate data with rules. 4. Check if validation passes.

Test Data	- Input: ['ids' => [1, 2, 3, 4, 5]] - Expected: validator->passes() == true
Expected Result	Validation passes for multiple expense IDs.
Actual Result	Validation passed successfully.
Status	Pass
Severity	High

Test Case ID	DER-017
Title	Validation handles non-array ids value
Objective	Validate that the request handles when 'ids' field is not an array.
Preconditions	- Application running - Validator available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Prepare data: ['ids' => 'not-an-array']. 3. Validate data using required rules. 4. Check if validation passes the 'required' rule.
Test Data	- Input: ['ids' => 'not-an-array'] - Expected: validator->passes() == true (for 'required' rule only)
Expected Result	Validation passes due to 'required' rule with string input.
Actual Result	Validation passed as expected with string.
Status	Pass
Severity	High

Test Case ID	DER-018
Title	DeleteExpensesRequest is in correct namespace
Objective	Confirm that DeleteExpensesRequest resides in the 'Crater\Http\Requests' namespace.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Use ReflectionClass on DeleteExpensesRequest. 2. Get namespace name. 3. Verify namespace is 'Crater\Http\Requests'.
Test Data	- No input data required. - Expected: 'Crater\Http\Requests'
Expected Result	Namespace is correct.
Actual Result	Namespace is correct.
Status	Pass
Severity	Medium

Test Case ID	DER-019
Title	DeleteExpensesRequest has correct class name
Objective	Confirm that the class name is 'DeleteExpensesRequest'.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Use ReflectionClass on DeleteExpensesRequest. 2. Get short name. 3. Verify short name is 'DeleteExpensesRequest'.
Test Data	- No input data required. - Expected: 'DeleteExpensesRequest'

Expected Result	Class name matches 'DeleteExpensesRequest'.
Actual Result	Class name is correct.
Status	Pass
Severity	Medium

Test Case ID	DER-020
Title	Authorize and rules methods are public
Objective	Verify that both 'authorize' and 'rules' methods are public in DeleteExpensesRequest.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Use ReflectionClass on DeleteExpensesRequest. 2. Check if authorize method is public. 3. Check if rules method is public.
Test Data	- No input data required. - Expected: Both methods are public
Expected Result	Both methods are public.
Actual Result	Both methods are public.
Status	Pass
Severity	High

Test Case ID	DER-021
Title	Rules method returns exactly 2 validation rules
Objective	Verify that the rules() method returns an array with exactly 2 keys.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Call rules() method. 3. Count number of keys in the array returned.
Test Data	- No input data required. - Expected: 2 keys in rules array
Expected Result	rules() returns array with 2 validation rules.
Actual Result	rules() array contains 2 rules.
Status	Pass
Severity	High

Test Case ID	DER-022
Title	ids.* field has exactly 2 validation rules
Objective	Validate that the 'ids.*' field contains exactly 2 validation rules.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Call rules() method. 3. Count number of elements in rules['ids.*'] array.
Test Data	- No input data required. - Expected: 2 elements in 'ids.*' rules
Expected Result	'ids.*' field contains exactly 2 rules.
Actual Result	'ids.*' array contains 2 rules.
Status	Pass

Severity	High

Test Case ID	DER-023
Title	ids field has exactly 1 validation rule
Objective	Validate that the 'ids' field contains exactly 1 validation rule.
Preconditions	- Application running - DeleteExpensesRequest class available
Test Steps	1. Instantiate DeleteExpensesRequest. 2. Call rules() method. 3. Count number of elements in rules['ids'] array.
Test Data	- No input data required. - Expected: 1 element in 'ids' rules
Expected Result	'ids' field contains exactly 1 rule.
Actual Result	'ids' array contains 1 rule.
Status	Pass
Severity	High

File: DeleteItemsRequest-Test.txt

Test Case ID	DIR-001
Title	Authorize method returns true
Objective	Verify that the DeleteItemsRequest authorize method correctly returns true, allowing the action to proceed.
Preconditions	- Application running - Database seeded - DeleteItemsRequest class is available
Test Steps	1. Instantiate the DeleteItemsRequest class. 2. Call the authorize() method. 3. Capture the returned value.
Test Data	- Input: DeleteItemsRequest instance - Expected value: true
Expected Result	- The authorize() method returns true.
Actual Result	- The authorize() method returned true.
Status	Pass
Severity	High

Test Case ID	DIR-002
Title	Rules method returns correct structure
Objective	Verify that the rules() method in DeleteItemsRequest returns an array with the required keys ("ids" and "ids.*").
Preconditions	- Application running - Database seeded - DeleteItemsRequest class is available
Test Steps	1. Instantiate the DeleteItemsRequest class. 2. Call the rules() method. 3. Check that the result is an array. 4. Assert that the array has the keys "ids" and "ids.*".
Test Data	- Input: DeleteItemsRequest instance - Expected keys: 'ids', 'ids.*'
Expected Result	- The rules() method returns an array with keys: 'ids' and 'ids.*'.
Actual Result	- The rules() method returned an array with keys: 'ids' and 'ids.*'.
Status	Pass
Severity	High

Test Case ID	DIR-003
Title	ids field has required validation
Objective	Validate that the rules for the "ids" field include the "required" validation rule.
Preconditions	- Application running - Database seeded - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Call the rules() method. 3. Check that 'ids' is an array. 4. Ensure 'ids' array contains 'required'.
Test Data	- Field: 'ids' - Expected rule: 'required'
Expected Result	- 'ids' field rules contain 'required'.
Actual Result	- 'ids' field rules contain 'required'.

Status	Pass
Severity	High

Test Case ID	DIR-004
Title	ids.* field has required validation
Objective	Ensure the "ids.*" field rules include the "required" validation rule.
Preconditions	- Application running - Database seeded - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Call rules() method. 3. Check that 'ids.*' is an array. 4. Ensure 'ids.*' array contains 'required'.
Test Data	- Field: 'ids.*' - Expected rule: 'required'
Expected Result	- 'ids.*' field rules contain 'required'.
Actual Result	- 'ids.*' field rules contain 'required'.
Status	Pass
Severity	High

Test Case ID	DIR-005
Title	ids.* has exists rule for items table
Objective	Validate that 'ids.*' field has an 'Exists' rule referencing the 'items' table and 'id' column.
Preconditions	- Application running - Database seeded with items - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Call rules() method. 3. Fetch 'ids.*' rules. 4. Find a rule instance of Exists. 5. Use reflection to verify table is 'items' and column is 'id'.
Test Data	- Field: 'ids.*' - Expected rule: Exists('items', 'id')
Expected Result	- 'ids.*' field rules include Exists rule that references 'items' table and 'id' column.
Actual Result	- 'ids.*' field rules include Exists rule that references 'items' and 'id'.
Status	Pass
Severity	High

Test Case ID	DIR-006
Title	ids.* has three RelationNotExist rules
Objective	Verify that 'ids.*' field has exactly three RelationNotExist rules applied.
Preconditions	- Application running - Database seeded - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Call rules() method. 3. Collect all RelationNotExist instances from 'ids.*' rules. 4. Count instances.

Test Data	- Field: 'ids.*' - Expected: 3 RelationNotExist rules
Expected Result	- 'ids.*' field rules include three RelationNotExist rules.
Actual Result	- 'ids.*' field rules include three RelationNotExist rules.
Status	Pass
Severity	High

Test Case ID	DIR-007
Title	Validation passes with valid item IDs
Objective	Ensure validation passes when 'ids' contains valid integer item IDs.
Preconditions	- Application running - Database seeded with items: 1, 2, 3 - Validator and rules defined
Test Steps	1. Prepare data: ['ids' => [1, 2, 3]]. 2. Define rules: 'ids' required, 'ids.*' required and integer. 3. Create validator. 4. Run validation.
Test Data	- Input: ['ids' => [1, 2, 3]] - Expected: passes() returns true
Expected Result	- Validation passes successfully.
Actual Result	- Validation passed.
Status	Pass
Severity	High

Test Case ID	DIR-008
Title	Validation fails when ids is missing
Objective	Ensure validation fails when the 'ids' field is omitted.
Preconditions	- Application running - Database seeded - Validator and rules defined
Test Steps	1. Prepare data: []. 2. Define rules: 'ids' required. 3. Create validator. 4. Run validation.
Test Data	- Input: [] - Expected: fails() returns true
Expected Result	- Validation fails.
Actual Result	- Validation failed as expected.
Status	Pass
Severity	High

Test Case ID	DIR-009
Title	Validation fails when ids array is empty
Objective	Verify validation fails when 'ids' field is present but empty.
Preconditions	- Application running - Database seeded - Validator and rules defined

Test Steps	1. Prepare data: ['ids' => []]. 2. Define rules: 'ids' required. 3. Create validator. 4. Run validation.
Test Data	- Input: ['ids' => []] - Expected: fails() returns true
Expected Result	- Validation fails.
Actual Result	- Validation failed as expected.
Status	Pass
Severity	High

Test Case ID	DIR-010
Title	DeleteItemsRequest extends FormRequest
Objective	Confirm that the DeleteItemsRequest class extends the FormRequest base class.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Check if instance is of type FormRequest.
Test Data	- Input: DeleteItemsRequest instance - Expected type: FormRequest
Expected Result	- DeleteItemsRequest instance is a FormRequest.
Actual Result	- DeleteItemsRequest is an instance of FormRequest.
Status	Pass
Severity	Medium

Test Case ID	DIR-011
Title	DeleteItemsRequest can be instantiated
Objective	Verify that the DeleteItemsRequest class can be instantiated successfully.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Check if the instance is of type DeleteItemsRequest.
Test Data	- Input: DeleteItemsRequest instance
Expected Result	- DeleteItemsRequest instance was created successfully.
Actual Result	- DeleteItemsRequest instance was instantiated.
Status	Pass
Severity	Medium

Test Case ID	DIR-012
Title	DeleteItemsRequest has authorize and rules methods
Objective	Verify that the DeleteItemsRequest class has public 'authorize' and 'rules' methods.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Check if 'authorize' method exists. 3. Check if 'rules' method exists.

Test Data	- Input: DeleteItemsRequest instance - Methods: 'authorize', 'rules'
Expected Result	- 'authorize' and 'rules' methods exist in DeleteItemsRequest.
Actual Result	- Both methods exist in DeleteItemsRequest.
Status	Pass
Severity	Medium

Test Case ID	DIR-013
Title	Rules method returns array
Objective	Ensure the rules() method returns an array data type.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Call rules() method. 3. Check if the returned value is an array.
Test Data	- Input: DeleteItemsRequest instance
Expected Result	- rules() method returns an array.
Actual Result	- rules() returned an array.
Status	Pass
Severity	Medium

Test Case ID	DIR-014
Title	ids.* has exactly 5 validation rules
Objective	Confirm that the 'ids.*' field contains precisely five validation rules.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Call rules() method. 3. Count number of rules under 'ids.*'.
Test Data	- Field: 'ids.*' - Expected count: 5
Expected Result	- 'ids.*' field has exactly five validation rules.
Actual Result	- 'ids.*' field had five validation rules.
Status	Pass
Severity	High

Test Case ID	DIR-015
Title	DeleteItemsRequest is in correct namespace
Objective	Verify that the DeleteItemsRequest class is declared in the 'Crater\Http\Requests' namespace.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Use reflection to check the namespace of DeleteItemsRequest.
Test Data	- Expected namespace: 'Crater\Http\Requests'
Expected Result	- DeleteItemsRequest is in 'Crater\Http\Requests' namespace.
Actual Result	- DeleteItemsRequest is in 'Crater\Http\Requests' namespace.
Status	Pass
Severity	Low

Test Case ID	DIR-016
Title	DeleteItemsRequest has correct class name
Objective	Confirm the short name of the class is 'DeleteItemsRequest'.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Use reflection to get the short name of the class.
Test Data	- Expected class name: 'DeleteItemsRequest'
Expected Result	- Class name is 'DeleteItemsRequest'.
Actual Result	- Class name is 'DeleteItemsRequest'.
Status	Pass
Severity	Low

Test Case ID	DIR-017
Title	Authorize and rules methods are public
Objective	Ensure that 'authorize' and 'rules' methods of DeleteItemsRequest are public.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Use reflection to check the visibility of 'authorize' and 'rules' methods.
Test Data	- Methods: 'authorize', 'rules' - Expected: Both public
Expected Result	- Methods 'authorize' and 'rules' are public.
Actual Result	- Both methods are public.
Status	Pass
Severity	Medium

Test Case ID	DIR-018
Title	Validation passes with single item ID
Objective	Ensure validation passes when only a single valid item ID is provided.
Preconditions	- Application running - Database contains item with ID 1 - Validator and rules defined
Test Steps	1. Prepare data: ['ids' => [1]]. 2. Define rules: 'ids' required, 'ids.*' required and integer. 3. Run validator. 4. Check if passes() is true.
Test Data	- Input: ['ids' => [1]] - Expected: passes() returns true
Expected Result	- Validation passes.
Actual Result	- Validation passed.
Status	Pass
Severity	High

Test Case ID	DIR-019
Title	Validation passes with multiple item IDs
Objective	Ensure validation passes when multiple valid item IDs are provided.

Preconditions	- Application running - Database seeded with items: 1, 2, 3, 4, 5 - Validator and rules defined
Test Steps	1. Prepare data: ['ids' => [1, 2, 3, 4, 5]]. 2. Define rules: 'ids' required, 'ids.*' required, integer. 3. Create validator and check validation.
Test Data	- Input: ['ids' => [1, 2, 3, 4, 5]] - Expected: passes() returns true
Expected Result	- Validation passes.
Actual Result	- Validation passed.
Status	Pass
Severity	High

Test Case ID	DIR-020
Title	ids.* contains correct rule types
Objective	Ensure that the 'ids.*' rules array contains a string, an Exists rule, and a RelationNotExist rule.
Preconditions	- Application running - DeleteItemsRequest class is available
Test Steps	1. Instantiate DeleteItemsRequest. 2. Call rules() method. 3. Check 'ids.*' rules for: a) A string rule. b) An Exists instance. c) A RelationNotExist instance.
Test Data	- Field: 'ids.*' - Expected: Contains string, Exists, and RelationNotExist
Expected Result	- 'ids.*' rules contain a string, Exists rule, and RelationNotExist rule.
Actual Result	- 'ids.*' rules contain all required rule types.
Status	Pass
Severity	High

File: DiskEnvironmentRequest-Test.txt

Test Case ID	DER-001
Title	Authorize method always returns true
Objective	Verify that the authorize() method of DiskEnvironmentRequest always returns true, regardless of context.
Preconditions	- Application running - DiskEnvironmentRequest class is available - No authentication or authorization configuration interfering
Test Steps	1. Instantiate a new DiskEnvironmentRequest object. 2. Call the authorize() method on the object. 3. Assert that the returned value is true.
Test Data	- DiskEnvironmentRequest instance (no input parameters) - Expected value: authorize() returns true
Expected Result	authorize() method returns true.
Actual Result	authorize() method returns true.
Status	Pass
Severity	Medium

Test Case ID	DER-002
Title	Rules method returns default rules when driver is not provided
Objective	Verify that rules() returns only the default rules for 'name' and 'driver' fields when no driver is provided.
Preconditions	- Application running - DiskEnvironmentRequest class is available - Database seeded - Mockery available for mocking
Test Steps	1. Create a partial mock of DiskEnvironmentRequest. 2. Configure mock to return null for get('driver'). 3. Call the rules() method. 4. Assert that rules contain only 'name' and 'driver' keys. 5. Assert validation rules for 'name' and 'driver'. 6. Assert no rules exist for any credentials fields.
Test Data	- get('driver') returns null - Expected rules keys: ['name', 'driver'] - Absent keys: ['credentials.key', 'credentials.secret', 'credentials.region', 'credentials.bucket', 'credentials.endpoint', 'credentials.token', 'credentials.app', 'credentials.root']
Expected Result	- rules contains 'name' and 'driver' keys only. - rules['name'] is ['required']. - rules['driver'] is ['required']. - rules does not contain any credential-related keys.
Actual Result	All assertions pass and rules are as expected.
Status	Pass
Severity	High

Test Case ID	DER-003
Title	Rules method returns default rules when driver is unknown
Objective	Ensure rules() returns only the default rules for 'name' and 'driver' when an unknown driver is provided.

Preconditions	- Application running - DiskEnvironmentRequest class is available - Database seeded - Mockery available for mocking
Test Steps	1. Create a partial mock of DiskEnvironmentRequest. 2. Set up mock to return 'unknown_driver' for get('driver'). 3. Call the rules() method. 4. Assert that rules contain 'name' and 'driver' only. 5. Assert correct validation rules for 'name' and 'driver'. 6. Confirm absence of rules for credential fields.
Test Data	- get('driver') returns 'unknown_driver' - Expected rules keys: ['name', 'driver'] - Absent keys: ['credentials.key', 'credentials.secret', 'credentials.region', 'credentials.bucket', 'credentials.endpoint', 'credentials.token', 'credentials.app', 'credentials.root']
Expected Result	- rules contains 'name' and 'driver' keys only. - rules['name'] is ['required']. - rules['driver'] is ['required']. - No credential-related keys in rules.
Actual Result	All assertions pass and rules match expected output.
Status	Pass
Severity	High

Test Case ID	DER-004
Title	Rules method returns S3-specific rules when driver is 's3'
Objective	Verify rules() returns appropriate validation rules for each S3 credential when driver is set to 's3'.
Preconditions	- Application running - DiskEnvironmentRequest class is available - Database seeded - Mockery available
Test Steps	1. Create a partial mock of DiskEnvironmentRequest. 2. Configure mock to return 's3' for get('driver'). 3. Call the rules() method. 4. Assert that rules contain proper keys for S3 credentials. 5. Assert each credential key's validation rules.
Test Data	- get('driver') returns 's3' - Expected rules keys: ['name', 'driver', 'credentials.key', 'credentials.secret', 'credentials.region', 'credentials.bucket', 'credentials.root'] - Expected rule values: ['required', 'string'] for each credential
Expected Result	- rules['name'] is ['required']. - rules['driver'] is ['required']. - rules has keys for all required S3 credential fields. - Each credential field rule is ['required', 'string'].
Actual Result	All assertions pass and S3 rules are set as expected.
Status	Pass
Severity	High

Test Case ID	DER-005
Title	Rules method returns doSpaces-specific rules when driver is 'doSpaces'
Objective	Ensure rules() returns correct validation rules for doSpaces credentials when driver is 'doSpaces'.
Preconditions	- Application running - DiskEnvironmentRequest class is available - Database seeded - Mockery available

Test Steps	1. Create a partial mock of DiskEnvironmentRequest. 2. Configure mock to return 'doSpaces' for get('driver'). 3. Call the rules() method. 4. Assert that rules contain all required doSpaces credential keys. 5. Assert each credential key contains the correct validation rules.
Test Data	- get('driver') returns 'doSpaces' - Expected keys: ['name', 'driver', 'credentials.key', 'credentials.secret', 'credentials.region', 'credentials.bucket', 'credentials.endpoint', 'credentials.root'] - Expected rule values: ['required', 'string'] for each credential
Expected Result	- rules['name'] is ['required']. - rules['driver'] is ['required']. - rules includes all relevant doSpaces credential keys. - Each credential-related rule is ['required', 'string'].
Actual Result	All assertions succeed; doSpaces credential rules are present and correct.
Status	Pass
Severity	High

Test Case ID	DER-006
Title	Rules method returns Dropbox-specific rules when driver is 'dropbox'
Objective	Confirm that rules() provides appropriate validation rules for Dropbox credentials when driver is 'dropbox'.
Preconditions	- Application running - DiskEnvironmentRequest class is available - Database seeded - Mockery available
Test Steps	1. Create a partial mock of DiskEnvironmentRequest. 2. Configure mock to return 'dropbox' for get('driver'). 3. Call the rules() method. 4. Assert that rules contain keys for all Dropbox credentials. 5. Assert each Dropbox credential key's validation rules.
Test Data	- get('driver') returns 'dropbox' - Expected keys: ['name', 'driver', 'credentials.token', 'credentials.key', 'credentials.secret', 'credentials.app', 'credentials.root'] - Expected rule values: ['required', 'string'] for each credential
Expected Result	- rules['name'] is ['required']. - rules['driver'] is ['required']. - rules includes all required Dropbox credential keys. - Each credential rule is ['required', 'string'].
Actual Result	All Dropbox credential rules appear and match expectations.
Status	Pass
Severity	High

File: DomainEnvironmentRequest-Test.txt

Test Case ID	DER-001
Title	Authorizes all users for DomainEnvironmentRequest
Objective	Verify that the DomainEnvironmentRequest request class authorizes all users to proceed.
Preconditions	- Application running - DomainEnvironmentRequest class loaded - No specific user authentication required
Test Steps	1. Instantiate the DomainEnvironmentRequest object. 2. Invoke the authorize() method on the request object.
Test Data	- Request object: DomainEnvironmentRequest
Expected Result	- The authorize() method returns true, allowing all users.
Actual Result	The authorize() method returned true; all users are authorized.
Status	Pass
Severity	High

Test Case ID	DER-002
Title	Returns correct validation rules for DomainEnvironmentRequest
Objective	Ensure that the DomainEnvironmentRequest class provides the correct validation rules for the 'app_domain' field.
Preconditions	- Application running - DomainEnvironmentRequest class loaded
Test Steps	1. Instantiate the DomainEnvironmentRequest object. 2. Call the rules() method on the request object. 3. Verify that the returned value is an array. 4. Check that the array contains the key 'app_domain'. 5. Ensure that 'app_domain' is itself an array. 6. Confirm that the 'app_domain' array contains the value 'required'. 7. Verify that the 'app_domain' array contains exactly one rule.
Test Data	- Request object: DomainEnvironmentRequest - Expected rules: - 'app_domain' => ['required']
Expected Result	- rules() method returns an array. - The returned array contains the key 'app_domain'. - The value of 'app_domain' is an array. - The 'app_domain' array contains the value 'required'. - The 'app_domain' array contains exactly one rule.
Actual Result	rules() returned an array containing key 'app_domain', which is an array with one rule: 'required'.
Status	Pass
Severity	High

File: DownloadModuleController-Test.txt

Test Case ID	DMC-001
Title	Authorizes module management and downloads module successfully
Objective	Verify that authorized users can download a module successfully.
Preconditions	- Application is running - ModuleInstaller class is available - User has permissions to manage modules - Database is seeded with relevant data
Test Steps	1. Create a request with module name "test-module" and version "1.0.0". 2. Mock the ModuleInstaller to expect a download call with "test-module" and "1.0.0" and return a success response. 3. Mock the DownloadModuleController to authorize "manage modules". 4. Invoke the controller with the request. 5. Verify the response is an instance of JsonResponse. 6. Verify the response data matches ['success' => true, 'message' => 'Module downloaded successfully'].
Test Data	- Input: module = "test-module", version = "1.0.0" - Expected Response: ['success' => true, 'message' => 'Module downloaded successfully']
Expected Result	- The response is an instance of Illuminate\Http\JsonResponse. - The response data equals the expected download response.
Actual Result	The response was a JsonResponse instance and returned ['success' => true, 'message' => 'Module downloaded successfully'].
Status	Pass
Severity	High

Test Case ID	DMC-002
Title	Handles authorization failure on module download
Objective	Verify that users without proper authorization cannot download modules and an AuthorizationException is thrown.
Preconditions	- Application is running - User lacks permission to manage modules - ModuleInstaller class is available
Test Steps	1. Create a request with module name "test-module" and version "1.0.0". 2. Mock the ModuleInstaller to prevent any download invocation. 3. Mock the DownloadModuleController to throw AuthorizationException upon authorization check. 4. Attempt to invoke the controller with the request. 5. Verify that an AuthorizationException is thrown with the message "Unauthorized."
Test Data	- Input: module = "test-module", version = "1.0.0" - Expected Exception: AuthorizationException with message "Unauthorized."
Expected Result	- The controller invocation throws AuthorizationException with the message "Unauthorized."
Actual Result	The controller threw AuthorizationException with the message "Unauthorized."
Status	Pass
Severity	High

Test Case ID	DMC-003
Title	Handles module download failure and returns error response
Objective	Verify that the system correctly returns an error response when module download fails due to internal issues.

Preconditions	- Application is running - User is authorized to manage modules - ModuleInstaller class is available - Download for "broken-module" will fail
Test Steps	1. Create a request with module name "broken-module" and version "1.0.0". 2. Mock the ModuleInstaller to expect a download call and return a failure response. 3. Mock the DownloadModuleController to authorize "manage modules". 4. Invoke the controller with the request. 5. Verify the response is an instance of JsonResponse. 6. Verify the response data equals ['success' => false, 'message' => 'Failed to download module: connection error']. 7. Verify the response status is 200.
Test Data	- Input: module = "broken-module", version = "1.0.0" - Expected Response: ['success' => false, 'message' => 'Failed to download module: connection error'] - Expected Status: 200
Expected Result	- The response is a JsonResponse instance. - The response data equals the expected error response. - The response status equals 200.
Actual Result	The response was JsonResponse with status 200 and data ['success' => false, 'message' => 'Failed to download module: connection error'].
Status	Pass
Severity	High

Test Case ID	DMC-004
Title	Handles null module or version parameters gracefully
Objective	Verify that API returns an error response when module or version parameters are null.
Preconditions	- Application is running - User is authorized to manage modules - ModuleInstaller class is available
Test Steps	1. Create a request with module name null and version null. 2. Mock the ModuleInstaller to expect a download call and return an error response. 3. Mock the DownloadModuleController to authorize "manage modules". 4. Invoke the controller with the request. 5. Verify the response is an instance of JsonResponse. 6. Verify the response data equals ['success' => false, 'message' => 'Module name or version cannot be empty.'].
Test Data	- Input: module = null, version = null - Expected Response: ['success' => false, 'message' => 'Module name or version cannot be empty.']
Expected Result	- The response is a JsonResponse instance. - The response data equals the expected error response.
Actual Result	The response was JsonResponse with data ['success' => false, 'message' => 'Module name or version cannot be empty.'].
Status	Pass
Severity	Medium

Test Case ID	DMC-005
Title	Handles empty string module or version parameters gracefully
Objective	Verify that API returns an error response when module or version parameters are empty strings.

Preconditions	- Application is running - User is authorized to manage modules - ModuleInstaller class is available
Test Steps	1. Create a request with module name "" and version "". 2. Mock the ModuleInstaller to expect a download call and return an error response. 3. Mock the DownloadModuleController to authorize "manage modules". 4. Invoke the controller with the request. 5. Verify the response is an instance of JsonResponse. 6. Verify the response data equals ['success' => false, 'message' => 'Module name or version cannot be empty.'].
Test Data	- Input: module = "", version = "" - Expected Response: ['success' => false, 'message' => 'Module name or version cannot be empty.']
Expected Result	- The response is a JsonResponse instance. - The response data equals the expected error response.
Actual Result	The response was JsonResponse with data ['success' => false, 'message' => 'Module name or version cannot be empty.'].
Status	Pass
Severity	Medium

File: DropboxServiceProvider-Test.txt

Test Case ID	DSP-001
Title	Instantiation of DropboxServiceProvider
Objective	Verify that the DropboxServiceProvider class can be instantiated successfully.
Preconditions	- Application running - DropboxServiceProvider class available and autoloaded
Test Steps	1. Instantiate the application using app(). 2. Create a new DropboxServiceProvider object with the application instance. 3. Check that the object is an instance of DropboxServiceProvider.
Test Data	- $app = app() - Instantiation: new DropboxServiceProvider($app)
Expected Result	- DropboxServiceProvider instance is created successfully and is of type DropboxServiceProvider.
Actual Result	DropboxServiceProvider instance created successfully.
Status	Pass
Severity	Medium

Test Case ID	DSP-002
Title	DropboxServiceProvider Extends ServiceProvider
Objective	Verify that DropboxServiceProvider is a subclass of ServiceProvider.
Preconditions	- Application running - DropboxServiceProvider class available
Test Steps	1. Instantiate the application using app(). 2. Create a new DropboxServiceProvider object. 3. Assert that the object is an instance of ServiceProvider.
Test Data	- $app = app() - Instance creation: new DropboxServiceProvider($app)
Expected Result	- DropboxServiceProvider instance is recognized as a subclass of ServiceProvider.
Actual Result	DropboxServiceProvider is a subclass of ServiceProvider.
Status	Pass
Severity	Medium

Test Case ID	DSP-003
Title	DropboxServiceProvider Has Register Method
Objective	Ensure DropboxServiceProvider has a register method defined.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Instantiate the application using app(). 2. Create a new DropboxServiceProvider object. 3. Check if the register method exists in the provider.
Test Data	- Class: DropboxServiceProvider
Expected Result	- DropboxServiceProvider contains a register method.
Actual Result	register method exists in DropboxServiceProvider.
Status	Pass
Severity	Medium

Test Case ID	DSP-004

Title	DropboxServiceProvider Has Boot Method
Objective	Ensure DropboxServiceProvider has a boot method defined.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Instantiate the application using app(). 2. Create a new DropboxServiceProvider object. 3. Check if the boot method exists in the provider.
Test Data	- Class: DropboxServiceProvider
Expected Result	- DropboxServiceProvider contains a boot method.
Actual Result	boot method exists in DropboxServiceProvider.
Status	Pass
Severity	Medium

Test Case ID	DSP-005
Title	Register Method Execution
Objective	Confirm that the register method can be called without errors.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Instantiate the application using app(). 2. Create a new DropboxServiceProvider object. 3. Call the register() method. 4. Assert that no errors are thrown.
Test Data	- Method call: $provider->register()
Expected Result	- register() method executes without any error.
Actual Result	register() method called successfully without errors.
Status	Pass
Severity	Medium

Test Case ID	DSP-006
Title	Register Method Is Public
Objective	Verify that the register method in DropboxServiceProvider is public.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Get the "register" method. 3. Check if the method is public.
Test Data	- Method checked: 'register'
Expected Result	- register method is public.
Actual Result	register method is public.
Status	Pass
Severity	Medium

Test Case ID	DSP-007
Title	Boot Method Is Public
Objective	Verify that the boot method in DropboxServiceProvider is public.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Get the "boot" method. 3. Check if the method is public.
Test Data	- Method checked: 'boot'

Expected Result	- boot method is public.
Actual Result	boot method is public.
Status	Pass
Severity	Medium

Test Case ID	DSP-008
Title	DropboxServiceProvider Correct Namespace
Objective	Ensure DropboxServiceProvider class is assigned to the correct namespace.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass to inspect DropboxServiceProvider. 2. Retrieve its namespace. 3. Check if it matches 'Crater\Providers'.
Test Data	- Namespace: Crater\Providers
Expected Result	- Namespace of DropboxServiceProvider is 'Crater\Providers'.
Actual Result	Namespace is correctly set to 'Crater\Providers'.
Status	Pass
Severity	Low

Test Case ID	DSP-009
Title	DropboxServiceProvider Correct Class Name
Objective	Verify DropboxServiceProvider class name is correct.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Retrieve short class name. 3. Check that it is 'DropboxServiceProvider'.
Test Data	- Class name: 'DropboxServiceProvider'
Expected Result	- Short name of class is 'DropboxServiceProvider'.
Actual Result	Short name matches 'DropboxServiceProvider'.
Status	Pass
Severity	Low

Test Case ID	DSP-010
Title	DropboxServiceProvider Is Not Abstract
Objective	Ensure DropboxServiceProvider is not defined as an abstract class.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Check if class is abstract.
Test Data	- Check: isAbstract()
Expected Result	- DropboxServiceProvider is not abstract.
Actual Result	DropboxServiceProvider is not abstract.
Status	Pass
Severity	Medium

Test Case ID	DSP-011
Title	DropboxServiceProvider Is Not Interface
Objective	Ensure DropboxServiceProvider is not defined as an interface.

Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Check if class is an interface.
Test Data	- Check: isInterface()
Expected Result	- DropboxServiceProvider is not an interface.
Actual Result	DropboxServiceProvider is not an interface.
Status	Pass
Severity	Medium

Test Case ID	DSP-012
Title	DropboxServiceProvider Is Not Trait
Objective	Ensure DropboxServiceProvider is not defined as a trait.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Check if class is a trait.
Test Data	- Check: isTrait()
Expected Result	- DropboxServiceProvider is not a trait.
Actual Result	DropboxServiceProvider is not a trait.
Status	Pass
Severity	Medium

Test Case ID	DSP-013
Title	DropboxServiceProvider Is Instantiable
Objective	Verify DropboxServiceProvider class is instantiable.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Check if class is instantiable.
Test Data	- Check: isInstantiable()
Expected Result	- DropboxServiceProvider is instantiable.
Actual Result	DropboxServiceProvider is instantiable.
Status	Pass
Severity	Medium

Test Case ID	DSP-014
Title	DropboxServiceProvider Uses Required Imports
Objective	Confirm that DropboxServiceProvider uses all necessary imported classes.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Read file contents of the class. 3. Check for the presence of necessary imports.
Test Data	- Imports to check: - use Illuminate\Support\Facades\Storage - use Illuminate\Support\ServiceProvider - use League\Flysystem\Filesystem - use Spatie\Dropbox\Client - use Spatie\FlysystemDropbox\DropboxAdapter
Expected Result	- All required imports found in the file.
Actual Result	All required imports are present.

Status	Pass
Severity	Medium

Test Case ID	DSP-015
Title	Register Method Has No Required Parameters
Objective	Validate that register method has zero required parameters.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Get the register method. 3. Check the number of parameters.
Test Data	- Method: register - Parameter count: 0
Expected Result	- register method has 0 required parameters.
Actual Result	register method requires no parameters.
Status	Pass
Severity	Medium

Test Case ID	DSP-016
Title	Boot Method Has No Required Parameters
Objective	Validate that boot method has zero required parameters.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Get the boot method. 3. Check the number of parameters.
Test Data	- Method: boot - Parameter count: 0
Expected Result	- boot method has 0 required parameters.
Actual Result	boot method requires no parameters.
Status	Pass
Severity	Medium

Test Case ID	DSP-017
Title	DropboxServiceProvider Has Expected Public Methods
Objective	Ensure DropboxServiceProvider defines exactly the expected public methods (register and boot).
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Retrieve all public methods. 3. Filter to only methods defined in DropboxServiceProvider. 4. Count the filtered methods.
Test Data	- Expected public methods: register, boot
Expected Result	- At least 2 public methods (register and boot) are defined in DropboxServiceProvider.
Actual Result	DropboxServiceProvider has register and boot as public methods.
Status	Pass
Severity	Medium

Test Case ID	DSP-018

Title	Parent Class of DropboxServiceProvider
Objective	Confirm DropboxServiceProvider parent class is ServiceProvider.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Retrieve parent class. 3. Check parent class name matches ServiceProvider.
Test Data	- Expected parent class: ServiceProvider
Expected Result	- DropboxServiceProvider parent class is ServiceProvider.
Actual Result	Parent class verified as ServiceProvider.
Status	Pass
Severity	Medium

Test Case ID	DSP-019
Title	Multiple Instances Creation
Objective	Ensure that multiple DropboxServiceProvider instances can be created and are independent.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Instantiate the application using app(). 2. Create two DropboxServiceProvider objects with the same app instance. 3. Check that both are instances of DropboxServiceProvider. 4. Check that the two objects are not the same instance.
Test Data	- $provider1 = new DropboxServiceProvider($app) - $provider2 = new DropboxServiceProvider($app)
Expected Result	- Both $provider1 and $provider2 are DropboxServiceProvider instances and represent different objects.
Actual Result	Multiple independent instances created successfully.
Status	Pass
Severity	Medium

Test Case ID	DSP-020
Title	DropboxServiceProvider Type-Hint Usage
Objective	Verify DropboxServiceProvider class can be used in type hints in PHP functions.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Define a function accepting DropboxServiceProvider as type-hinted parameter. 2. Instantiate DropboxServiceProvider object. 3. Pass the instance to the function. 4. Check that returned value matches the passed instance.
Test Data	- Function: function (DropboxServiceProvider $provider) - Passed instance: $provider = new DropboxServiceProvider($app)
Expected Result	- Type hint works; returned value matches DropboxServiceProvider instance.
Actual Result	DropboxServiceProvider accepted in type hint successfully.
Status	Pass
Severity	Medium

Test Case ID	DSP-021
Title	DropboxServiceProvider Is Not Final
Objective	Confirm that DropboxServiceProvider can be extended (is not final).

Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Check if class is final.
Test Data	- Check: isFinal()
Expected Result	- DropboxServiceProvider is not final.
Actual Result	DropboxServiceProvider not final; can be extended.
Status	Pass
Severity	Medium

Test Case ID	DSP-022
Title	Register and Boot Methods Are Not Static
Objective	Confirm that register and boot methods are not static.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Check if register method is static. 3. Check if boot method is static.
Test Data	- Methods: register, boot
Expected Result	- Neither register nor boot are static methods.
Actual Result	Both methods are confirmed non-static.
Status	Pass
Severity	Medium

Test Case ID	DSP-023
Title	Register and Boot Methods Are Not Abstract
Objective	Confirm that register and boot methods are not abstract.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Check if register is abstract. 3. Check if boot is abstract.
Test Data	- Methods: register, boot
Expected Result	- Both methods are concrete (not abstract).
Actual Result	Both methods are not abstract.
Status	Pass
Severity	Medium

Test Case ID	DSP-024
Title	DropboxServiceProvider Class is Loaded
Objective	Ensure that DropboxServiceProvider class exists and is loaded.
Preconditions	- Application running - DropboxServiceProvider class properly autoloaded
Test Steps	1. Check if DropboxServiceProvider class exists in environment.
Test Data	- Class: DropboxServiceProvider
Expected Result	- DropboxServiceProvider class exists in runtime.
Actual Result	DropboxServiceProvider class exists and is loaded.
Status	Pass
Severity	High

Test Case ID	DSP-025
Title	DropboxServiceProvider Namespace Depth
Objective	Ensure DropboxServiceProvider is within correct namespace depth and structure.
Preconditions	- DropboxServiceProvider class available
Test Steps	1. Use ReflectionClass on DropboxServiceProvider. 2. Retrieve the namespace string. 3. Split namespace by backslash. 4. Check that parts include 'Crater' and 'Providers'.
Test Data	- Namespace: Crater\Providers
Expected Result	- Namespace structure contains both 'Crater' and 'Providers'.
Actual Result	Namespace structure confirmed to contain 'Crater' and 'Providers'.
Status	Pass
Severity	Low

File: EmailLog-Test.txt

Test Case ID	EL-001
Title	Instantiate EmailLog Model
Objective	Verify that the EmailLog model can be successfully instantiated.
Preconditions	- Application running - EmailLog model class is available - Database seeded
Test Steps	1. Attempt to instantiate a new EmailLog object. 2. Verify that the created object is an instance of EmailLog.
Test Data	- Instantiation: new EmailLog()
Expected Result	- The newly created object is an instance of the EmailLog class.
Actual Result	The EmailLog model was successfully instantiated and recognized as an instance of EmailLog.
Status	Pass
Severity	Medium

Test Case ID	EL-002
Title	EmailLog Extends Eloquent Model
Objective	Confirm that EmailLog extends the base Eloquent Model class.
Preconditions	- Application running - EmailLog and Model classes are available
Test Steps	1. Instantiate an EmailLog object. 2. Verify the created object is an instance of Model.
Test Data	- Instantiation: new EmailLog()
Expected Result	- The EmailLog object is an instance of Model.
Actual Result	EmailLog was successfully recognized as extending Model.
Status	Pass
Severity	High

Test Case ID	EL-003
Title	EmailLog Has mailable Method
Objective	Ensure that EmailLog model defines a 'mailable' method.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Instantiate an EmailLog object. 2. Check if 'mailable' method exists in the EmailLog object.
Test Data	- Method checked: 'mailable'
Expected Result	- The 'mailable' method exists in EmailLog.
Actual Result	'mailable' method was found in EmailLog.
Status	Pass
Severity	Medium

Test Case ID	EL-004
Title	EmailLog Has isExpired Method
Objective	Ensure that EmailLog model defines an 'isExpired' method.

Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Instantiate an EmailLog object. 2. Check if 'isExpired' method exists in the EmailLog object.
Test Data	- Method checked: 'isExpired'
Expected Result	- The 'isExpired' method exists in EmailLog.
Actual Result	'isExpired' method was found in EmailLog.
Status	Pass
Severity	Medium

Test Case ID	EL-005
Title	mailable Method is Public
Objective	Verify that the 'mailable' method in EmailLog is public.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection on EmailLog class. 2. Retrieve the 'mailable' method. 3. Check that the method is public.
Test Data	- Method: 'mailable'
Expected Result	- 'mailable' method has public visibility.
Actual Result	'mailable' method is confirmed to be public.
Status	Pass
Severity	Medium

Test Case ID	EL-006
Title	isExpired Method is Public
Objective	Verify that the 'isExpired' method in EmailLog is public.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection on EmailLog class. 2. Retrieve the 'isExpired' method. 3. Check that the method is public.
Test Data	- Method: 'isExpired'
Expected Result	- 'isExpired' method has public visibility.
Actual Result	'isExpired' method is confirmed to be public.
Status	Pass
Severity	Medium

Test Case ID	EL-007
Title	EmailLog Namespace Verification
Objective	Validate that EmailLog is declared in the 'Crater\Models' namespace.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to obtain EmailLog class namespace. 2. Compare namespace with expected value.
Test Data	- Expected namespace: 'Crater\Models'
Expected Result	- EmailLog is in the 'Crater\Models' namespace.
Actual Result	EmailLog namespace verified as 'Crater\Models'.

Status	Pass
Severity	Low

Test Case ID	EL-008
Title	EmailLog Class Name Verification
Objective	Confirm that the EmailLog model class name is correctly set.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to get the class short name. 2. Check if the class name is 'EmailLog'.
Test Data	- Expected class name: 'EmailLog'
Expected Result	- The short name of the class is 'EmailLog'.
Actual Result	Class name returned as 'EmailLog'.
Status	Pass
Severity	Low

Test Case ID	EL-009
Title	EmailLog is Not Abstract
Objective	Ensure the EmailLog model class is not abstract and can be instantiated.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to check if the class is abstract.
Test Data	- Class: EmailLog
Expected Result	- EmailLog class is not abstract.
Actual Result	EmailLog confirmed as non-abstract.
Status	Pass
Severity	Medium

Test Case ID	EL-010
Title	EmailLog is Not an Interface
Objective	Ensure the EmailLog class is not defined as an interface.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to check if the class is an interface.
Test Data	- Class: EmailLog
Expected Result	- EmailLog is not an interface.
Actual Result	EmailLog class is not an interface.
Status	Pass
Severity	Low

Test Case ID	EL-011
Title	EmailLog is Not a Trait
Objective	Ensure that the EmailLog model is not a trait.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to check if the class is a trait.

Test Data	- Class: EmailLog
Expected Result	- EmailLog is not a trait.
Actual Result	EmailLog class confirmed as not a trait.
Status	Pass
Severity	Low

Test Case ID	EL-012
Title	EmailLog is Instantiable
Objective	Verify that the EmailLog model class can be instantiated (not abstract, not interface).
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to check if the class is instantiable.
Test Data	- Class: EmailLog
Expected Result	- EmailLog class is instantiable.
Actual Result	EmailLog class is instantiable.
Status	Pass
Severity	Medium

Test Case ID	EL-013
Title	EmailLog Uses HasFactory Trait
Objective	Verify that the EmailLog model uses the HasFactory trait for model factories.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to retrieve trait names from EmailLog. 2. Check if 'Illuminate\Database\Eloquent\Factories\HasFactory' is among the traits.
Test Data	- Expected trait: 'Illuminate\Database\Eloquent\Factories\HasFactory'
Expected Result	- EmailLog uses HasFactory trait.
Actual Result	HasFactory trait is used by EmailLog.
Status	Pass
Severity	Medium

Test Case ID	EL-014
Title	EmailLog Has Guarded Property
Objective	Ensure the EmailLog model includes the 'guarded' property for mass assignment protection.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to verify if the 'guarded' property exists in EmailLog.
Test Data	- Property: 'guarded'
Expected Result	- 'guarded' property exists in EmailLog.
Actual Result	'guarded' property found in EmailLog.
Status	Pass
Severity	High

Test Case ID	EL-015

Title	mailable Method Has No Required Parameters
Objective	Verify that the mailable() method does not require parameters for invocation.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to get the 'mailable' method parameters. 2. Confirm the method has zero parameters.
Test Data	- Method: 'mailable'
Expected Result	- mailable method has zero parameters.
Actual Result	mailable method requires no parameters.
Status	Pass
Severity	Medium

Test Case ID	EL-016
Title	isExpired Method Has No Required Parameters
Objective	Verify that the isExpired() method does not require parameters.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to get 'isExpired' method parameters. 2. Confirm the method has zero parameters.
Test Data	- Method: 'isExpired'
Expected Result	- isExpired method has zero parameters.
Actual Result	isExpired method requires no parameters.
Status	Pass
Severity	Medium

Test Case ID	EL-017
Title	EmailLog Uses Required Class Imports
Objective	Ensure the EmailLog model file contains all necessary 'use' statements for dependencies.
Preconditions	- Application running - EmailLog model file is accessible
Test Steps	1. Use reflection to load the EmailLog file contents. 2. Check for the presence of 'use Carbon\Carbon', 'use Illuminate\Database\Eloquent\Factories\HasFactory', and 'use Illuminate\Database\Eloquent\Model' imports.
Test Data	- Imports: 'Carbon\Carbon', 'HasFactory', 'Model'
Expected Result	- All required use statements are present in the file.
Actual Result	All dependencies successfully imported into EmailLog model.
Status	Pass
Severity	High

Test Case ID	EL-018
Title	Multiple EmailLog Instances Creation
Objective	Verify that multiple instances of EmailLog can be created independently.
Preconditions	- Application running - EmailLog model class is available

Test Steps	1. Instantiate two EmailLog objects. 2. Verify both are instances of EmailLog. 3. Confirm they are not the same object in memory.
Test Data	- Instantiations: emailLog1 = new EmailLog(), emailLog2 = new EmailLog()
Expected Result	- Both objects are instances of EmailLog. - They are distinct objects.
Actual Result	Both EmailLog instances created and verified as distinct.
Status	Pass
Severity	Medium

Test Case ID	EL-019
Title	EmailLog Type Hinting Capability
Objective	Ensure EmailLog model supports type-hinting in function signatures.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Create a function with an EmailLog type hint as parameter. 2. Pass an EmailLog object to the function. 3. Validate output matches input object.
Test Data	- Function: function(EmailLog $emailLog) { return $emailLog; } - Input: new EmailLog()
Expected Result	- Function accepts and returns EmailLog object without errors.
Actual Result	Function accepted and returned EmailLog object correctly.
Status	Pass
Severity	Low

Test Case ID	EL-020
Title	EmailLog is Not Final
Objective	Verify that EmailLog model class is not declared as final and can be extended.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to check if the EmailLog class is final.
Test Data	- Class: EmailLog
Expected Result	- EmailLog class is not final.
Actual Result	EmailLog class confirmed as non-final.
Status	Pass
Severity	Medium

Test Case ID	EL-021
Title	mailable and isExpired Methods are Not Static
Objective	Ensure that mailable() and isExpired() methods are instance methods (not static).
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to check if mailable and isExpired methods are static.
Test Data	- Methods: 'mailable', 'isExpired'
Expected Result	- Both methods are not static.
Actual Result	Both methods confirmed as non-static.
Status	Pass

Severity	Medium

Test Case ID	EL-022
Title	mailable and isExpired Methods are Not Abstract
Objective	Ensure mailable() and isExpired() methods have implementations (not abstract).
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to check if mailable and isExpired methods are abstract.
Test Data	- Methods: 'mailable', 'isExpired'
Expected Result	- Both methods are not abstract.
Actual Result	Both methods confirmed as non-abstract.
Status	Pass
Severity	Medium

Test Case ID	EL-023
Title	EmailLog Class File Exists and is Loaded
Objective	Confirm EmailLog model class is loaded and available through autoloading.
Preconditions	- Application running
Test Steps	1. Check if EmailLog class exists in memory.
Test Data	- Class: EmailLog
Expected Result	- EmailLog class exists and is loaded.
Actual Result	EmailLog class is loaded and available.
Status	Pass
Severity	High

Test Case ID	EL-024
Title	EmailLog Has Expected Number of Own Public Methods
Objective	Ensure EmailLog defines at least two public methods of its own.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to retrieve all public methods on EmailLog. 2. Filter for methods defined on EmailLog, not inherited. 3. Confirm there are two or more.
Test Data	- Expected: >=2 own public methods
Expected Result	- At least two EmailLog own public methods are present.
Actual Result	EmailLog contains two or more own public methods.
Status	Pass
Severity	Medium

Test Case ID	EL-025
Title	EmailLog Parent Class Verification
Objective	Confirm that the parent class of EmailLog is Model.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to get EmailLog parent class. 2. Verify parent class is Model.

Test Data	- Expected parent: Model
Expected Result	- Parent class of EmailLog is Model.
Actual Result	EmailLog parent is Model.
Status	Pass
Severity	High

Test Case ID	EL-026
Title	EmailLog Namespace Depth Verification
Objective	Ensure correct namespace depth is used for EmailLog.
Preconditions	- Application running - EmailLog model class is available
Test Steps	1. Use reflection to extract namespace of EmailLog. 2. Split namespace by backslash. 3. Confirm that 'Crater' and 'Models' are present.
Test Data	- Expected namespace parts: 'Crater', 'Models'
Expected Result	- Namespace contains both 'Crater' and 'Models'.
Actual Result	Namespace correctly includes 'Crater' and 'Models'.
Status	Pass
Severity	Low

File: EnableModuleController-Test.txt

Test Case ID	EMC-001
Title	Instantiation of EnableModuleController
Objective	Verify that EnableModuleController can be instantiated.
Preconditions	- Application running - Proper autoloading setup - Crater\Http\Controllers\V1\Admin\Modules\EnableModuleController class exists
Test Steps	1. Attempt to instantiate EnableModuleController class. 2. Check the type of the resulting object.
Test Data	- Input: N/A (class instantiation) - Expected: Object of type EnableModuleController
Expected Result	- An object of type EnableModuleController is instantiated successfully.
Actual Result	EnableModuleController is instantiated and is of the correct type.
Status	Pass
Severity	High

Test Case ID	EMC-002
Title	Inheritance from base Controller
Objective	Verify that EnableModuleController extends the base Controller class.
Preconditions	- Application running - Base Controller class available
Test Steps	1. Instantiate EnableModuleController. 2. Check if instance is also of type Controller.
Test Data	- Input: N/A - Expected: Object is instance of Controller.
Expected Result	- EnableModuleController instance is an instance of Controller.
Actual Result	EnableModuleController correctly extends Controller.
Status	Pass
Severity	High

Test Case ID	EMC-003
Title	Invokable Controller Class
Objective	Confirm EnableModuleController is invokable (i.e., implements __invoke).
Preconditions	- Application running
Test Steps	1. Instantiate EnableModuleController. 2. Check if the object is callable.
Test Data	- Input: N/A - Expected: Object is callable.
Expected Result	- EnableModuleController instance is callable.
Actual Result	Object is callable as expected.
Status	Pass
Severity	High

Test Case ID	EMC-004
Title	Existence of __invoke Method
Objective	Ensure EnableModuleController has an __invoke method.

Preconditions	- Application running
Test Steps	1. Instantiate EnableModuleController. 2. Check if __invoke method exists on the object.
Test Data	- Input: N/A - Expected: Method exists.
Expected Result	- __invoke method exists on the controller.
Actual Result	__invoke method exists.
Status	Pass
Severity	High

Test Case ID	EMC-005
Title	Public __invoke Method
Objective	Verify that __invoke method of EnableModuleController is public.
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController class. 2. Get the __invoke method. 3. Check if the method is public.
Test Data	- Input: N/A - Expected: Method is public.
Expected Result	- __invoke method is public.
Actual Result	__invoke is public as required.
Status	Pass
Severity	High

Test Case ID	EMC-006
Title	__invoke Method Accepts Request and String Parameters
Objective	Ensure __invoke method accepts a Request and a string as parameters.
Preconditions	- Application running - Illuminate\Http\Request is imported
Test Steps	1. Use reflection on EnableModuleController. 2. Get parameters for __invoke method. 3. Verify parameter count is 2. 4. Verify parameter names are "request" and "module".
Test Data	- Input: (parameters of __invoke) - Expected: Parameters are: request, module
Expected Result	- __invoke method has exactly two parameters: request and module.
Actual Result	Parameters are correct.
Status	Pass
Severity	High

Test Case ID	EMC-007
Title	Correct Namespace of EnableModuleController
Objective	Verify EnableModuleController is in 'Crater\Http\Controllers\V1\Admin\Modules'.
Preconditions	- Application running
Test Steps	1. Use reflection to get EnableModuleController's namespace. 2. Verify that it matches the expected string.
Test Data	- Input: Reflection namespace value - Expected: 'Crater\Http\Controllers\V1\Admin\Modules'

Expected Result	- Namespace is 'Crater\Http\Controllers\V1\Admin\Modules'.
Actual Result	Namespace matches as required.
Status	Pass
Severity	Medium

Test Case ID	EMC-008
Title	Correct Class Name for Controller
Objective	Ensure the short name of the controller is 'EnableModuleController'.
Preconditions	- Application running
Test Steps	1. Use reflection to get short class name. 2. Verify that it is 'EnableModuleController'.
Test Data	- Input: Reflection short name - Expected: 'EnableModuleController'
Expected Result	- Class name is 'EnableModuleController'.
Actual Result	Class name matches.
Status	Pass
Severity	Medium

Test Case ID	EMC-009
Title	Controller is Not Abstract
Objective	Validate that EnableModuleController is not abstract.
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController. 2. Verify isAbstract returns false.
Test Data	- Input: Reflection isAbstract value - Expected: false
Expected Result	- EnableModuleController is not abstract.
Actual Result	Not abstract as expected.
Status	Pass
Severity	High

Test Case ID	EMC-010
Title	Controller is Not an Interface
Objective	Confirm EnableModuleController is not an interface.
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController. 2. Verify isInterface returns false.
Test Data	- Input: Reflection isInterface value - Expected: false
Expected Result	- EnableModuleController is not an interface.
Actual Result	Not an interface.
Status	Pass
Severity	High

Test Case ID	EMC-011
Title	Controller is Not a Trait

Objective	Confirm EnableModuleController is not a trait.
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController. 2. Verify isTrait returns false.
Test Data	- Input: Reflection isTrait value - Expected: false
Expected Result	- EnableModuleController is not a trait.
Actual Result	Not a trait.
Status	Pass
Severity	High

Test Case ID	EMC-012
Title	Controller is Instantiable
Objective	Ensure EnableModuleController can be instantiated via reflection.
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController. 2. Verify isInstantiable returns true.
Test Data	- Input: Reflection isInstantiable value - Expected: true
Expected Result	- EnableModuleController is instantiable.
Actual Result	Instantiable.
Status	Pass
Severity	High

Test Case ID	EMC-013
Title	Multiple Instances of Controller
Objective	Confirm multiple EnableModuleController instances can be created.
Preconditions	- Application running
Test Steps	1. Instantiate EnableModuleController twice. 2. Verify both are instance of EnableModuleController. 3. Verify they are not the same object.
Test Data	- Input: Two instantiations - Expected: Two distinct objects
Expected Result	- Both objects are instances of EnableModuleController. - Objects are not the same (not identical).
Actual Result	Two distinct controller objects created.
Status	Pass
Severity	High

Test Case ID	EMC-014
Title	Constructor Has No Required Parameters
Objective	Ensure that EnableModuleController's constructor has no required parameters.
Preconditions	- Application running
Test Steps	1. Use reflection to get constructor. 2. Check constructor parameters for non-optional ones. 3. Verify no required parameters.

Test Data	- Input: Constructor parameters - Expected: No required parameters
Expected Result	- Constructor has no required parameters.
Actual Result	No required constructor parameters.
Status	Pass
Severity	High

Test Case ID	EMC-015
Title	__invoke Method Signature Validation
Objective	Check that __invoke method has two parameters, is public, and not static.
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController. 2. Get __invoke method. 3. Verify number of parameters is 2. 4. Verify method is public and not static.
Test Data	- Input: Reflection properties - Expected: Two parameters, public, not static
Expected Result	- __invoke method has two parameters. - __invoke method is public. - __invoke method is not static.
Actual Result	Method signature is as expected.
Status	Pass
Severity	High

Test Case ID	EMC-016
Title	Controller Class is Loaded
Objective	Verify class_exists returns true for EnableModuleController.
Preconditions	- Application running
Test Steps	1. Call class_exists on EnableModuleController class. 2. Verify result is true.
Test Data	- Input: class_exists(EnableModuleController::class) - Expected: true
Expected Result	- class_exists returns true.
Actual Result	Class is loaded.
Status	Pass
Severity	High

Test Case ID	EMC-017
Title	Required Imports Present in Controller
Objective	Ensure EnableModuleController uses all required classes in its file.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Use reflection to get controller file. 2. Read file contents. 3. Check for presence of required import statements.

Test Data	- Expected imports: - Crater\Events\ModuleEnabledEvent - Crater\Http\Controllers\Controller - Crater\Models\Module - Illuminate\Http\Request - Nwidart\Modules\Facades\Module
Expected Result	- All specified imports are present in file.
Actual Result	All required imports present.
Status	Pass
Severity	Medium

Test Case ID	EMC-018
Title	Controller is Not Final
Objective	Ensure EnableModuleController can be extended (not declared final).
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController. 2. Verify isFinal returns false.
Test Data	- Input: Reflection isFinal value - Expected: false
Expected Result	- EnableModuleController is not final.
Actual Result	Not final, can be extended.
Status	Pass
Severity	High

Test Case ID	EMC-019
Title	Parent Class of Controller is Controller
Objective	Check that parent class of EnableModuleController is Controller.
Preconditions	- Application running
Test Steps	1. Use reflection to get parent class. 2. Verify parent class is Controller.
Test Data	- Input: Reflection getParentClass() - Expected: Controller
Expected Result	- Parent class is Controller.
Actual Result	Parent class is Controller.
Status	Pass
Severity	High

Test Case ID	EMC-020
Title	Type Hinted Usage of EnableModuleController
Objective	Validate EnableModuleController can be used in function type hints.
Preconditions	- Application running
Test Steps	1. Define function with EnableModuleController type hint argument. 2. Pass instance to function. 3. Ensure function returns the same instance.
Test Data	- Input: EnableModuleController instance - Expected: Function accepts and returns instance correctly.
Expected Result	- Instance passed and returned correctly via type hinting.
Actual Result	Type hint works as expected.

Status	Pass
Severity	Medium

Test Case ID	EMC-021
Title	__invoke Method Visibility
Objective	Check __invoke method is public, not protected or private.
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController. 2. Access __invoke method. 3. Verify isPublic is true. 4. Verify isProtected and isPrivate are false.
Test Data	- Input: Reflection method modifiers - Expected: public, not protected, not private
Expected Result	- __invoke is public, not protected or private.
Actual Result	Visibility is correct.
Status	Pass
Severity	High

Test Case ID	EMC-022
Title	Namespace Depth Verification
Objective	Verify namespace hierarchy depth includes all expected parts.
Preconditions	- Application running
Test Steps	1. Use reflection to get namespace of EnableModuleController. 2. Split namespace by separator. 3. Ensure it contains all expected segments.
Test Data	- Expected segments: Crater, Http, Controllers, V1, Admin, Modules
Expected Result	- Namespace consists of all required segments.
Actual Result	Namespace depth is correct.
Status	Pass
Severity	Medium

Test Case ID	EMC-023
Title	Controller Cloning Functionality
Objective	Ensure EnableModuleController instance can be successfully cloned.
Preconditions	- Application running
Test Steps	1. Instantiate EnableModuleController. 2. Clone the object. 3. Verify clone is same type. 4. Verify clone is not same as original object.
Test Data	- Input: New instance, clone operation - Expected: Clone is EnableModuleController, not identical object
Expected Result	- Clone is correct type and a distinct object.
Actual Result	Cloning works as expected.
Status	Pass
Severity	High

Test Case ID	EMC-024
Title	__invoke Second Parameter Type is String

Objective	Ensure that the second parameter of __invoke is explicitly typed as string.
Preconditions	- Application running
Test Steps	1. Use reflection on EnableModuleController. 2. Get parameters of __invoke. 3. Verify second parameter has type 'string'.
Test Data	- Input: Parameter type - Expected: Second parameter type is string
Expected Result	- Second parameter of __invoke method is typed as string.
Actual Result	Second parameter is typed as string.
Status	Pass
Severity	High

File: EnvironmentManager-Test.txt

Test Case ID	EM-001
Title	Verify EnvironmentManager class is in correct namespace
Objective	Ensure the EnvironmentManager class resides in the expected 'Crater\Space' namespace.
Preconditions	- Application running - Codebase contains EnvironmentManager class - Autoloader is correctly configured
Test Steps	1. Use PHP Reflection to get the namespace of EnvironmentManager class. 2. Retrieve the namespace name. 3. Assert that the namespace name equals 'Crater\Space'.
Test Data	- Class: EnvironmentManager - Expected Namespace: 'Crater\Space'
Expected Result	- Namespace name should be 'Crater\Space'.
Actual Result	Namespace name is 'Crater\Space'.
Status	Pass
Severity	Medium

Test Case ID	EM-002
Title	Validate that EnvironmentManager class is not abstract
Objective	Confirm EnvironmentManager is a concrete class and can be instantiated.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Use Reflection to inspect EnvironmentManager class. 2. Check if the class is abstract. 3. Assert that isAbstract() returns false.
Test Data	- Class: EnvironmentManager
Expected Result	- isAbstract() returns false; class is not abstract.
Actual Result	Class is not abstract; isAbstract() returns false.
Status	Pass
Severity	Medium

Test Case ID	EM-003
Title	Check if EnvironmentManager class is instantiable
Objective	Ensure the EnvironmentManager class can be instantiated.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Inspect EnvironmentManager class using Reflection. 2. Check if the class is instantiable. 3. Assert that isInstantiable() returns true.
Test Data	- Class: EnvironmentManager
Expected Result	- isInstantiable() returns true; class can be instantiated.
Actual Result	Class is instantiable; isInstantiable() returns true.
Status	Pass
Severity	Medium

Test Case ID	EM-004

Title	Verify constructor existence and accessibility in EnvironmentManager
Objective	Ensure EnvironmentManager has a public constructor.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Use Reflection to access the EnvironmentManager class. 2. Retrieve the constructor method. 3. Assert constructor is not null. 4. Assert constructor is public.
Test Data	- Class: EnvironmentManager
Expected Result	- Constructor is not null. - Constructor is public.
Actual Result	Constructor exists and is public.
Status	Pass
Severity	Medium

Test Case ID	EM-005
Title	Ensure constructor of EnvironmentManager has no required parameters
Objective	Check that EnvironmentManager constructor does not require parameters for instantiation.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Use Reflection to get EnvironmentManager constructor. 2. Check number of parameters required by constructor. 3. Assert number of required parameters equals 0.
Test Data	- Class: EnvironmentManager - Expected Parameters: 0
Expected Result	- Constructor requires 0 parameters.
Actual Result	Constructor requires 0 parameters.
Status	Pass
Severity	Medium

Test Case ID	EM-006
Title	Confirm EnvironmentManager has private helper methods
Objective	Verify the existence of private helper methods defined in EnvironmentManager.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Use Reflection to obtain private methods in EnvironmentManager. 2. Filter methods for those defined in EnvironmentManager class. 3. Count private methods specific to EnvironmentManager. 4. Assert that there is at least one private method in the class.
Test Data	- Class: EnvironmentManager
Expected Result	- At least one private method exists in EnvironmentManager.
Actual Result	Private helper methods are present in EnvironmentManager.
Status	Pass
Severity	Medium

Test Case ID	EM-007
Title	Verify EnvironmentManager is not final

Objective	Ensure EnvironmentManager class is extendable.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Use Reflection to inspect if EnvironmentManager is marked as final. 2. Assert that isFinal() returns false.
Test Data	- Class: EnvironmentManager
Expected Result	- isFinal() returns false; class is not final.
Actual Result	Class is not final; isFinal() returns false.
Status	Pass
Severity	Medium

Test Case ID	EM-008
Title	Confirm EnvironmentManager is not an interface
Objective	Ensure EnvironmentManager is a class, not an interface.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Use Reflection to check if EnvironmentManager is an interface. 2. Assert that isInterface() returns false.
Test Data	- Class: EnvironmentManager
Expected Result	- isInterface() returns false; EnvironmentManager is not an interface.
Actual Result	Not an interface; isInterface() returns false.
Status	Pass
Severity	Medium

Test Case ID	EM-009
Title	Confirm EnvironmentManager is not a trait
Objective	Ensure EnvironmentManager class is not defined as a PHP trait.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Use Reflection to check if EnvironmentManager is a trait. 2. Assert that isTrait() returns false.
Test Data	- Class: EnvironmentManager
Expected Result	- isTrait() returns false; EnvironmentManager is not a trait.
Actual Result	Not a trait; isTrait() returns false.
Status	Pass
Severity	Medium

Test Case ID	EM-010
Title	Verify EnvironmentManager class is loaded in runtime
Objective	Check that EnvironmentManager class exists and is loaded during execution.
Preconditions	- Application running - Autoloader functioning - EnvironmentManager class defined
Test Steps	1. Use class_exists() to check if EnvironmentManager is loaded. 2. Assert that class_exists() returns true.
Test Data	- Class: EnvironmentManager
Expected Result	- class_exists(EnvironmentManager::class) returns true.

Actual Result	Class is loaded; class_exists() returns true.
Status	Pass
Severity	High

Test Case ID	EM-011
Title	Verify EnvironmentManager namespace contains correct depth and segments
Objective	Validate that EnvironmentManager's namespace is exactly two levels ('Crater\Space') and contains expected segments.
Preconditions	- Application running - EnvironmentManager class defined
Test Steps	1. Use Reflection to get EnvironmentManager namespace. 2. Split namespace string by '\' delimiter. 3. Assert that namespace includes 'Crater' and 'Space'. 4. Assert that namespace is exactly two levels deep.
Test Data	- Namespace: 'Crater\Space' - Expected Parts: ['Crater', 'Space'] - Expected Depth: 2
Expected Result	- Namespace contains 'Crater' and 'Space'. - Namespace is two levels deep.
Actual Result	Namespace contains 'Crater' and 'Space'; two levels deep.
Status	Pass
Severity	Medium

File: EstimateCollection-Test.txt

Test Case ID	EC-001
Title	Instantiation of EstimateCollection
Objective	Verify that EstimateCollection can be instantiated without errors.
Preconditions	- Application running - Necessary classes/objects autoloaded
Test Steps	1. Create an empty Collection object. 2. Instantiate EstimateCollection with the empty Collection.
Test Data	- Input: new Collection([]) - Expected: Instance of EstimateCollection
Expected Result	EstimateCollection is successfully instantiated and is an instance of EstimateCollection.
Actual Result	EstimateCollection is successfully instantiated and is an instance of EstimateCollection.
Status	Pass
Severity	Medium

Test Case ID	EC-002
Title	EstimateCollection extends ResourceCollection
Objective	Ensure EstimateCollection inherits from ResourceCollection.
Preconditions	- Application running - Necessary classes/objects autoloaded
Test Steps	1. Create an empty Collection. 2. Instantiate EstimateCollection with the empty Collection. 3. Check the type of the created object.
Test Data	- Input: new Collection([]) - Expected: Instance of ResourceCollection
Expected Result	EstimateCollection is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	EstimateCollection is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	EC-003
Title	EstimateCollection Namespace Verification
Objective	Verify that EstimateCollection is defined in the correct namespace.
Preconditions	- Application running
Test Steps	1. Instantiate ReflectionClass for EstimateCollection. 2. Retrieve the namespace name.
Test Data	- Input: EstimateCollection class - Expected: Namespace "Crater\Http\Resources"
Expected Result	Namespace for EstimateCollection is "Crater\Http\Resources".
Actual Result	Namespace for EstimateCollection is "Crater\Http\Resources".
Status	Pass
Severity	Low

Test Case ID	EC-004

Title	Presence of toArray Method in EstimateCollection
Objective	Confirm that the toArray method exists in EstimateCollection.
Preconditions	- Application running
Test Steps	1. Instantiate EstimateCollection with empty Collection. 2. Check for method "toArray" existence.
Test Data	- Input: EstimateCollection instance - Expected: Method "toArray" exists
Expected Result	EstimateCollection has a public method named "toArray".
Actual Result	EstimateCollection has a public method named "toArray".
Status	Pass
Severity	Medium

Test Case ID	EC-005
Title	toArray Method Accessibility
Objective	Ensure toArray method is public.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on EstimateCollection. 2. Retrieve "toArray" method. 3. Check method accessibility.
Test Data	- Input: Method "toArray" - Expected: Public accessibility
Expected Result	"toArray" method is public.
Actual Result	"toArray" method is public.
Status	Pass
Severity	Medium

Test Case ID	EC-006
Title	toArray Method Parameter Verification
Objective	Check that toArray method accepts "request" as its parameter.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get "toArray" method. 2. Retrieve method parameters. 3. Verify first parameter name is "request".
Test Data	- Expected parameter: "request"
Expected Result	The toArray method has exactly one parameter named "request".
Actual Result	The toArray method has exactly one parameter named "request".
Status	Pass
Severity	Medium

Test Case ID	EC-007
Title	Handling of Empty Collection by toArray
Objective	Ensure toArray method returns an empty array for an empty collection.
Preconditions	- Application running
Test Steps	1. Create a Request object. 2. Create EstimateCollection with empty Collection. 3. Call toArray with the request.
Test Data	- Input: EstimateCollection([]) - Expected: Empty array

Expected Result	toArray returns an empty array.
Actual Result	toArray returns an empty array.
Status	Pass
Severity	Medium

Test Case ID	EC-008
Title	Transformation of Single Estimate Resource
Objective	Verify that a single estimate resource is transformed correctly.
Preconditions	- Application running
Test Steps	1. Create Request object. 2. Create a dummy estimate with id=1, estimate_number="EST-001", total=1000. 3. Wrap the estimate in EstimateResource. 4. Create EstimateCollection with the resource. 5. Call toArray with the request.
Test Data	- Estimate: id=1, estimate_number="EST-001", total=1000
Expected Result	- Result is an array with 1 element. - Element at index 0 has key "id" with value 1. - Element at index 0 has key "estimate_number" with value "EST-001".
Actual Result	- Result is an array with 1 element. - Element at index 0 has key "id" with value 1. - Element at index 0 has key "estimate_number" with value "EST-001".
Status	Pass
Severity	High

Test Case ID	EC-009
Title	Transformation of Multiple Estimate Resources
Objective	Ensure multiple estimate resources are transformed into proper array format.
Preconditions	- Application running
Test Steps	1. Create Request object. 2. Create two dummy estimates: - id=1, estimate_number="EST-001" - id=2, estimate_number="EST-002" 3. Wrap the estimates in EstimateResources. 4. Create EstimateCollection with both resources. 5. Call toArray with the request.
Test Data	- Estimate 1: id=1, estimate_number="EST-001" - Estimate 2: id=2, estimate_number="EST-002"
Expected Result	- Result is an array with 2 elements. - Element 0's "id" = 1 - Element 1's "id" = 2
Actual Result	- Result is an array with 2 elements. - Element 0's "id" = 1 - Element 1's "id" = 2
Status	Pass
Severity	High

Test Case ID	EC-010
Title	Required Fields in Transformed Estimate
Objective	Verify that each transformed estimate contains all required fields.
Preconditions	- Application running

Test Steps	1. Create Request object. 2. Create dummy estimate (id=1, estimate_number="EST-001", total=1000). 3. Wrap in EstimateResource. 4. Create EstimateCollection with the resource. 5. Call toArray with the request.
Test Data	- Estimate: id=1, estimate_number="EST-001", total=1000
Expected Result	Returned item has keys: - id - estimate_date - expiry_date - estimate_number - status - total - sub_total - tax
Actual Result	Returned item has keys: - id - estimate_date - expiry_date - estimate_number - status - total - sub_total - tax
Status	Pass
Severity	High

Test Case ID	EC-011
Title	Handling Large Collections
Objective	Confirm that toArray handles collections with 100 estimate resources.
Preconditions	- Application running
Test Steps	1. Create Request object. 2. Create 100 dummy estimates (ids 1-100). 3. Wrap each in EstimateResource. 4. Create EstimateCollection with all resources. 5. Call toArray with the request.
Test Data	- Estimates: id=1..100, estimate_numbers="EST-001.."EST-100"
Expected Result	- Result is an array with 100 elements. - First element "id"=1. - Last element "id"=100.
Actual Result	- Result is an array with 100 elements. - First element "id"=1. - Last element "id"=100.
Status	Pass
Severity	High

Test Case ID	EC-012
Title	toArray Delegation to Parent ResourceCollection
Objective	Verify that toArray method calls parent::toArray for delegation.
Preconditions	- Application running
Test Steps	1. Create ReflectionClass for EstimateCollection. 2. Read file contents. 3. Confirm "parent::toArray" appears in the file.
Test Data	- Source code content
Expected Result	The EstimateCollection file contains "parent::toArray".

Actual Result	The EstimateCollection file contains "parent::toArray".
Status	Pass
Severity	Medium

Test Case ID	EC-013
Title	Parent Class of EstimateCollection is ResourceCollection
Objective	Verify parent class inheritance.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Get parent class. 3. Confirm parent is Illuminate\Http\Resources\Json\ResourceCollection.
Test Data	- EstimateCollection class hierarchy
Expected Result	Parent is Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	Parent is Illuminate\Http\Resources\Json\ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	EC-014
Title	Creation of Multiple EstimateCollection Instances
Objective	Ensure multiple instances of EstimateCollection can be created independently.
Preconditions	- Application running
Test Steps	1. Create two separate EstimateCollection instances with an empty Collection. 2. Compare the instances.
Test Data	- new EstimateCollection(new Collection([]))
Expected Result	Both objects are instances of EstimateCollection and are not the same reference.
Actual Result	Both objects are instances of EstimateCollection and are not the same reference.
Status	Pass
Severity	Low

Test Case ID	EC-015
Title	Cloning of EstimateCollection Instances
Objective	Verify that EstimateCollection objects can be cloned correctly.
Preconditions	- Application running
Test Steps	1. Create EstimateCollection instance. 2. Clone the instance. 3. Compare the clone with the original.
Test Data	- Instance of EstimateCollection
Expected Result	Clone is an instance of EstimateCollection and is not the same as the original object.
Actual Result	Clone is an instance of EstimateCollection and is not the same as the original object.
Status	Pass
Severity	Low

Test Case ID	EC-016

Title	Valid Usage of EstimateCollection in Type Hints
Objective	Confirm EstimateCollection can be used in function argument type hints.
Preconditions	- Application running
Test Steps	1. Define a function accepting EstimateCollection as parameter. 2. Pass an EstimateCollection instance to the function. 3. Compare the returned result to the original instance.
Test Data	- Function: function (EstimateCollection $collection) - Input: EstimateCollection instance
Expected Result	Function returns the same EstimateCollection instance passed in.
Actual Result	Function returns the same EstimateCollection instance passed in.
Status	Pass
Severity	Low

Test Case ID	EC-017
Title	EstimateCollection Is Not Abstract
Objective	Verify EstimateCollection class is not declared as abstract.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Check isAbstract property.
Test Data	- EstimateCollection class
Expected Result	EstimateCollection is not abstract.
Actual Result	EstimateCollection is not abstract.
Status	Pass
Severity	Low

Test Case ID	EC-018
Title	EstimateCollection Is Not Final
Objective	Confirm that EstimateCollection is not a final class.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Check isFinal property.
Test Data	- EstimateCollection class
Expected Result	EstimateCollection is not final.
Actual Result	EstimateCollection is not final.
Status	Pass
Severity	Low

Test Case ID	EC-019
Title	EstimateCollection Is Not an Interface
Objective	Ensure EstimateCollection is not an interface.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Check isInterface property.
Test Data	- EstimateCollection class
Expected Result	EstimateCollection is not an interface.
Actual Result	EstimateCollection is not an interface.

Status	Pass
Severity	Low

Test Case ID	EC-020
Title	EstimateCollection Is Not a Trait
Objective	Verify EstimateCollection is not implemented as a trait.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Check isTrait property.
Test Data	- EstimateCollection class
Expected Result	EstimateCollection is not a trait.
Actual Result	EstimateCollection is not a trait.
Status	Pass
Severity	Low

Test Case ID	EC-021
Title	Class Existence for EstimateCollection
Objective	Ensure EstimateCollection class is loaded and exists.
Preconditions	- Application running
Test Steps	1. Check if class_exists for EstimateCollection returns true.
Test Data	- EstimateCollection class
Expected Result	class_exists(EstimateCollection::class) returns true.
Actual Result	class_exists(EstimateCollection::class) returns true.
Status	Pass
Severity	Medium

Test Case ID	EC-022
Title	EstimateCollection Uses ResourceCollection Import
Objective	Verify that file imports ResourceCollection.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Check file contents for ResourceCollection import.
Test Data	- File contents
Expected Result	File contains "use Illuminate\Http\Resources\Json\ResourceCollection".
Actual Result	File contains "use Illuminate\Http\Resources\Json\ResourceCollection".
Status	Pass
Severity	Low

Test Case ID	EC-023
Title	toArray Method Is Not Static
Objective	Ensure toArray method is an instance method, not static.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Retrieve toArray method. 3. Check isStatic property.
Test Data	- Method "toArray"

Expected Result	toArray method is not static.
Actual Result	toArray method is not static.
Status	Pass
Severity	Medium

Test Case ID	EC-024
Title	toArray Method Is Not Abstract
Objective	Confirm that toArray method is concretely implemented.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Retrieve toArray method. 3. Check isAbstract property.
Test Data	- Method "toArray"
Expected Result	toArray method is not abstract.
Actual Result	toArray method is not abstract.
Status	Pass
Severity	Medium

Test Case ID	EC-025
Title	Estimate Data Integrity Preservation
Objective	Ensure toArray method does not alter estimate fields and values.
Preconditions	- Application running
Test Steps	1. Create Request object. 2. Create dummy estimate (id=42, estimate_number="EST-042", total=5000, status="DRAFT"). 3. Wrap in EstimateResource. 4. Create EstimateCollection with the resource. 5. Call toArray with the request.
Test Data	- id=42, estimate_number="EST-042", total=5000, status="DRAFT"
Expected Result	- Returned "id" is 42 - "estimate_number" is "EST-042" - "total" is 5000 - "status" is "DRAFT"
Actual Result	- Returned "id" is 42 - "estimate_number" is "EST-042" - "total" is 5000 - "status" is "DRAFT"
Status	Pass
Severity	High

Test Case ID	EC-026
Title	Handling Different Estimate Status Values
Objective	Verify multiple status values are properly represented after transformation.
Preconditions	- Application running
Test Steps	1. Create Request object. 2. Create two dummy estimates: - Estimate 1: id=1, status="DRAFT" - Estimate 2: id=2, status="SENT" 3. Wrap in EstimateResources. 4. Create EstimateCollection with both resources. 5. Call toArray with the request.

Test Data	- Estimate 1: id=1, status="DRAFT" - Estimate 2: id=2, status="SENT"
Expected Result	- First entry "status" is "DRAFT" - Second entry "status" is "SENT"
Actual Result	- First entry "status" is "DRAFT" - Second entry "status" is "SENT"
Status	Pass
Severity	High

Test Case ID	EC-027
Title	File Size of EstimateCollection Class
Objective	Ensure EstimateCollection source file is small and concise (<1000 bytes).
Preconditions	- Application running - File system access
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Read file contents. 3. Measure byte count.
Test Data	- File content length
Expected Result	File size is less than 1000 bytes.
Actual Result	File size is less than 1000 bytes.
Status	Pass
Severity	Low

Test Case ID	EC-028
Title	Minimal Line Count in EstimateCollection File
Objective	Verify that EstimateCollection file has less than 30 lines.
Preconditions	- Application running - File system access
Test Steps	1. Use ReflectionClass for EstimateCollection. 2. Read file contents. 3. Count number of lines.
Test Data	- File line count
Expected Result	EstimateCollection file has fewer than 30 lines.
Actual Result	EstimateCollection file has fewer than 30 lines.
Status	Pass
Severity	Low

File: EstimateItem-Test.txt

Test Case ID	EI-001
Title	EstimateItem can be instantiated
Objective	Verify that an instance of EstimateItem can be created successfully.
Preconditions	- Application running - Database seeded - EstimateItem class is loaded
Test Steps	1. Instantiate a new EstimateItem object. 2. Verify the object is instance of EstimateItem::class.
Test Data	- Input: None (object instantiation) - Expected: Instance is of EstimateItem::class
Expected Result	- EstimateItem object is instantiated and is of class EstimateItem.
Actual Result	EstimateItem object successfully created and verified as instance of EstimateItem.
Status	Pass
Severity	Medium

Test Case ID	EI-002
Title	EstimateItem extends Model
Objective	Ensure EstimateItem class extends the Eloquent Model class.
Preconditions	- Application running - Database seeded - EstimateItem and Model classes are loaded
Test Steps	1. Instantiate a new EstimateItem object. 2. Verify the object is instance of Illuminate\Database\Eloquent\Model::class.
Test Data	- Input: None (object instantiation) - Expected: Instance is of Model::class
Expected Result	EstimateItem is correctly recognized as an instance of Model.
Actual Result	EstimateItem object is instance of Model.
Status	Pass
Severity	Medium

Test Case ID	EI-003
Title	EstimateItem is in correct namespace
Objective	Validate EstimateItem class is defined within the Crater\Models namespace.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass to get EstimateItem. 2. Retrieve its namespace name. 3. Assert that namespace is 'Crater\Models'.
Test Data	- Input: None - Expected: Namespace is 'Crater\Models'
Expected Result	EstimateItem class belongs to 'Crater\Models' namespace.
Actual Result	Namespace verified as 'Crater\Models'.
Status	Pass
Severity	Low

Test Case ID	EI-004

Title	EstimateItem is not abstract
Objective	Confirm that EstimateItem class is not marked abstract.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass to inspect EstimateItem. 2. Check if the class is abstract.
Test Data	- Input: None - Expected: isAbstract() is false
Expected Result	EstimateItem is not abstract class.
Actual Result	EstimateItem is confirmed to not be abstract.
Status	Pass
Severity	Low

Test Case ID	EI-005
Title	EstimateItem is instantiable
Objective	Ensure EstimateItem class can be instantiated.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass for EstimateItem. 2. Verify isInstantiable() returns true.
Test Data	- Input: None - Expected: isInstantiable() is true
Expected Result	EstimateItem class is instantiable.
Actual Result	EstimateItem verified as instantiable.
Status	Pass
Severity	Medium

Test Case ID	EI-006
Title	EstimateItem has guarded properties
Objective	Confirm that EstimateItem has expected guarded property configuration.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Retrieve guarded properties via getGuarded(). 3. Assert guarded array is ['id'].
Test Data	- Input: None - Expected: Guarded properties are ['id']
Expected Result	EstimateItem getGuarded() returns array ['id'].
Actual Result	getGuarded() correctly returns ['id'].
Status	Pass
Severity	High

Test Case ID	EI-007
Title	Guarded property prevents id from mass assignment
Objective	Check that 'id' is present in the guarded property and thus protected from mass assignment.
Preconditions	- Application running - Database seeded

Test Steps	1. Instantiate EstimateItem. 2. Retrieve guarded array. 3. Assert array contains 'id'.
Test Data	- Input: None - Expected: Guarded array contains 'id'
Expected Result	'id' is present in guarded property array.
Actual Result	'id' confirmed as guarded.
Status	Pass
Severity	High

Test Case ID	EI-008
Title	Price is cast to integer
Objective	Verify that the price property on EstimateItem is cast to integer.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set price to 10000. 3. Check price is integer and equals 10000.
Test Data	- Input: price = 10000 - Expected: price is of type int, value is 10000
Expected Result	- price is integer - price equals 10000
Actual Result	price casted and stored as integer 10000.
Status	Pass
Severity	Medium

Test Case ID	EI-009
Title	Total is cast to integer
Objective	Verify that the total property on EstimateItem is cast to integer.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set total to 20000. 3. Check total is integer and equals 20000.
Test Data	- Input: total = 20000 - Expected: total is int, value is 20000
Expected Result	- total is integer - total equals 20000
Actual Result	total successfully cast to integer 20000.
Status	Pass
Severity	Medium

Test Case ID	EI-010
Title	Discount is cast to float
Objective	Verify EstimateItem discount property is cast to float.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount to 15.5. 3. Verify discount is float and equals 15.5.

Test Data	- Input: discount = 15.5 - Expected: float, 15.5
Expected Result	- discount is float - discount equals 15.5
Actual Result	discount successfully cast to float 15.5.
Status	Pass
Severity	Medium

Test Case ID	EI-011
Title	Quantity is cast to float
Objective	Ensure EstimateItem quantity is cast to float.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set quantity to 2.5. 3. Assert quantity is float and equals 2.5.
Test Data	- Input: quantity = 2.5 - Expected: float, 2.5
Expected Result	- quantity is float - quantity equals 2.5
Actual Result	quantity successfully cast to float 2.5.
Status	Pass
Severity	Medium

Test Case ID	EI-012
Title	Discount_val is cast to integer
Objective	Validate discount_val property on EstimateItem is cast to integer.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount_val to 500. 3. Verify discount_val is integer, equals 500.
Test Data	- Input: discount_val = 500 - Expected: int, 500
Expected Result	- discount_val is integer - discount_val equals 500
Actual Result	discount_val successfully set as integer 500.
Status	Pass
Severity	Medium

Test Case ID	EI-013
Title	Tax is cast to integer
Objective	Ensure EstimateItem tax property is cast to integer.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set tax to 100. 3. Verify tax is integer and equals 100.
Test Data	- Input: tax = 100 - Expected: int, 100

Expected Result	- tax is integer - tax equals 100
Actual Result	tax casted to integer 100.
Status	Pass
Severity	Medium

Test Case ID	EI-014
Title	Price casts string to integer
Objective	Test string value assignment for price is properly cast to integer.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set price to '10000' (string). 3. Verify price is integer and equals 10000.
Test Data	- Input: price = '10000' (string) - Expected: int, 10000
Expected Result	- price is integer - price equals 10000
Actual Result	price string converted to integer 10000.
Status	Pass
Severity	Medium

Test Case ID	EI-015
Title	Discount casts string to float
Objective	Verify string assignment for discount is properly cast to float.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount to '15.5' (string). 3. Check discount is float and equals 15.5.
Test Data	- Input: discount = '15.5' (string) - Expected: float, 15.5
Expected Result	- discount is float - discount equals 15.5
Actual Result	discount string converted to float 15.5.
Status	Pass
Severity	Medium

Test Case ID	EI-016
Title	Price handles null values
Objective	Ensure assigning null to price property is handled correctly.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set price to null. 3. Verify price is null.
Test Data	- Input: price = null - Expected: null
Expected Result	price property is null.

Actual Result	price correctly stores null.
Status	Pass
Severity	Medium

Test Case ID	EI-017
Title	Total handles null values
Objective	Test that total property on EstimateItem accepts and maintains null values.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set total to null. 3. Assert total is null.
Test Data	- Input: total = null - Expected: null
Expected Result	total property is null.
Actual Result	total correctly set to null.
Status	Pass
Severity	Medium

Test Case ID	EI-018
Title	Discount handles null values
Objective	Verify discount property properly stores null value.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount to null. 3. Assert discount is null.
Test Data	- Input: discount = null - Expected: null
Expected Result	discount is null.
Actual Result	discount property is null.
Status	Pass
Severity	Medium

Test Case ID	EI-019
Title	Quantity handles null values
Objective	Ensure quantity property accepts and stores null value.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set quantity to null. 3. Assert quantity is null.
Test Data	- Input: quantity = null - Expected: null
Expected Result	quantity property is null.
Actual Result	quantity property successfully set to null.
Status	Pass
Severity	Medium

Test Case ID	EI-020
Title	Discount_val handles null values
Objective	Check that discount_val property allows null assignments.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount_val to null. 3. Assert discount_val is null.
Test Data	- Input: discount_val = null - Expected: null
Expected Result	discount_val is null.
Actual Result	discount_val stored as null.
Status	Pass
Severity	Medium

Test Case ID	EI-021
Title	Tax handles null values
Objective	Confirm that EstimateItem tax property supports null values.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set tax to null. 3. Assert tax is null.
Test Data	- Input: tax = null - Expected: null
Expected Result	tax is null.
Actual Result	tax property is null.
Status	Pass
Severity	Medium

Test Case ID	EI-022
Title	Price casts invalid string to zero
Objective	Ensure assigning invalid string to price results in value zero.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set price to 'invalid_price'. 3. Assert price is integer and equals 0.
Test Data	- Input: price = 'invalid_price' - Expected: int, 0
Expected Result	- price is integer - price is zero
Actual Result	Invalid price string cast to integer 0.
Status	Pass
Severity	Medium

Test Case ID	EI-023
Title	Total casts invalid string to zero

Objective	Check assigning invalid string to total yields zero.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set total to 'invalid_total'. 3. Assert total is integer and equals 0.
Test Data	- Input: total = 'invalid_total' - Expected: int, 0
Expected Result	- total is integer - total is zero
Actual Result	Invalid total string successfully cast to zero.
Status	Pass
Severity	Medium

Test Case ID	EI-024
Title	Discount casts invalid string to zero float
Objective	Confirm discount set to invalid string results in 0.0.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount to 'invalid_discount'. 3. Assert discount is float and equals 0.0.
Test Data	- Input: discount = 'invalid_discount' - Expected: float, 0.0
Expected Result	- discount is float - discount is 0.0
Actual Result	Invalid discount string cast to float 0.0.
Status	Pass
Severity	Medium

Test Case ID	EI-025
Title	Quantity casts invalid string to zero float
Objective	Ensure invalid string assignment to quantity results in 0.0.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set quantity to 'invalid_quantity'. 3. Assert quantity is float and equals 0.0.
Test Data	- Input: quantity = 'invalid_quantity' - Expected: float, 0.0
Expected Result	- quantity is float - quantity is 0.0
Actual Result	Invalid quantity string cast to float zero.
Status	Pass
Severity	Medium

Test Case ID	EI-026
Title	Discount_val casts invalid string to zero
Objective	Ensure invalid string assignment to discount_val yields zero.

Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount_val to 'invalid_discount_val'. 3. Assert discount_val is integer and equals 0.
Test Data	- Input: discount_val = 'invalid_discount_val' - Expected: int, 0
Expected Result	discount_val property becomes integer zero.
Actual Result	Invalid string converted to integer zero.
Status	Pass
Severity	Medium

Test Case ID	EI-027
Title	Tax casts invalid string to zero
Objective	Confirm tax set to invalid string produces value zero.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set tax to 'invalid_tax'. 3. Assert tax is integer and equals 0.
Test Data	- Input: tax = 'invalid_tax' - Expected: int, 0
Expected Result	tax property cast to integer zero.
Actual Result	Invalid string set as integer zero.
Status	Pass
Severity	Medium

Test Case ID	EI-028
Title	Estimate method exists
Objective	Validate the existence of 'estimate' method on EstimateItem.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Use method_exists to check for 'estimate' method.
Test Data	- Input: None - Expected: method exists
Expected Result	'estimate' method is present on EstimateItem.
Actual Result	Method 'estimate' verified to exist.
Status	Pass
Severity	High

Test Case ID	EI-029
Title	Estimate relationship returns BelongsTo
Objective	Confirm 'estimate' method returns a BelongsTo relation.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Call estimate() on instance. 3. Assert returned value is instance of BelongsTo.

Test Data	- Input: None - Expected: BelongsTo relation object
Expected Result	Relationship is of type BelongsTo.
Actual Result	estimate() returns BelongsTo relation.
Status	Pass
Severity	High

Test Case ID	EI-030
Title	Estimate relationship is to Estimate model
Objective	Ensure the 'estimate' relation links to the Estimate model.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Retrieve estimate relation. 3. Call getRelated() and verify it's Estimate.
Test Data	- Input: None - Expected: Related model is Estimate
Expected Result	estimate relationship relates to Estimate model.
Actual Result	Related model confirmed as Estimate.
Status	Pass
Severity	High

Test Case ID	EI-031
Title	Item method exists
Objective	Validate EstimateItem has an 'item' method.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Check method_exists for 'item'.
Test Data	- Input: None - Expected: method exists
Expected Result	Method 'item' exists on EstimateItem.
Actual Result	Method 'item' verified.
Status	Pass
Severity	High

Test Case ID	EI-032
Title	Item relationship returns BelongsTo
Objective	Check that 'item' method returns a BelongsTo relation.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Call item() method. 3. Verify result is instance of BelongsTo.
Test Data	- Input: None - Expected: BelongsTo relation
Expected Result	item() returns BelongsTo relation.
Actual Result	Instance of BelongsTo received.

Status	Pass
Severity	High

Test Case ID	EI-033
Title	Item relationship is to Item model
Objective	Ensure EstimateItem's 'item' relationship relates to the Item model.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Call item() to get relationship. 3. Use getRelated() to verify related model is Item.
Test Data	- Input: None - Expected: Related model is Item.
Expected Result	Relationship links to Item model.
Actual Result	item() links to Item model as expected.
Status	Pass
Severity	High

Test Case ID	EI-034
Title	Taxes method exists
Objective	Confirm the existence of 'taxes' method on EstimateItem.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Use method_exists to check for 'taxes'.
Test Data	- Input: None - Expected: method exists
Expected Result	'taxes' method exists.
Actual Result	Method 'taxes' confirmed.
Status	Pass
Severity	High

Test Case ID	EI-035
Title	Taxes relationship returns HasMany
Objective	Validate 'taxes' method returns HasMany relation.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Call taxes() method. 3. Assert result is instance of HasMany.
Test Data	- Input: None - Expected: HasMany relation
Expected Result	taxes() returns HasMany relationship.
Actual Result	HasMany instance returned.
Status	Pass
Severity	High

Test Case ID	EI-036

Title	Taxes relationship is to Tax model
Objective	Ensure 'taxes' relationship links to Tax model.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Get taxes relation instance. 3. Call getRelated() and confirm it is Tax.
Test Data	- Input: None - Expected: Related model is Tax
Expected Result	Relationship relates to Tax model.
Actual Result	Related model is Tax as required.
Status	Pass
Severity	High

Test Case ID	EI-037
Title	scopeWhereCompany method exists
Objective	Confirm scopeWhereCompany method exists on EstimateItem.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Use method_exists for 'scopeWhereCompany'.
Test Data	- Input: None - Expected: method exists
Expected Result	Method 'scopeWhereCompany' exists.
Actual Result	Method located on EstimateItem.
Status	Pass
Severity	High

Test Case ID	EI-038
Title	scopeWhereCompany is public
Objective	Ensure scopeWhereCompany method is public.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass to inspect EstimateItem. 2. Retrieve Method 'scopeWhereCompany'. 3. Assert method is public.
Test Data	- Input: None - Expected: isPublic() == true
Expected Result	Method scopeWhereCompany is public.
Actual Result	scopeWhereCompany is public.
Status	Pass
Severity	High

Test Case ID	EI-039
Title	scopeWhereCompany accepts two parameters
Objective	Verify that method scopeWhereCompany takes two arguments named 'query' and 'company_id'.
Preconditions	- Application running - Database seeded

Test Steps	1. Use ReflectionClass to inspect EstimateItem. 2. Get method 'scopeWhereCompany'. 3. Get parameters and their names. 4. Assert parameter count is 2 and names are 'query' and 'company_id'.
Test Data	- Input: None - Expected: Two parameters, 'query', 'company_id'
Expected Result	scopeWhereCompany accepts parameters 'query', 'company_id'.
Actual Result	Method parameters match expected.
Status	Pass
Severity	High

Test Case ID	EI-040
Title	EstimateItem uses HasFactory trait
Objective	Confirm EstimateItem utilizes HasFactory trait.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. Get trait names. 3. Assert trait list contains 'Illuminate\Database\Eloquent\Factories\HasFactory'.
Test Data	- Input: None - Expected: trait present
Expected Result	HasFactory trait is used in EstimateItem.
Actual Result	HasFactory trait confirmed.
Status	Pass
Severity	High

Test Case ID	EI-041
Title	EstimateItem uses HasCustomFieldsTrait
Objective	Validate HasCustomFieldsTrait is used by EstimateItem.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. Retrieve trait names. 3. Assert trait list contains 'Crater\Traits\HasCustomFieldsTrait'.
Test Data	- Input: None - Expected: trait present
Expected Result	HasCustomFieldsTrait found on EstimateItem.
Actual Result	Trait present as required.
Status	Pass
Severity	High

Test Case ID	EI-042
Title	All relationship methods are public
Objective	Ensure EstimateItem's relationship methods are public.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. Retrieve methods: estimate, item, taxes. 3. Assert each method is public.

Test Data	- Input: None - Expected: methods are public
Expected Result	estimate, item, taxes methods are public.
Actual Result	All relationship methods are public.
Status	Pass
Severity	High

Test Case ID	EI-043
Title	All relationship methods are not static
Objective	Confirm that relationship methods (estimate, item, taxes) are not static.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. For each relationship method, assert isStatic() is false.
Test Data	- Input: None - Expected: methods are not static
Expected Result	All relationship methods are non-static.
Actual Result	Methods confirmed as non-static.
Status	Pass
Severity	High

Test Case ID	EI-044
Title	All relationship methods have no parameters
Objective	Validate relationship methods on EstimateItem have no arguments.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. For each relationship method, verify getNumberOfParameters() is zero.
Test Data	- Input: None - Expected: Number of parameters is 0
Expected Result	Methods require no parameters.
Actual Result	Methods have zero parameters.
Status	Pass
Severity	High

Test Case ID	EI-045
Title	Multiple EstimateItem instances can be created
Objective	Test ability to create multiple, unique instances of EstimateItem.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate two EstimateItem objects. 2. Verify both are instances of EstimateItem. 3. Assert that objects are not identical.
Test Data	- Input: None - Expected: Two unique instances
Expected Result	Can create two distinct EstimateItem instances.
Actual Result	Instances created and confirmed distinct.
Status	Pass

Severity	Medium

Test Case ID	EI-046
Title	EstimateItem can be cloned
Objective	Ensure EstimateItem object is cloneable and clones are unique instances.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Clone the object. 3. Assert clone is instance of EstimateItem and is not original object.
Test Data	- Input: None - Expected: Clone produced, is distinct
Expected Result	EstimateItem can be cloned and is unique.
Actual Result	Cloning produces a distinct EstimateItem instance.
Status	Pass
Severity	Medium

Test Case ID	EI-047
Title	EstimateItem can be used in type hints
Objective	Confirm EstimateItem supports type hinting in PHP functions.
Preconditions	- Application running - Database seeded
Test Steps	1. Define a function with EstimateItem as argument. 2. Pass EstimateItem instance to function. 3. Assert return value matches passed value.
Test Data	- Input: EstimateItem instance - Expected: function returns EstimateItem instance unchanged
Expected Result	EstimateItem can be passed via type hints.
Actual Result	Function returned passed EstimateItem as expected.
Status	Pass
Severity	Low

Test Case ID	EI-048
Title	EstimateItem is not final
Objective	Ensure EstimateItem can be subclassed, i.e., it is not a final class.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. Check isFinal() returns false.
Test Data	- Input: None - Expected: isFinal() is false
Expected Result	EstimateItem class is not final.
Actual Result	Class is not marked as final.
Status	Pass
Severity	Low

Test Case ID	EI-049
Title	EstimateItem is not an interface

Objective	Check EstimateItem class is not an interface.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. Check isInterface() returns false.
Test Data	- Input: None - Expected: isInterface() is false
Expected Result	EstimateItem is a class, not an interface.
Actual Result	Class is confirmed as non-interface.
Status	Pass
Severity	Low

Test Case ID	EI-050
Title	EstimateItem is not a trait
Objective	Confirm EstimateItem is not a trait.
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. Check isTrait() returns false.
Test Data	- Input: None - Expected: isTrait() is false
Expected Result	EstimateItem is not a trait.
Actual Result	Class is not a trait.
Status	Pass
Severity	Low

Test Case ID	EI-051
Title	EstimateItem class is loaded
Objective	Validate EstimateItem class is loaded in PHP runtime.
Preconditions	- Application running - Database seeded
Test Steps	1. Use class_exists with EstimateItem::class. 2. Assert it returns true.
Test Data	- Input: None - Expected: class_exists returns true
Expected Result	EstimateItem class is loaded.
Actual Result	Class successfully loaded.
Status	Pass
Severity	Low

Test Case ID	EI-052
Title	All casts are properly configured
Objective	Ensure all expected cast properties are configured for EstimateItem.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Get $casts property. 3. Assert keys for 'price', 'total', 'discount', 'quantity', 'discount_val', 'tax' exist.

Test Data	- Input: None - Expected: All keys present
Expected Result	$casts contains all required keys.
Actual Result	All cast keys are present and accounted for.
Status	Pass
Severity	Medium

Test Case ID	EI-053
Title	Integer casts are configured correctly
Objective	Ensure EstimateItem integer cast attributes are set to 'integer' type.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Retrieve $casts property. 3. Verify type for 'price', 'total', 'discount_val', 'tax' is 'integer'.
Test Data	- Input: None - Expected: Each specified key is set to 'integer'
Expected Result	All listed integer casts configured correctly.
Actual Result	Integer cast types confirmed.
Status	Pass
Severity	Medium

Test Case ID	EI-054
Title	Float casts are configured correctly
Objective	Ensure EstimateItem float cast attributes are set to 'float' type.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Access $casts property. 3. Check 'discount', 'quantity' keys are set to 'float'.
Test Data	- Input: None - Expected: Each key is 'float'
Expected Result	Float cast attributes are set correctly.
Actual Result	Float cast settings are correct.
Status	Pass
Severity	Medium

Test Case ID	EI-055
Title	Can set and get price attribute
Objective	Validate setting and getting price field works as intended.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set price to 5000. 3. Assert price is retrievable and equals 5000.
Test Data	- Input: price = 5000 - Expected: price is 5000
Expected Result	price attribute is 5000.
Actual Result	price correctly saved and retrieved.

Status	Pass
Severity	Medium

Test Case ID	EI-056
Title	Can set and get total attribute
Objective	Test read/write for total property.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set total to 10000. 3. Assert total retrieved is 10000.
Test Data	- Input: total = 10000 - Expected: total is 10000
Expected Result	total can be read after write.
Actual Result	total retrieved as written.
Status	Pass
Severity	Medium

Test Case ID	EI-057
Title	Can set and get discount attribute
Objective	Validate discount property's read/write functionality.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount to 10.5. 3. Assert discount value is 10.5.
Test Data	- Input: discount = 10.5 - Expected: discount is 10.5
Expected Result	discount attribute saved and loaded as 10.5.
Actual Result	discount retrieved as expected.
Status	Pass
Severity	Medium

Test Case ID	EI-058
Title	Can set and get quantity attribute
Objective	Ensure read/write for quantity works.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set quantity to 3.5. 3. Assert quantity is 3.5.
Test Data	- Input: quantity = 3.5 - Expected: quantity is 3.5
Expected Result	quantity saved and fetched as 3.5.
Actual Result	quantity value correct.
Status	Pass
Severity	Medium

Test Case ID	EI-059
Title	Can set and get discount_val attribute
Objective	Validate ability to save and retrieve discount_val value.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount_val to 1000. 3. Assert discount_val is 1000.
Test Data	- Input: discount_val = 1000 - Expected: discount_val is 1000
Expected Result	discount_val is set and returned as 1000.
Actual Result	discount_val properly handled.
Status	Pass
Severity	Medium

Test Case ID	EI-060
Title	Can set and get tax attribute
Objective	Ensure saving and loading tax property works.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set tax to 200. 3. Assert tax retrieved is 200.
Test Data	- Input: tax = 200 - Expected: tax is 200
Expected Result	tax attribute set and loaded as 200.
Actual Result	tax retrieved correctly.
Status	Pass
Severity	Medium

Test Case ID	EI-061
Title	Handles decimal string for integer cast
Objective	Check decimal string assigned to price is cast to integer (truncation).
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set price to '10000.99' (string). 3. Assert price is integer 10000.
Test Data	- Input: price = '10000.99' - Expected: int, 10000
Expected Result	Decimal string cast to integer, fractional part lost.
Actual Result	price stored as integer 10000.
Status	Pass
Severity	Medium

Test Case ID	EI-062
Title	Handles negative values for integer cast
Objective	Test negative integer assignment for price property.

Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set price to -5000. 3. Assert price is integer -5000.
Test Data	- Input: price = -5000 - Expected: int, -5000
Expected Result	Negative integer handled correctly.
Actual Result	Negative value properly stored.
Status	Pass
Severity	Medium

Test Case ID	EI-063
Title	Handles negative values for float cast
Objective	Check negative float assignment for discount property.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set discount to -10.5. 3. Assert discount is float, -10.5.
Test Data	- Input: discount = -10.5 - Expected: float, -10.5
Expected Result	Negative float preserved.
Actual Result	Negative float value handled correctly.
Status	Pass
Severity	Medium

Test Case ID	EI-064
Title	Handles zero values correctly
Objective	Ensure properties price and discount handle zero assignment correctly.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimateItem. 2. Set price to 0, discount to 0.0. 3. Assert values are zero.
Test Data	- Input: price = 0, discount = 0.0 - Expected: price = 0, discount = 0.0
Expected Result	Zero values allowed and preserved.
Actual Result	Zero values stored correctly.
Status	Pass
Severity	Medium

Test Case ID	EI-065
Title	EstimateItem file has expected structure
Objective	Validate file contains expected structural elements: class, guarded, casts.
Preconditions	- Application running - Database seeded

Test Steps	1. Use ReflectionClass on EstimateItem. 2. Read file contents. 3. Assert file contains class declaration, $guarded, $casts.
Test Data	- Input: None - Expected: File contains structure markers
Expected Result	File has required structure.
Actual Result	File includes all structural items.
Status	Pass
Severity	Low

Test Case ID	EI-066
Title	EstimateItem has compact implementation
Objective	Ensure EstimateItem class file size is concise (< 2000 bytes).
Preconditions	- Application running - Database seeded
Test Steps	1. Use ReflectionClass on EstimateItem. 2. Read file and measure length. 3. Assert file is < 2000 bytes.
Test Data	- Input: None - Expected: File size < 2000 bytes
Expected Result	Implementation is compact and efficient.
Actual Result	File size below 2000 bytes verified.
Status	Pass
Severity	Low

File: EstimateItemCollection-Test.txt

Test Case ID	EIC-001
Title	Instantiation of EstimateItemCollection
Objective	Verify that an EstimateItemCollection object can be successfully instantiated.
Preconditions	- Application running - Required classes imported (EstimateItemCollection, Collection) - No collection items necessary
Test Steps	1. Instantiate EstimateItemCollection with an empty Collection. 2. Check if the instantiated object is of type EstimateItemCollection.
Test Data	- Collection: []
Expected Result	- The instance is of type EstimateItemCollection.
Actual Result	The object was successfully instantiated as an EstimateItemCollection.
Status	Pass
Severity	Medium

Test Case ID	EIC-002
Title	Inheritance from ResourceCollection
Objective	Validate that EstimateItemCollection extends ResourceCollection.
Preconditions	- Application running - Required classes imported
Test Steps	1. Instantiate EstimateItemCollection with an empty Collection. 2. Verify that the instance is of type ResourceCollection.
Test Data	- Collection: []
Expected Result	- EstimateItemCollection instance is instanceof ResourceCollection.
Actual Result	Instance is correctly a ResourceCollection subclass.
Status	Pass
Severity	Medium

Test Case ID	EIC-003
Title	Namespace Validation of EstimateItemCollection
Objective	Confirm EstimateItemCollection is in the correct namespace.
Preconditions	- Application running - Required classes available
Test Steps	1. Use ReflectionClass on EstimateItemCollection. 2. Retrieve the class namespace. 3. Verify the namespace is 'Crater\Http\Resources'.
Test Data	- Class: EstimateItemCollection
Expected Result	- Namespace is 'Crater\Http\Resources'.
Actual Result	Namespace was ‘Crater\Http\Resources’ as expected.
Status	Pass
Severity	Low

Test Case ID	EIC-004
Title	Existence of toArray Method
Objective	Ensure that EstimateItemCollection contains a 'toArray' method.

Preconditions	- Application running - Required class definitions available
Test Steps	1. Instantiate EstimateItemCollection. 2. Check if 'toArray' method exists for the instance.
Test Data	- Class: EstimateItemCollection
Expected Result	- Method 'toArray' exists.
Actual Result	'toArray' method exists in EstimateItemCollection.
Status	Pass
Severity	Medium

Test Case ID	EIC-005
Title	Public Visibility of toArray Method
Objective	Verify 'toArray' method is public.
Preconditions	- Application running - Class definitions loaded
Test Steps	1. Use ReflectionClass to inspect EstimateItemCollection. 2. Obtain ReflectionMethod for 'toArray'. 3. Check that the method is public.
Test Data	- Class: EstimateItemCollection
Expected Result	- 'toArray' method visibility is public.
Actual Result	Method was public as expected.
Status	Pass
Severity	Medium

Test Case ID	EIC-006
Title	toArray Method Accepts Request Parameter
Objective	Check that 'toArray' method accepts a single 'request' parameter.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on EstimateItemCollection. 2. Get the method parameters for 'toArray'. 3. Validate the parameter count and name is 'request'.
Test Data	- Method: toArray
Expected Result	- Method accepts single parameter named 'request'.
Actual Result	Parameter ‘request’ was present and correct.
Status	Pass
Severity	Medium

Test Case ID	EIC-007
Title	Handles Empty EstimateItemCollection
Objective	Verify that EstimateItemCollection can handle and return an empty array for an empty collection.
Preconditions	- Application running
Test Steps	1. Create an empty Collection. 2. Pass to EstimateItemCollection. 3. Call toArray with a Request. 4. Assert that returned array is empty.
Test Data	- Collection: []
Expected Result	- Returned value is an empty array.

Actual Result	Empty array was returned as expected.
Status	Pass
Severity	Medium

Test Case ID	EIC-008
Title	Transformation of Single Estimate Item Resource
Objective	Validate transformation of a single EstimateItemResource within the collection.
Preconditions	- Application running - EstimateItemResource and EstimateItemCollection classes available
Test Steps	1. Create a dummy estimate item (id: 1, name: 'Item 1', price: 1000). 2. Wrap item in EstimateItemResource. 3. Add to EstimateItemCollection. 4. Call toArray with a Request. 5. Assert returned array has one item with correct 'id' and 'name' keys.
Test Data	- item: {id: 1, name: 'Item 1', price: 1000}
Expected Result	- Array with one element - Element has 'id': 1 and 'name': 'Item 1'
Actual Result	Array contained correct item with id and name as specified.
Status	Pass
Severity	High

Test Case ID	EIC-009
Title	Transformation of Multiple EstimateItemResource Objects
Objective	Ensure EstimateItemCollection properly transforms multiple EstimateItemResource objects.
Preconditions	- Application running
Test Steps	1. Create two dummy items: {id: 1, name: 'Item 1', price: 1000}, {id: 2, name: 'Item 2', price: 2000}. 2. Wrap each item by EstimateItemResource. 3. Add both resources to EstimateItemCollection. 4. Call toArray with a Request. 5. Assert the resulting array has two elements, with correct ids (1, 2).
Test Data	- items: {id: 1, ...}, {id: 2, ...}
Expected Result	- Array with count 2 - First element has 'id': 1 - Second element has 'id': 2
Actual Result	Array included both items with correct ids.
Status	Pass
Severity	High

Test Case ID	EIC-010
Title	Required Fields in Transformed Estimate Item
Objective	Assert that each transformed item includes required fields ('id', 'name', 'description', etc.).
Preconditions	- Application running
Test Steps	1. Create a dummy item (id: 1, name: 'Item 1', price: 1000). 2. Wrap item by EstimateItemResource. 3. Insert resource in EstimateItemCollection. 4. Call toArray. 5. Validate that result[0] has all required keys.
Test Data	- item: as above

Expected Result	- Item contains keys: 'id', 'name', 'description', 'discount_type', 'quantity', 'price', 'total', 'tax'
Actual Result	All required fields were present in the transformed item.
Status	Pass
Severity	High

Test Case ID	EIC-011
Title	Handles Large Collection of Estimate Items
Objective	Validate that EstimateItemCollection correctly transforms a large number of items.
Preconditions	- Application running
Test Steps	1. Create 50 dummy items with ids 1..50 and names 'Item N'. 2. Wrap each in EstimateItemResource. 3. Add all to EstimateItemCollection. 4. Call toArray. 5. Assert array length is 50. 6. Check first item's id is 1 and last item's id is 50.
Test Data	- items: ids 1..50, name: 'Item N', price: N * 100
Expected Result	- Array has 50 elements - First element id: 1 - Last element id: 50
Actual Result	All 50 items present and id boundaries correct.
Status	Pass
Severity	High

Test Case ID	EIC-012
Title	Delegation from toArray to Parent ResourceCollection
Objective	Confirm that toArray function in EstimateItemCollection delegates to the parent ResourceCollection.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to locate 'toArray' in EstimateItemCollection. 2. Check that function body contains 'parent::toArray'.
Test Data	- Class source code
Expected Result	- 'parent::toArray' is found in method implementation.
Actual Result	Delegation to parent::toArray confirmed.
Status	Pass
Severity	Medium

Test Case ID	EIC-013
Title	Parent Class of EstimateItemCollection
Objective	Ensure EstimateItemCollection parent class is ResourceCollection.
Preconditions	- Application running
Test Steps	1. Reflect EstimateItemCollection. 2. Get parent class name using getParentClass(). 3. Assert parent is ResourceCollection.
Test Data	- Class: EstimateItemCollection
Expected Result	- Parent class is ResourceCollection.
Actual Result	Parent was verified as ResourceCollection.
Status	Pass

Severity	Medium

Test Case ID	EIC-014
Title	Multiple Instances Creation
Objective	Verify multiple independent EstimateItemCollection instances can coexist.
Preconditions	- Application running
Test Steps	1. Create two EstimateItemCollection objects. 2. Assert that both are instances of EstimateItemCollection. 3. Ensure they are not the same object.
Test Data	- Instances: 2
Expected Result	- Both are instances - Both are distinct
Actual Result	Instances created successfully and are independent.
Status	Pass
Severity	Low

Test Case ID	EIC-015
Title	Cloneability of EstimateItemCollection
Objective	Confirm EstimateItemCollection can be cloned.
Preconditions	- Application running
Test Steps	1. Instantiate EstimateItemCollection. 2. Clone the object. 3. Check if clone is a new instance and distinct from original.
Test Data	- Collection: []
Expected Result	- Clone is EstimateItemCollection - Clone != original
Actual Result	Cloned object is a valid, separate instance.
Status	Pass
Severity	Low

Test Case ID	EIC-016
Title	EstimateItemCollection in Type Hint Usage
Objective	Ensure EstimateItemCollection can be used as a function parameter type hint.
Preconditions	- Application running
Test Steps	1. Define a test function that type-hints EstimateItemCollection. 2. Instantiate EstimateItemCollection. 3. Call the function with the instance. 4. Assert returned value matches instance passed in.
Test Data	- Collection: []
Expected Result	- Function accepts and returns EstimateItemCollection successfully.
Actual Result	Instance accepted in type-hint context and returned as expected.
Status	Pass
Severity	Low

Test Case ID	EIC-017
Title	Non-Abstract Status of EstimateItemCollection
Objective	Validate EstimateItemCollection is not declared abstract.

Preconditions	- Application running
Test Steps	1. Use ReflectionClass on EstimateItemCollection. 2. Check isAbstract property.
Test Data	- Class: EstimateItemCollection
Expected Result	- Class is not abstract.
Actual Result	Class is not abstract as expected.
Status	Pass
Severity	Low

Test Case ID	EIC-018
Title	Non-Final Status of EstimateItemCollection
Objective	Confirm EstimateItemCollection is not declared final.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on EstimateItemCollection. 2. Check isFinal property.
Test Data	- Class: EstimateItemCollection
Expected Result	- Class is not final.
Actual Result	Class is not final.
Status	Pass
Severity	Low

Test Case ID	EIC-019
Title	EstimateItemCollection is Not Interface
Objective	Ensure EstimateItemCollection is not defined as an interface.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to check for isInterface.
Test Data	- Class: EstimateItemCollection
Expected Result	- Class is not an interface.
Actual Result	Confirmed not an interface.
Status	Pass
Severity	Low

Test Case ID	EIC-020
Title	EstimateItemCollection is Not a Trait
Objective	Ensure EstimateItemCollection is not a trait.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass; check isTrait.
Test Data	- Class: EstimateItemCollection
Expected Result	- Class is not trait.
Actual Result	Class is not a trait.
Status	Pass
Severity	Low

Test Case ID	EIC-021
Title	EstimateItemCollection Class Loaded

Objective	Validate that EstimateItemCollection class is loaded in the application.
Preconditions	- Application running
Test Steps	1. Use class_exists on EstimateItemCollection.
Test Data	- Class: EstimateItemCollection
Expected Result	- class_exists returns true.
Actual Result	Class was loaded.
Status	Pass
Severity	Medium

Test Case ID	EIC-022
Title	Import of ResourceCollection in Source File
Objective	Confirm that EstimateItemCollection imports ResourceCollection.
Preconditions	- Application running
Test Steps	1. Get file contents of EstimateItemCollection. 2. Check for 'use Illuminate\Http\Resources\Json\ResourceCollection'.
Test Data	- File contents
Expected Result	- ResourceCollection import statement is present.
Actual Result	Import statement was found.
Status	Pass
Severity	Medium

Test Case ID	EIC-023
Title	toArray Method is Not Static
Objective	Verify that the 'toArray' method in EstimateItemCollection is not static.
Preconditions	- Application running
Test Steps	1. Use Reflection to get 'toArray' method definition. 2. Assert method is not static.
Test Data	- Method: toArray
Expected Result	- Method is not static.
Actual Result	Method was not static.
Status	Pass
Severity	Medium

Test Case ID	EIC-024
Title	toArray Method is Not Abstract
Objective	Verify that the 'toArray' method in EstimateItemCollection is not abstract.
Preconditions	- Application running
Test Steps	1. Use Reflection to get 'toArray' method definition. 2. Check isAbstract property.
Test Data	- Method: toArray
Expected Result	- Method is not abstract.
Actual Result	Method was not abstract.
Status	Pass
Severity	Medium

Test Case ID	EIC-025
Title	Preserves Item Data Integrity
Objective	Confirm that EstimateItemCollection accurately preserves data for transformed items.
Preconditions	- Application running
Test Steps	1. Create a dummy item (id: 42, name: 'Special Item', price: 5000). 2. Wrap in EstimateItemResource. 3. Add to EstimateItemCollection. 4. Call toArray. 5. Assert result has correct id, name, price, and total.
Test Data	- item: {id: 42, name: 'Special Item', price: 5000}
Expected Result	- result[0]['id']: 42 - result[0]['name']: 'Special Item' - result[0]['price']: 5000 - result[0]['total']: 5000
Actual Result	All values matched item data.
Status	Pass
Severity	High

Test Case ID	EIC-026
Title	Handles Different Discount Types
Objective	Ensure EstimateItemCollection transformation respects various discount types.
Preconditions	- Application running
Test Steps	1. Create item1 (id: 1) with discount_type 'fixed'. 2. Create item2 (id: 2) with discount_type 'percentage'. 3. Wrap both in EstimateItemResource and add to EstimateItemCollection. 4. Call toArray. 5. Assert result[0]['discount_type'] == 'fixed' 6. Assert result[1]['discount_type'] == 'percentage'
Test Data	- items: {id: 1, discount_type: 'fixed'}, {id: 2, discount_type: 'percentage'}
Expected Result	- result[0]['discount_type']: 'fixed' - result[1]['discount_type']: 'percentage'
Actual Result	Each item’s discount_type field was correctly set.
Status	Pass
Severity	High

Test Case ID	EIC-027
Title	Handles Different Quantities per Item
Objective	Confirm EstimateItemCollection transforms items with varying quantities.
Preconditions	- Application running
Test Steps	1. Create item1 (id: 1) with quantity: 1.0 2. Create item2 (id: 2) with quantity: 2.5 3. Wrap both in EstimateItemResource and add to EstimateItemCollection. 4. Call toArray. 5. Assert result[0]['quantity'] == 1.0 6. Assert result[1]['quantity'] == 2.5
Test Data	- items: {id: 1, quantity: 1.0}, {id: 2, quantity: 2.5}
Expected Result	- result[0]['quantity']: 1.0 - result[1]['quantity']: 2.5
Actual Result	Quantities are transformed and preserved as specified.

Status	Pass
Severity	High

Test Case ID	EIC-028
Title	File Size Constraint for EstimateItemCollection
Objective	Validate the source file size for simplicity and conciseness.
Preconditions	- Application running
Test Steps	1. Obtain file contents of EstimateItemCollection using Reflection. 2. Calculate file byte size. 3. Ensure size is below 1000 bytes.
Test Data	- File contents
Expected Result	- File size < 1000 bytes
Actual Result	File size was within constraint.
Status	Pass
Severity	Low

Test Case ID	EIC-029
Title	Minimal Line Count in Source File
Objective	Ensure EstimateItemCollection source file does not exceed line count threshold.
Preconditions	- Application running
Test Steps	1. Get file content using Reflection. 2. Count lines. 3. Assert line count < 30.
Test Data	- Source file
Expected Result	- Line count is less than 30.
Actual Result	File contained fewer than 30 lines.
Status	Pass
Severity	Low

Test Case ID	EIC-030
Title	Comprehensive Field Coverage in Transformation
Objective	Ensure all base fields of an estimate item are present in transformed output.
Preconditions	- Application running
Test Steps	1. Create a dummy item with all base fields (id: 1, name: 'Complete Item', price: 1500, ...). 2. Wrap in EstimateItemResource. 3. Add to EstimateItemCollection. 4. Call toArray. 5. Assert that result[0] has all fields: id, name, description, discount_type, quantity, unit_name, discount, discount_val, price, tax, total, item_id, estimate_id, company_id, exchange_rate, base_discount_val, base_price, base_tax, base_total.
Test Data	- item: {id: 1, name: 'Complete Item', price: 1500, all other base fields}
Expected Result	- Array element includes all base fields.
Actual Result	Output contained all specified fields.
Status	Pass
Severity	High

File: EstimateItemResource-Test.txt

Test Case ID	EIR-001
Title	Instantiating EstimateItemResource
Objective	Verify that the EstimateItemResource class can be properly instantiated.
Preconditions	- Application running - EstimateItemResource and JsonResource classes autoloaded - Valid Estimate item object available
Test Steps	1. Create an Estimate item object with id=1, name="Item 1", price=1000. 2. Instantiate a new EstimateItemResource with the item. 3. Check if the instantiated object is an instance of EstimateItemResource.
Test Data	- Input: id=1, name="Item 1", price=1000 - Expected class: EstimateItemResource
Expected Result	- The object is an instance of EstimateItemResource.
Actual Result	The EstimateItemResource instance was created successfully and matched the expected class.
Status	Pass
Severity	Medium

Test Case ID	EIR-002
Title	EstimateItemResource Extends JsonResource
Objective	Confirm EstimateItemResource inherits from JsonResource.
Preconditions	- Application running - EstimateItemResource and JsonResource classes available
Test Steps	1. Create an Estimate item object. 2. Instantiate EstimateItemResource with the item. 3. Verify the object is an instance of JsonResource.
Test Data	- Input: id=1, name="Item 1", price=1000
Expected Result	- EstimateItemResource instance is also an instance of JsonResource.
Actual Result	The EstimateItemResource class correctly extended JsonResource.
Status	Pass
Severity	Medium

Test Case ID	EIR-003
Title	Namespace of EstimateItemResource
Objective	Ensure EstimateItemResource is in the correct namespace.
Preconditions	- Application running - EstimateItemResource loaded
Test Steps	1. Use Reflection to inspect EstimateItemResource. 2. Get the namespace of the class.
Test Data	- Class: EstimateItemResource
Expected Result	- Namespace is 'Crater\Http\Resources'.
Actual Result	Reflection returned 'Crater\Http\Resources' namespace.
Status	Pass
Severity	Low

Test Case ID	EIR-004
Title	Presence of toArray Method

Objective	Confirm that EstimateItemResource has a public 'toArray' method.
Preconditions	- Application running - EstimateItemResource loaded
Test Steps	1. Create an EstimateItemResource instance. 2. Check if the 'toArray' method exists.
Test Data	- Input: Any valid Estimate item
Expected Result	- 'toArray' method exists in class.
Actual Result	The 'toArray' method was found in EstimateItemResource.
Status	Pass
Severity	Medium

Test Case ID	EIR-005
Title	toArray Method Visibility
Objective	Ensure the 'toArray' method is public.
Preconditions	- Application running - EstimateItemResource loaded
Test Steps	1. Use Reflection to get 'toArray' method. 2. Verify its visibility is public.
Test Data	- Method: toArray
Expected Result	- Method is publicly accessible.
Actual Result	Reflected method showed public visibility.
Status	Pass
Severity	Medium

Test Case ID	EIR-006
Title	toArray Method Accepts Request Parameter
Objective	Ensure toArray accepts a single parameter named 'request'.
Preconditions	- Application running - EstimateItemResource loaded
Test Steps	1. Use Reflection to query 'toArray' method signature. 2. Check parameters count and name.
Test Data	- Method: toArray
Expected Result	- toArray has one parameter named 'request'.
Actual Result	Parameter list matched expectations.
Status	Pass
Severity	Medium

Test Case ID	EIR-007
Title	Basic Transformation of Resource by toArray
Objective	Validate that toArray transforms resource with all basic properties.
Preconditions	- Application running
Test Steps	1. Create item: id=1, name="Test Item", price=1500. 2. Pass item to EstimateItemResource. 3. Call toArray with a Request. 4. Check if result is array with keys and values for 'id', 'name', and 'price'.
Test Data	- Input: id=1, name="Test Item", price=1500

Expected Result	- Result is array - 'id' is 1 - 'name' is "Test Item" - 'price' is 1500
Actual Result	All expected keys and values present with correct types.
Status	Pass
Severity	High

Test Case ID	EIR-008
Title	Inclusion of All Required Fields in toArray
Objective	Ensure all necessary fields are present in the transformation output.
Preconditions	- Application running
Test Steps	1. Create item: id=1, name="Item", price=1000. 2. Transform via toArray. 3. Check array for expected keys.
Test Data	- Input: id=1, name="Item", price=1000
Expected Result	- Array contains all required fields: 'id', 'name', 'description', 'discount_type', 'quantity', 'unit_name', 'discount', 'discount_val', 'price', 'tax', 'total', 'item_id', 'estimate_id', 'company_id', 'exchange_rate', 'base_discount_val', 'base_price', 'base_tax', 'base_total', 'taxes', 'fields'.
Actual Result	All required fields were present in array output.
Status	Pass
Severity	High

Test Case ID	EIR-009
Title	Correct Data Types Returned by toArray
Objective	Verify that each output field has the appropriate data type.
Preconditions	- Application running
Test Steps	1. Create item: id=42, name="Typed Item", price=2500. 2. Transform via toArray. 3. Assert types: - 'id' is integer - 'name' is string - 'price' is integer - 'quantity' is float - 'discount' is float
Test Data	- Input: id=42, name="Typed Item", price=2500
Expected Result	- All fields have correct types
Actual Result	Field types matched expectations.
Status	Pass
Severity	High

Test Case ID	EIR-010
Title	Handling of Null Values in toArray
Objective	Ensure toArray can process items returning null for all properties.
Preconditions	- Application running
Test Steps	1. Create a stub item where all __get accesses return null. 2. Transform via toArray. 3. Assert that certain keys (id, name, price) are null.

Test Data	- Input: Custom item object returning null
Expected Result	- Keys 'id', 'name', 'price' are null in output
Actual Result	Null values handled gracefully as required.
Status	Pass
Severity	High

Test Case ID	EIR-011
Title	Data Integrity Preservation
Objective	Confirm that item data is preserved after transformation.
Preconditions	- Application running
Test Steps	1. Create item: id=99, name="Special Item", price=7500. 2. Transform via toArray. 3. Assert output matches input values for id, name, price, total.
Test Data	- Input: id=99, name="Special Item", price=7500
Expected Result	- Output 'id' is 99, 'name' is "Special Item", 'price' and 'total' are 7500
Actual Result	Data integrity preserved for all asserted fields.
Status	Pass
Severity	High

Test Case ID	EIR-012
Title	Description Field Inclusion
Objective	Ensure 'description' field is present and a string.
Preconditions	- Application running
Test Steps	1. Create item with description: id=1, name="Item with Description", price=1000. 2. Transform via toArray. 3. Verify presence and type of 'description'.
Test Data	- Input: id=1, name="Item with Description", price=1000
Expected Result	- 'description' field exists and is a string
Actual Result	Field present and type correct.
Status	Pass
Severity	Medium

Test Case ID	EIR-013
Title	Discount Fields Presence
Objective	Validate inclusion of discount field information.
Preconditions	- Application running
Test Steps	1. Create item: id=1, name="Item", price=1000. 2. Transform via toArray. 3. Assert keys 'discount_type', 'discount', 'discount_val' exist.
Test Data	- Input: id=1, name="Item", price=1000
Expected Result	- All discount-related fields exist
Actual Result	All discount fields were present
Status	Pass
Severity	Medium

Test Case ID	EIR-014

Title	Inclusion of Base Currency Fields
Objective	Ensure base currency related fields are present in output.
Preconditions	- Application running
Test Steps	1. Create item: id=1, name="Item", price=1000. 2. Transform via toArray. 3. Check for fields: 'base_price', 'base_tax', 'base_total', 'base_discount_val'.
Test Data	- Input: id=1, name="Item", price=1000
Expected Result	- All base currency fields present
Actual Result	All expected base fields found
Status	Pass
Severity	Medium

Test Case ID	EIR-015
Title	Exchange Rate Field Inclusion
Objective	Ensure estimate item output includes 'exchange_rate' field.
Preconditions	- Application running
Test Steps	1. Create item: id=1, name="Item", price=1000. 2. Transform via toArray. 3. Validate presence of 'exchange_rate'.
Test Data	- Input: id=1, name="Item", price=1000
Expected Result	- Field 'exchange_rate' present
Actual Result	Field present as expected
Status	Pass
Severity	Medium

Test Case ID	EIR-016
Title	Inclusion of Item ID and Estimate ID
Objective	Confirm output contains correct item_id, estimate_id, company_id values.
Preconditions	- Application running
Test Steps	1. Create item: id=1, name="Item", price=1000. 2. Transform via toArray. 3. Check for presence of 'item_id', 'estimate_id', 'company_id'.
Test Data	- Input: id=1, name="Item", price=1000
Expected Result	- All ID fields present
Actual Result	item_id, estimate_id, company_id were all included
Status	Pass
Severity	Medium

Test Case ID	EIR-017
Title	Multiple Instances of EstimateItemResource
Objective	Verify that multiple instances can be created and are unique.
Preconditions	- Application running
Test Steps	1. Create two distinct item objects. 2. Instantiate two EstimateItemResource objects. 3. Assert object uniqueness and correct type.
Test Data	- Item 1: id=1, name="Item 1", price=1000 - Item 2: id=2, name="Item 2", price=2000

Expected Result	- Both objects created and not equal, type correct
Actual Result	Multiple instances created and unique
Status	Pass
Severity	Low

Test Case ID	EIR-018
Title	Cloning EstimateItemResource
Objective	Check that EstimateItemResource objects can be cloned.
Preconditions	- Application running
Test Steps	1. Create item and EstimateItemResource instance. 2. Clone the resource object. 3. Ensure clone is distinct and correct type.
Test Data	- id=1, name="Item", price=1000
Expected Result	- Clone is new instance, type EstimateItemResource
Actual Result	Clone created successfully and was unique
Status	Pass
Severity	Low

Test Case ID	EIR-019
Title	Use of EstimateItemResource in Type Hints
Objective	Validate EstimateItemResource can be used as a function argument type.
Preconditions	- Application running
Test Steps	1. Define a function accepting EstimateItemResource type. 2. Create instance and call the function. 3. Verify argument passes as expected.
Test Data	- id=1, name="Item", price=1000
Expected Result	- Function receives resource and returns correctly
Actual Result	Type hint worked successfully
Status	Pass
Severity	Low

Test Case ID	EIR-020
Title	EstimateItemResource is Not Abstract
Objective	Confirm EstimateItemResource is not declared abstract.
Preconditions	- Application running
Test Steps	1. Reflect on EstimateItemResource class. 2. Assert 'isAbstract' returns false.
Test Data	- Class: EstimateItemResource
Expected Result	- Class is not abstract
Actual Result	EstimateItemResource is concrete
Status	Pass
Severity	Low

Test Case ID	EIR-021
Title	EstimateItemResource is Not Final
Objective	Ensure EstimateItemResource can be extended.

Preconditions	- Application running
Test Steps	1. Reflect on EstimateItemResource. 2. Verify 'isFinal' is false.
Test Data	- Class: EstimateItemResource
Expected Result	- Class is not final
Actual Result	isFinal returned false
Status	Pass
Severity	Low

Test Case ID	EIR-022
Title	EstimateItemResource is Not an Interface
Objective	Confirm EstimateItemResource is a class, not an interface.
Preconditions	- Application running
Test Steps	1. Reflect on EstimateItemResource. 2. Assert 'isInterface' returns false.
Test Data	- Class: EstimateItemResource
Expected Result	- Not an interface
Actual Result	Correctly identified as class
Status	Pass
Severity	Low

Test Case ID	EIR-023
Title	EstimateItemResource is Not a Trait
Objective	Confirm EstimateItemResource is not a trait.
Preconditions	- Application running
Test Steps	1. Reflect on EstimateItemResource. 2. Assert 'isTrait' is false.
Test Data	- Class: EstimateItemResource
Expected Result	- Not a trait
Actual Result	isTrait returned false
Status	Pass
Severity	Low

Test Case ID	EIR-024
Title	EstimateItemResource Class is Loaded
Objective	Validate class_exists works for EstimateItemResource.
Preconditions	- Application running
Test Steps	1. Call class_exists for EstimateItemResource. 2. Assert result is true.
Test Data	- Class name: EstimateItemResource
Expected Result	- class_exists returns true
Actual Result	Returned true
Status	Pass
Severity	Low

Test Case ID	EIR-025

Title	toArray Method is Not Static
Objective	Ensure toArray is not declared as static.
Preconditions	- Application running
Test Steps	1. Reflect on toArray method. 2. Assert isStatic is false.
Test Data	- Method: toArray
Expected Result	- Non-static method
Actual Result	isStatic returned false
Status	Pass
Severity	Medium

Test Case ID	EIR-026
Title	toArray Method is Not Abstract
Objective	Confirm toArray provides implementation and is not abstract.
Preconditions	- Application running
Test Steps	1. Reflect on toArray method. 2. Assert isAbstract is false.
Test Data	- Method: toArray
Expected Result	- isAbstract returns false
Actual Result	Method is concrete
Status	Pass
Severity	Medium

Test Case ID	EIR-027
Title	Array Output Structure (21 Keys)
Objective	Ensure toArray result has exactly 21 keys.
Preconditions	- Application running
Test Steps	1. Create item. 2. Transform via toArray. 3. Count keys on array.
Test Data	- id=1, name="Item", price=1000
Expected Result	- Output array has 21 keys
Actual Result	Returned array had 21 keys
Status	Pass
Severity	High

Test Case ID	EIR-028
Title	toArray Result is a Valid Array
Objective	Confirm toArray output is an array and is valid.
Preconditions	- Application running
Test Steps	1. Create item. 2. Transform via toArray. 3. Check type and assert array validity.
Test Data	- id=1, name="Item", price=1000
Expected Result	- Output is array and passes is_array()
Actual Result	Output was a valid array

Status	Pass
Severity	High

Test Case ID	EIR-029
Title	Discount Type Handling
Objective	Verify handling and value of 'discount_type' field.
Preconditions	- Application running
Test Steps	1. Create item. 2. Transform via toArray. 3. Assert 'discount_type' is 'fixed'.
Test Data	- id=1, name="Item 1", price=1000
Expected Result	- 'discount_type' equals 'fixed'
Actual Result	discount_type returned 'fixed'
Status	Pass
Severity	Medium

Test Case ID	EIR-030
Title	Quantity Field Handling
Objective	Confirm correct value for 'quantity' in output.
Preconditions	- Application running
Test Steps	1. Create item. 2. Transform via toArray. 3. Assert 'quantity' is 1.0.
Test Data	- id=1, name="Item", price=1000
Expected Result	- 'quantity' equals 1.0
Actual Result	quantity was 1.0
Status	Pass
Severity	Medium

Test Case ID	EIR-031
Title	Unit Name Field Handling
Objective	Confirm correct value for 'unit_name' in output.
Preconditions	- Application running
Test Steps	1. Create item. 2. Transform via toArray. 3. Assert 'unit_name' is 'piece'.
Test Data	- id=1, name="Item", price=1000
Expected Result	- 'unit_name' equals 'piece'
Actual Result	unit_name was 'piece'
Status	Pass
Severity	Medium

Test Case ID	EIR-032
Title	EstimateItemResource File Structure Validation
Objective	Check presence of class and public method declarations in file contents.
Preconditions	- Application running

Test Steps	1. Reflect on EstimateItemResource class. 2. Read file via getFileName(). 3. Assert file contains 'class EstimateItemResource extends JsonResource' and 'public function toArray'.
Test Data	- N/A
Expected Result	- Declarations found in file content
Actual Result	Both strings found in file
Status	Pass
Severity	Low

Test Case ID	EIR-033
Title	Use of when() for Conditional Relationships
Objective	Verify use of $this->when in EstimateItemResource file source.
Preconditions	- Application running
Test Steps	1. Fetch EstimateItemResource source code. 2. Assert file contains '$this->when'.
Test Data	- N/A
Expected Result	- '$this->when' present
Actual Result	String present in file as expected
Status	Pass
Severity	Low

Test Case ID	EIR-034
Title	Compact File Implementation Size
Objective	Ensure source file size is less than 3000 bytes.
Preconditions	- Application running
Test Steps	1. Read file content via Reflection. 2. Check file size is under 3000 bytes.
Test Data	- N/A
Expected Result	- File size < 3000 bytes
Actual Result	File size met compact implementation criteria
Status	Pass
Severity	Low

Test Case ID	EIR-035
Title	Numeric Fields Are Present
Objective	Confirm presence of all expected numeric fields in output.
Preconditions	- Application running
Test Steps	1. Create item. 2. Transform via toArray. 3. Check array has keys: ['id', 'quantity', 'discount', 'discount_val', 'price', 'tax', 'total', 'item_id', 'estimate_id', 'company_id', 'exchange_rate', 'base_discount_val', 'base_price', 'base_tax', 'base_total'].
Test Data	- id=1, name="Item", price=1000
Expected Result	- All listed numeric fields present in array
Actual Result	All fields found
Status	Pass

Severity	High

Test Case ID	EIR-036
Title	String Fields Are Present
Objective	Confirm all expected string fields are in output.
Preconditions	- Application running
Test Steps	1. Create item. 2. Transform via toArray. 3. Check for keys: ['name', 'description', 'discount_type', 'unit_name'].
Test Data	- id=1, name="Item", price=1000
Expected Result	- All listed string fields present
Actual Result	All string fields included
Status	Pass
Severity	Medium

Test Case ID	EIR-037
Title	Relationship Fields Presence
Objective	Validate that 'taxes' and 'fields' fields are included in output.
Preconditions	- Application running
Test Steps	1. Create item. 2. Transform via toArray. 3. Check for keys 'taxes' and 'fields'.
Test Data	- id=1, name="Item", price=1000
Expected Result	- Relationship fields present
Actual Result	Fields present as expected
Status	Pass
Severity	Medium

File: EstimatePdfController-Test.txt

Test Case ID	EPC-001
Title	Returns PDF data when preview parameter is present
Objective	Verify that the EstimatePdfController returns PDF data as an array when the 'preview' parameter is present in the request.
Preconditions	- Application running - Database seeded with valid Estimate data - Valid EstimatePdfController instance available - Mocked Request instance available - Mocked Estimate instance available
Test Steps	1. Create a mock Request that returns true when 'has' method is called with 'preview'. 2. Set up the Estimate mock to return a predefined PDF data array when 'getPDFData' is called. 3. Ensure 'getGeneratedPDFOrStream' is not called on the Estimate mock. 4. Invoke the EstimatePdfController with the mocked Request and Estimate. 5. Capture the return value.
Test Data	- Request parameter: 'preview' = true - Estimate mock returns: ['pdf_data' => 'sample_pdf_json_data']
Expected Result	- The controller returns: ['pdf_data' => 'sample_pdf_json_data']
Actual Result	The controller returned ['pdf_data' => 'sample_pdf_json_data'] as expected.
Status	Pass
Severity	High

Test Case ID	EPC-002
Title	Returns generated PDF stream Response when preview parameter is absent
Objective	Verify that the EstimatePdfController returns a generated PDF stream Response when the 'preview' parameter is absent in the request.
Preconditions	- Application running - Database seeded with valid Estimate data - Valid EstimatePdfController instance available - Mocked Request instance available - Mocked Estimate instance available - Mocked Response instance available
Test Steps	1. Create a mock Request that returns false when 'has' method is called with 'preview'. 2. Set up the Estimate mock to return a mocked Response when 'getGeneratedPDFOrStream' is called with 'estimate'. 3. Ensure 'getPDFData' is not called on the Estimate mock. 4. Invoke the EstimatePdfController with the mocked Request and Estimate. 5. Capture the return value.
Test Data	- Request parameter: 'preview' = false - Estimate mock returns: Mocked Response object (header methods may be called)
Expected Result	- The controller returns the mocked Response instance representing a generated PDF stream.
Actual Result	The controller returned the mocked Response instance as expected.
Status	Pass
Severity	High

File: EstimateResource-Test.txt

Test Case ID	ER-001
Title	Instantiate EstimateResource object
Objective	Verify that an EstimateResource instance can be created without error.
Preconditions	- Application running - Database seeded - Estimate object available with valid properties
Test Steps	1. Create an estimate object with id=1, estimate_number='EST-001', total=1000. 2. Instantiate a new EstimateResource object with the created estimate. 3. Assert that the resource is an instance of EstimateResource.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- The resource is a valid instance of EstimateResource.
Actual Result	Resource was successfully instantiated and verified as EstimateResource.
Status	Pass
Severity	Medium

Test Case ID	ER-002
Title	Validate EstimateResource inheritance from JsonResource
Objective	Ensure that EstimateResource extends the JsonResource class.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create an estimate object. 2. Instantiate EstimateResource with the estimate. 3. Assert that the resource is an instance of JsonResource.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- EstimateResource is an instance of Illuminate\Http\Resources\Json\JsonResource.
Actual Result	EstimateResource confirmed to extend JsonResource.
Status	Pass
Severity	Medium

Test Case ID	ER-003
Title	Check EstimateResource namespace
Objective	Verify that EstimateResource exists in the correct namespace.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Use ReflectionClass on EstimateResource. 2. Retrieve and verify the class namespace.
Test Data	- None
Expected Result	- Namespace is 'Crater\Http\Resources'.
Actual Result	Namespace 'Crater\Http\Resources' is correctly applied.
Status	Pass
Severity	Low

Test Case ID	ER-004

Title	Check presence of toArray method in EstimateResource
Objective	Confirm that EstimateResource class has a toArray method.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create an estimate object. 2. Instantiate EstimateResource. 3. Check for the existence of the toArray method in the resource.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- method_exists($resource, 'toArray') returns true.
Actual Result	toArray method exists in EstimateResource.
Status	Pass
Severity	Medium

Test Case ID	ER-005
Title	Validate public visibility of toArray method
Objective	Ensure the toArray method in EstimateResource is publicly accessible.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Use ReflectionClass to inspect EstimateResource. 2. Retrieve the toArray method. 3. Check if the method is public.
Test Data	- None
Expected Result	- toArray method is public.
Actual Result	toArray method is confirmed to be public.
Status	Pass
Severity	Medium

Test Case ID	ER-006
Title	Verify toArray method parameter
Objective	Validate that the toArray method accepts a single parameter named 'request'.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Use ReflectionClass to inspect EstimateResource. 2. Retrieve toArray method parameters. 3. Assert parameter count is 1 with name 'request'.
Test Data	- None
Expected Result	- toArray method accepts one parameter named 'request'.
Actual Result	Parameter is present and named 'request'.
Status	Pass
Severity	Medium

Test Case ID	ER-007
Title	Transform resource via toArray with basic properties
Objective	Confirm toArray transforms resource correctly and includes basic keys.

Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request object. 2. Create estimate object: id=1, estimate_number='EST-001', total=1500. 3. Instantiate EstimateResource and call toArray with Request. 4. Verify returned array contains keys 'id' and 'estimate_number' matching expected values.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1500
Expected Result	- Returned array contains 'id' = 1, 'estimate_number' = 'EST-001'.
Actual Result	Both 'id' and 'estimate_number' present with correct values.
Status	Pass
Severity	High

Test Case ID	ER-008
Title	Validate presence of all required fields in toArray output
Objective	Ensure the transformed array includes all required fields.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request object. 2. Create estimate object: id=1, estimate_number='EST-001', total=1000. 3. Instantiate EstimateResource and call toArray. 4. Check for presence of all required keys in returned array.
Test Data	- Fields: id, estimate_date, expiry_date, estimate_number, status, reference_number, tax_per_item, discount_per_item, notes, discount, discount_type, discount_val, sub_total, total, tax, unique_hash, creator_id, template_name, customer_id, exchange_rate, base_discount_val, base_sub_total, base_total, base_tax, sequence_number, currency_id, formatted_expiry_date, formatted_estimate_date, estimate_pdf_url, sales_tax_type, sales_tax_address_type
Expected Result	- All listed fields are present in the array.
Actual Result	All required fields successfully present in output.
Status	Pass
Severity	High

Test Case ID	ER-009
Title	Estimate data integrity preservation
Objective	Verify that EstimateResource preserves original estimate values.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request object. 2. Create estimate object: id=42, estimate_number='EST-042', total=5000. 3. Instantiate EstimateResource and call toArray. 4. Assert that output matches input values for id, estimate_number, total, and status.
Test Data	- Estimate: id=42, estimate_number='EST-042', total=5000, status='DRAFT'
Expected Result	- 'id' = 42, 'estimate_number' = 'EST-042', 'total' = 5000, 'status' = 'DRAFT'
Actual Result	All values correctly preserved in output.
Status	Pass

Severity	High

Test Case ID	ER-010
Title	Multiple EstimateResource instances creation test
Objective	Ensure multiple EstimateResource objects can be created and are unique.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create two estimate objects with different IDs/numbers. 2. Instantiate EstimateResource for both estimates. 3. Assert both instances exist and are distinct.
Test Data	- Estimate1: id=1, estimate_number='EST-001', total=1000 - Estimate2: id=2, estimate_number='EST-002', total=2000
Expected Result	- Both instances are EstimateResource; instances are not the same.
Actual Result	Distinct EstimateResource instances successfully created.
Status	Pass
Severity	Medium

Test Case ID	ER-011
Title	EstimateResource cloning capability
Objective	Verify that EstimateResource object can be cloned and remains valid.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create estimate object. 2. Instantiate EstimateResource. 3. Clone the EstimateResource object. 4. Assert the clone is of type EstimateResource and not equal to the original.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- Clone is an instance of EstimateResource and distinct from original.
Actual Result	Cloning yielded valid, unique EstimateResource instance.
Status	Pass
Severity	Medium

Test Case ID	ER-012
Title	Use EstimateResource in function type hints
Objective	Ensure EstimateResource objects can be used as type-hinted parameters.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create estimate object. 2. Instantiate EstimateResource. 3. Pass instance to a function expecting EstimateResource type. 4. Assert returned value matches the resource.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- Passed resource matches the received parameter.
Actual Result	Type hinting with EstimateResource works as expected.
Status	Pass
Severity	Low

Test Case ID	ER-013
Title	Confirm EstimateResource is not abstract
Objective	Ensure EstimateResource class is concrete and can be instantiated.
Preconditions	- Application running - EstimateResource class available
Test Steps	1. Use ReflectionClass on EstimateResource. 2. Check if isAbstract() returns false.
Test Data	- None
Expected Result	- isAbstract() returns false.
Actual Result	EstimateResource is not abstract.
Status	Pass
Severity	Low

Test Case ID	ER-014
Title	Confirm EstimateResource is not final
Objective	Ensure EstimateResource class can be extended.
Preconditions	- Application running - EstimateResource class available
Test Steps	1. Use ReflectionClass on EstimateResource. 2. Check if isFinal() returns false.
Test Data	- None
Expected Result	- isFinal() returns false.
Actual Result	EstimateResource is not final.
Status	Pass
Severity	Low

Test Case ID	ER-015
Title	Confirm EstimateResource is not an interface
Objective	Validate that EstimateResource is a concrete class, not an interface.
Preconditions	- Application running - EstimateResource class available
Test Steps	1. Use ReflectionClass on EstimateResource. 2. Check if isInterface() returns false.
Test Data	- None
Expected Result	- isInterface() returns false.
Actual Result	EstimateResource is not an interface.
Status	Pass
Severity	Low

Test Case ID	ER-016
Title	Confirm EstimateResource is not a trait
Objective	Validate EstimateResource is a class, not a trait.
Preconditions	- Application running - EstimateResource class available
Test Steps	1. Use ReflectionClass on EstimateResource. 2. Check if isTrait() returns false.
Test Data	- None

Expected Result	- isTrait() returns false.
Actual Result	EstimateResource is not a trait.
Status	Pass
Severity	Low

Test Case ID	ER-017
Title	Confirm EstimateResource class is loaded
Objective	Ensure class is properly defined and loaded by the runtime.
Preconditions	- Application running
Test Steps	1. Use class_exists() with EstimateResource::class. 2. Verify result is true.
Test Data	- None
Expected Result	- class_exists returns true.
Actual Result	EstimateResource class successfully loaded.
Status	Pass
Severity	Medium

Test Case ID	ER-018
Title	Confirm toArray method is not static
Objective	Validate that toArray must be called on an instance, not statically.
Preconditions	- Application running - EstimateResource class available
Test Steps	1. Use ReflectionClass on EstimateResource. 2. Retrieve toArray method. 3. Check method's static status.
Test Data	- None
Expected Result	- toArray method is not static.
Actual Result	Method is instance-based (not static).
Status	Pass
Severity	Medium

Test Case ID	ER-019
Title	Confirm toArray method is not abstract
Objective	Ensure toArray is implemented and not abstract.
Preconditions	- Application running - EstimateResource class available
Test Steps	1. Use ReflectionClass on EstimateResource. 2. Retrieve toArray method. 3. Check method's abstract status.
Test Data	- None
Expected Result	- toArray method is not abstract.
Actual Result	Method is implemented (not abstract).
Status	Pass
Severity	Medium

Test Case ID	ER-020
Title	Includes estimate date fields in output

Objective	Check that toArray returns all date-related estimate fields.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. Assert output includes keys: 'estimate_date', 'expiry_date', 'formatted_estimate_date', 'formatted_expiry_date'.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- All date fields are present in array.
Actual Result	All date fields successfully present.
Status	Pass
Severity	High

Test Case ID	ER-021
Title	Includes discount fields in output
Objective	Check that discount-related fields are present in the output array.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. Assert output includes 'discount', 'discount_type', 'discount_val'.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- Discount fields are present in array.
Actual Result	Discount fields correctly included.
Status	Pass
Severity	High

Test Case ID	ER-022
Title	Includes base currency fields in output
Objective	Check that all base currency-related fields appear in array output.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. Assert output includes 'base_discount_val', 'base_sub_total', 'base_total', 'base_tax'.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- Base currency fields are present.
Actual Result	Base currency fields successfully verified.
Status	Pass
Severity	High

Test Case ID	ER-023
Title	Includes tax configuration fields in output
Objective	Confirm presence of tax configuration fields in output array.

Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. Assert array contains 'tax_per_item', 'discount_per_item', 'sales_tax_type', 'sales_tax_address_type'.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- All tax configuration fields are present.
Actual Result	Tax configuration fields present and correct.
Status	Pass
Severity	High

Test Case ID	ER-024
Title	Includes PDF URL field in output
Objective	Confirm estimate_pdf_url is available in transformed output.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. Assert array contains key 'estimate_pdf_url'.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- Output includes 'estimate_pdf_url'.
Actual Result	estimate_pdf_url present and properly formatted.
Status	Pass
Severity	Medium

Test Case ID	ER-025
Title	Includes notes field in output
Objective	Check that the notes field is included in toArray output.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. Assert array contains key 'notes'.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- Notes field present.
Actual Result	Notes field correctly included.
Status	Pass
Severity	Medium

Test Case ID	ER-026
Title	EstimateResource file structure validation
Objective	Confirm EstimateResource file contains the expected class and method definitions.
Preconditions	- Application running - EstimateResource class available

Test Steps	1. Use ReflectionClass on EstimateResource. 2. Retrieve file contents. 3. Assert file contains 'class EstimateResource extends JsonResource' and 'public function toArray'.
Test Data	- None
Expected Result	- File contains expected class and toArray method declaration.
Actual Result	File structure matches requirements.
Status	Pass
Severity	Low

Test Case ID	ER-027
Title	Uses when() for conditional relationships
Objective	Validate EstimateResource uses when() for conditionally including relationships.
Preconditions	- Application running - EstimateResource class available
Test Steps	1. Use ReflectionClass on EstimateResource. 2. Retrieve file contents. 3. Search for occurrences of '$this->when'.
Test Data	- None
Expected Result	- '$this->when' found in file content.
Actual Result	Conditional inclusion with when() verified.
Status	Pass
Severity	Medium

Test Case ID	ER-028
Title	Comprehensive implementation size in EstimateResource
Objective	Confirm EstimateResource file is substantial, suggesting comprehensive implementation.
Preconditions	- Application running - EstimateResource class available
Test Steps	1. Use ReflectionClass to locate EstimateResource file. 2. Retrieve and measure file content size. 3. Assert file size is greater than 2000 bytes.
Test Data	- None
Expected Result	- File size exceeds 2000 bytes.
Actual Result	Implementation size is sufficient for comprehensive resource.
Status	Pass
Severity	Medium

Test Case ID	ER-029
Title	All numeric fields present in toArray output
Objective	Ensure all expected numeric fields are present in output array.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. For each listed numeric field, assert key presence.

Test Data	- Numeric fields: id, discount, discount_val, sub_total, total, tax, creator_id, customer_id, exchange_rate, base_discount_val, base_sub_total, base_total, base_tax, sequence_number, currency_id
Expected Result	- All numeric fields present.
Actual Result	All numeric fields are present in output.
Status	Pass
Severity	High

Test Case ID	ER-030
Title	All string fields present in toArray output
Objective	Ensure all expected string fields are present in output array.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. For each listed string field, assert key presence.
Test Data	- String fields: estimate_number, status, reference_number, discount_type, unique_hash, template_name
Expected Result	- All string fields present.
Actual Result	All string fields are present in output.
Status	Pass
Severity	High

Test Case ID	ER-031
Title	Output is valid array structure with sufficient count
Objective	Confirm toArray returns an array with significant field count.
Preconditions	- Application running - Database seeded - EstimateResource class available
Test Steps	1. Create Request and estimate object. 2. Instantiate EstimateResource, call toArray. 3. Assert output is an array, type is array, and array count exceeds 20.
Test Data	- Estimate: id=1, estimate_number='EST-001', total=1000
Expected Result	- Output is an array type. - Output array has more than 20 elements.
Actual Result	Output is a valid array of sufficient length.
Status	Pass
Severity	High

File: EstimateTemplatesController-Test.txt

Test Case ID	ETC-001
Title	Controller Instantiation
Objective	Verify EstimateTemplatesController can be instantiated.
Preconditions	- Application running - Required classes are available - No constructor dependencies fail
Test Steps	1. Attempt to instantiate EstimateTemplatesController. 2. Verify the resulting object is of type EstimateTemplatesController.
Test Data	- Instantiation: new EstimateTemplatesController() - Expected value: instance of EstimateTemplatesController
Expected Result	- Controller instance is successfully created and is an object of EstimateTemplatesController.
Actual Result	Instance was successfully created as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-002
Title	Controller Extends Base Controller
Objective	Ensure EstimateTemplatesController extends the base Controller class.
Preconditions	- Application running - Base Controller class available
Test Steps	1. Instantiate EstimateTemplatesController. 2. Check if the instance is also an instance of Crater\Http\Controllers\Controller.
Test Data	- Instantiation: new EstimateTemplatesController() - Expected value: instance of Crater\Http\Controllers\Controller
Expected Result	- Controller instance extends the Controller base class.
Actual Result	Confirmed that EstimateTemplatesController extends the Controller class.
Status	Pass
Severity	Medium

Test Case ID	ETC-003
Title	Verify Controller Namespace
Objective	Ensure EstimateTemplatesController is in the correct namespace.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Retrieve the namespace. 3. Compare to expected namespace string.
Test Data	- Class namespace: Crater\Http\Controllers\V1\Admin\Estimate
Expected Result	- Namespace name matches 'Crater\Http\Controllers\V1\Admin\Estimate'.
Actual Result	Namespace is correct as expected.
Status	Pass
Severity	Low

Test Case ID	ETC-004

Title	Controller Is Not Abstract
Objective	Ensure that EstimateTemplatesController is not an abstract class.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Check if isAbstract() returns false.
Test Data	- isAbstract property: expected to be false
Expected Result	- Controller class is concrete, not abstract.
Actual Result	Controller is not abstract as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-005
Title	Controller Is Instantiable
Objective	Ensure EstimateTemplatesController can be instantiated.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Check if isInstantiable() returns true.
Test Data	- isInstantiable property: expected to be true
Expected Result	- Controller class is instantiable.
Actual Result	Controller is instantiable as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-006
Title	Controller Has __invoke Method
Objective	Verify EstimateTemplatesController contains the __invoke method.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Instantiate EstimateTemplatesController. 2. Check method_exists for '__invoke'.
Test Data	- Method name: __invoke
Expected Result	- __invoke method exists in the controller.
Actual Result	__invoke method found as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-007
Title	__invoke Method Accessibility
Objective	Verify that __invoke method is public.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Get the __invoke method. 3. Check isPublic() returns true.
Test Data	- Method: __invoke - Visibility: public

Expected Result	- __invoke method is public.
Actual Result	__invoke method is public as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-008
Title	__invoke Method Parameter Verification
Objective	Ensure __invoke method accepts a single Request parameter.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Get the __invoke method parameters. 3. Verify there is one parameter named 'request'. 4. Verify parameter type includes 'Request'.
Test Data	- Parameter name: request - Parameter type: Request
Expected Result	- __invoke method accepts one parameter named 'request' of type Request.
Actual Result	Parameter verification successful as expected.
Status	Pass
Severity	High

Test Case ID	ETC-009
Title	__invoke Method Is Not Static
Objective	Ensure __invoke method is not static.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Get the __invoke method. 3. Check isStatic() returns false.
Test Data	- Method: __invoke - isStatic: false
Expected Result	- Method is not static.
Actual Result	__invoke method is not static as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-010
Title	__invoke Method Is Not Abstract
Objective	Ensure __invoke method is not abstract.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Get the __invoke method. 3. Check isAbstract() returns false.
Test Data	- Method: __invoke - isAbstract: false
Expected Result	- Method is not abstract.
Actual Result	__invoke method is not abstract as expected.
Status	Pass

Severity	Medium

Test Case ID	ETC-011
Title	Controller Is Invokable
Objective	Verify that the controller instance is invokable.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Instantiate EstimateTemplatesController. 2. Check if is_callable on the instance returns true.
Test Data	- is_callable($controller): expected true
Expected Result	- Controller instance is invokable.
Actual Result	Controller is invokable as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-012
Title	Controller Callable As Function
Objective	Ensure controller can be called as a function.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Instantiate EstimateTemplatesController. 2. Create an instance of Request. 3. Verify is_callable([$controller, '__invoke']) returns true.
Test Data	- Instance: EstimateTemplatesController - Callable: [$controller, '__invoke']
Expected Result	- Controller can be called as a function.
Actual Result	Callable check successful as expected.
Status	Pass
Severity	High

Test Case ID	ETC-013
Title	Multiple Instances Creation
Objective	Ensure multiple distinct EstimateTemplatesController instances can be created.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Instantiate two EstimateTemplatesController objects. 2. Confirm both are instances of EstimateTemplatesController. 3. Verify they are distinct objects.
Test Data	- Instances: $controller1, $controller2
Expected Result	- Both instances are of correct type and distinct.
Actual Result	Multiple instances created and distinct as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-014
Title	Controller Cloning
Objective	Ensure EstimateTemplatesController can be cloned correctly.

Preconditions	- Application running - Controller class file loaded
Test Steps	1. Instantiate EstimateTemplatesController. 2. Clone the instance. 3. Check cloned object is of same type and distinct from original.
Test Data	- Original: $controller - Cloned: $clone
Expected Result	- Cloned controller is a distinct, correct instance.
Actual Result	Cloning successful; distinct instance as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-015
Title	Controller Can Be Used in Type Hints
Objective	Verify EstimateTemplatesController can be used in type hints.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a function that accepts EstimateTemplatesController type hint. 2. Pass controller instance to function. 3. Verify returned value matches original instance.
Test Data	- Function argument: EstimateTemplatesController
Expected Result	- Function accepts the controller, returns the same instance.
Actual Result	Type hint accepted and matched as expected.
Status	Pass
Severity	Low

Test Case ID	ETC-016
Title	Controller Is Not Final
Objective	Ensure EstimateTemplatesController is not final.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Check isFinal returns false.
Test Data	- isFinal: false
Expected Result	- Controller class is not final.
Actual Result	Controller is not final as expected.
Status	Pass
Severity	Low

Test Case ID	ETC-017
Title	Controller Is Not an Interface
Objective	Ensure EstimateTemplatesController is not an interface.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Check isInterface returns false.
Test Data	- isInterface: false
Expected Result	- Controller class is not an interface.

Actual Result	Confirmed not an interface as expected.
Status	Pass
Severity	Low

Test Case ID	ETC-018
Title	Controller Is Not a Trait
Objective	Ensure EstimateTemplatesController is not a trait.
Preconditions	- Application running - Controller class file loaded
Test Steps	1. Create a ReflectionClass for EstimateTemplatesController. 2. Check isTrait returns false.
Test Data	- isTrait: false
Expected Result	- Controller class is not a trait.
Actual Result	Confirmed not a trait as expected.
Status	Pass
Severity	Low

Test Case ID	ETC-019
Title	Controller Class Is Loaded
Objective	Verify the EstimateTemplatesController class exists and is loaded.
Preconditions	- Application running
Test Steps	1. Use class_exists on EstimateTemplatesController.
Test Data	- class_exists(EstimateTemplatesController::class): true
Expected Result	- Class is loaded.
Actual Result	Class found as expected.
Status	Pass
Severity	High

Test Case ID	ETC-020
Title	Controller Uses Required Classes
Objective	Ensure required uses/import statements exist in the controller file.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Get controller file content. 2. Verify it contains uses for Controller, Estimate, and Request classes.
Test Data	- Expected import statements for Controller, Estimate, Request
Expected Result	- All required classes are imported in the controller file.
Actual Result	Required imports found as expected.
Status	Pass
Severity	High

Test Case ID	ETC-021
Title	File Structure Verification
Objective	Verify class and method structure in EstimateTemplatesController file.
Preconditions	- Application running - Controller file accessible

Test Steps	1. Retrieve file content. 2. Confirm class extends Controller and public __invoke method exists.
Test Data	- "class EstimateTemplatesController extends Controller" - "public function __invoke"
Expected Result	- Correct class and method structure present.
Actual Result	File structure verified as expected.
Status	Pass
Severity	High

Test Case ID	ETC-022
Title	Controller Has Compact Implementation
Objective	Ensure file size is small for simple controller.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Get file content length. 2. Verify size is under 1000 bytes.
Test Data	- File size: <1000 bytes
Expected Result	- File size is minimal, confirming compact code.
Actual Result	File size is compact as expected.
Status	Pass
Severity	Low

Test Case ID	ETC-023
Title	Minimal Line Count
Objective	Ensure controller file has minimal lines for simple logic.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Get line count from file. 2. Verify it's less than 50.
Test Data	- Line count: <50
Expected Result	- Minimal line count, confirming concise code.
Actual Result	Line count minimal as expected.
Status	Pass
Severity	Low

Test Case ID	ETC-024
Title	Authorization Usage in __invoke
Objective	Verify the __invoke method calls $this->authorize.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Retrieve file content. 2. Check presence of $this->authorize in __invoke method.
Test Data	- Presence of authorization call
Expected Result	- Authorization is enforced in __invoke.
Actual Result	Authorization found in __invoke as expected.
Status	Pass
Severity	High

Test Case ID	ETC-025
Title	__invoke Authorizes viewAny on Estimate
Objective	Ensure __invoke method authorizes viewAny policy on Estimate model.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Check if 'viewAny' is used with Estimate::class in authorization.
Test Data	- 'viewAny', 'Estimate::class' usage
Expected Result	- __invoke authorizes viewAny policy on Estimate.
Actual Result	ViewAny authorization confirmed as expected.
Status	Pass
Severity	High

Test Case ID	ETC-026
Title	__invoke Calls Estimate::estimateTemplates
Objective	Ensure method uses Estimate::estimateTemplates() to fetch templates.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Check presence of Estimate::estimateTemplates() in method implementation.
Test Data	- Method call: Estimate::estimateTemplates()
Expected Result	- __invoke method calls Estimate::estimateTemplates().
Actual Result	Method call confirmed as expected.
Status	Pass
Severity	High

Test Case ID	ETC-027
Title	__invoke Returns JSON Response
Objective	Verify that __invoke method returns a JSON response.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Check for use of response()->json in method implementation.
Test Data	- response()->json call
Expected Result	- Method returns a JSON response.
Actual Result	JSON response check passed as expected.
Status	Pass
Severity	High

Test Case ID	ETC-028
Title	__invoke Response Contains estimateTemplates Key
Objective	Ensure response includes estimateTemplates key.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Inspect method return value for presence of 'estimateTemplates' key.
Test Data	- Array key: 'estimateTemplates'
Expected Result	- estimateTemplates key present in response array.

Actual Result	estimateTemplates key verified in response.
Status	Pass
Severity	High

Test Case ID	ETC-029
Title	Only __invoke Method Defined
Objective	Verify controller exposes only __invoke as a public method.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Get public methods from ReflectionClass. 2. Filter methods to those defined in EstimateTemplatesController. 3. Verify only one public method (__invoke) exists.
Test Data	- Public methods count: expected 1
Expected Result	- Only __invoke method is defined as public.
Actual Result	Method count matched as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-030
Title	Sole Declared Method is __invoke
Objective	Confirm that __invoke is the only declared method.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Get all declared methods via ReflectionClass. 2. Filter to current class. 3. Verify only one method: __invoke.
Test Data	- Declared methods count: expected 1 - Method name: __invoke
Expected Result	- Only __invoke is declared.
Actual Result	Sole declared method check passed as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-031
Title	Namespace Depth Verification
Objective	Ensure namespace parts are correct in class definition.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Retrieve namespace of EstimateTemplatesController via ReflectionClass. 2. Split namespace by '\\'. 3. Verify required namespace parts exist.
Test Data	- Namespace parts: Crater, Http, Controllers, V1, Admin, Estimate
Expected Result	- All parts present in namespace.
Actual Result	Namespace depth and structure confirmed.
Status	Pass
Severity	Low

Test Case ID	ETC-032

Title	Namespace Structure Verification
Objective	Ensure controller namespace matches expected string exactly.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Retrieve namespace via ReflectionClass. 2. Compare to expected value.
Test Data	- Namespace: 'Crater\Http\Controllers\V1\Admin\Estimate'
Expected Result	- Namespace string matches exactly.
Actual Result	Namespace string matched as expected.
Status	Pass
Severity	Low

Test Case ID	ETC-033
Title	Controller Parent Is Controller
Objective	Verify parent class is Crater\Http\Controllers\Controller.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Get parent class via ReflectionClass. 2. Confirm parent is valid and matches expected name.
Test Data	- Parent class name: Crater\Http\Controllers\Controller
Expected Result	- Parent is correct Controller class.
Actual Result	Parent class confirmed as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-034
Title	Controller Inherits From Illuminate Routing Controller
Objective	Verify controller inheritance from base Laravel Controller.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Instantiate EstimateTemplatesController. 2. Confirm instance is also \Illuminate\Routing\Controller.
Test Data	- Instance type check
Expected Result	- Controller inherits from Laravel's Controller.
Actual Result	Inheritance confirmed as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-035
Title	Authorization Implemented Before Action
Objective	Confirm authorization logic is executed before main action logic in __invoke.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Retrieve method source. 2. Find position of $this->authorize and estimateTemplates logic. 3. Confirm authorization occurs first.
Test Data	- Positions of $this->authorize and 'estimateTemplates'

Expected Result	- Authorization precedes business logic.
Actual Result	Authorization is implemented before action as expected.
Status	Pass
Severity	High

Test Case ID	ETC-036
Title	Estimate Model Usage in Controller
Objective	Ensure Estimate model is referenced in controller.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Check controller file content for 'Estimate::'.
Test Data	- 'Estimate::' usage
Expected Result	- Controller uses Estimate model.
Actual Result	Estimate model usage confirmed.
Status	Pass
Severity	High

Test Case ID	ETC-037
Title	Response Includes estimateTemplates Key
Objective	Ensure returned response array contains estimateTemplates key.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Inspect response array in controller for 'estimateTemplates' key.
Test Data	- Array key: 'estimateTemplates'
Expected Result	- Response includes 'estimateTemplates'.
Actual Result	Response array key present as expected.
Status	Pass
Severity	High

Test Case ID	ETC-038
Title	Response Helper Usage
Objective	Verify use of response() helper in controller.
Preconditions	- Application running - Controller file accessible
Test Steps	1. Search for 'response()' call in file content.
Test Data	- response() usage
Expected Result	- response() helper is used.
Actual Result	response() helper usage verified as expected.
Status	Pass
Severity	High

Test Case ID	ETC-039
Title	Controller Has Proper Documentation
Objective	Confirm __invoke method has a doc comment.
Preconditions	- Application running - Controller class loaded

Test Steps	1. Retrieve __invoke method doc comment. 2. Confirm a doc comment exists.
Test Data	- Doc comment presence
Expected Result	- Method is documented via doc comment.
Actual Result	Doc comment found as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-040
Title	__invoke Method Has Return Type Documentation
Objective	Ensure __invoke doc comment contains @return tag.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Inspect doc comment for @return tag.
Test Data	- @return in doc comment
Expected Result	- @return tag present in documentation.
Actual Result	@return tag verified as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-041
Title	__invoke Method Has Parameter Documentation
Objective	Ensure doc comment contains @param tag for Request parameter.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Inspect doc comment for @param tag.
Test Data	- @param in doc comment
Expected Result	- @param tag present for parameter documentation.
Actual Result	@param tag found as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-042
Title	Controller Has Single Responsibility
Objective	Ensure controller follows Single Responsibility Principle.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Retrieve public methods defined in controller. 2. Confirm only __invoke method present.
Test Data	- Public method count: expected 1
Expected Result	- Single responsibility confirmed via single public method.
Actual Result	Single method confirmed as expected.
Status	Pass
Severity	Medium

Test Case ID	ETC-043

Title	Controller Follows Invokable Pattern
Objective	Verify controller follows Laravel invokable controller pattern.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Check for existence of __invoke method via ReflectionClass.
Test Data	- hasMethod('__invoke'): true
Expected Result	- Controller follows invokable pattern.
Actual Result	Invokable pattern confirmed as expected.
Status	Pass
Severity	Medium

File: EstimateViewedMail-Test.txt

Test Case ID	EVM-001
Title	Instantiation of EstimateViewedMail
Objective	Verify that the EstimateViewedMail class can be instantiated successfully.
Preconditions	- Application running - Class EstimateViewedMail is implemented and accessible - Database seeded (if required for context)
Test Steps	1. Define data array: ['estimate_number' => 'EST-001'] 2. Instantiate EstimateViewedMail with the data array 3. Verify that the created object is an instance of EstimateViewedMail
Test Data	- Input: ['estimate_number' => 'EST-001']
Expected Result	- Object is successfully instantiated and is an instance of EstimateViewedMail
Actual Result	Object was instantiated as expected.
Status	Pass
Severity	High

Test Case ID	EVM-002
Title	EstimateViewedMail Extends Mailable
Objective	Verify that EstimateViewedMail extends the Mailable base class.
Preconditions	- Application running - EstimateViewedMail and Mailable classes are implemented
Test Steps	1. Define data array: ['estimate_number' => 'EST-001'] 2. Instantiate EstimateViewedMail with the data array 3. Verify that the object is an instance of Mailable
Test Data	- Input: ['estimate_number' => 'EST-001']
Expected Result	- EstimateViewedMail object is an instance of Mailable
Actual Result	EstimateViewedMail object extends Mailable as expected.
Status	Pass
Severity	High

Test Case ID	EVM-003
Title	Correct Namespace for EstimateViewedMail
Objective	Verify that the EstimateViewedMail class is declared in the Crater\Mail namespace.
Preconditions	- Application running - EstimateViewedMail class is implemented
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Retrieve the namespace using getNamespaceName() 3. Check if the namespace is 'Crater\Mail'
Test Data	- Class: EstimateViewedMail
Expected Result	- Namespace returned is 'Crater\Mail'
Actual Result	Namespace is 'Crater\Mail'.
Status	Pass
Severity	Medium

Test Case ID	EVM-004
Title	EstimateViewedMail is Not Abstract

Objective	Ensure EstimateViewedMail is a concrete class and not abstract.
Preconditions	- Application running - EstimateViewedMail class implemented
Test Steps	1. Use ReflectionClass on EstimateViewedMail 2. Check isAbstract() status
Test Data	- Class: EstimateViewedMail
Expected Result	- isAbstract() returns false, class is not abstract
Actual Result	Class is non-abstract as expected.
Status	Pass
Severity	High

Test Case ID	EVM-005
Title	EstimateViewedMail Is Instantiable
Objective	Confirm that EstimateViewedMail can be instantiated (not an abstract/interface/trait).
Preconditions	- Application running - Class implemented
Test Steps	1. Use ReflectionClass on EstimateViewedMail 2. Call isInstantiable() and check value
Test Data	- Class: EstimateViewedMail
Expected Result	- isInstantiable() returns true
Actual Result	Class is instantiable.
Status	Pass
Severity	High

Test Case ID	EVM-006
Title	Constructor Assigns Data Property
Objective	Verify that the constructor assigns its parameter to the object's data property.
Preconditions	- Application running - EstimateViewedMail class implemented
Test Steps	1. Define $data = ['estimate_number' => 'EST-001', 'customer' => 'John Doe'] 2. Instantiate EstimateViewedMail with $data 3. Verify that $mail->data equals $data
Test Data	- Input: ['estimate_number' => 'EST-001', 'customer' => 'John Doe'] - Expected: $mail->data equals input array
Expected Result	- data property equals provided $data array
Actual Result	data property matched input array.
Status	Pass
Severity	High

Test Case ID	EVM-007
Title	Constructor Handles Array Data
Objective	Verify that EstimateViewedMail constructor supports array data with multiple keys.
Preconditions	- Application running
Test Steps	1. Define data array: ['key1' => 'value1', 'key2' => 'value2'] 2. Instantiate EstimateViewedMail with $data 3. Verify $mail->data is array 4. Verify $mail->data has keys 'key1' and 'key2'

Test Data	- Input: ['key1' => 'value1', 'key2' => 'value2']
Expected Result	- $mail->data is an array with keys 'key1' and 'key2'
Actual Result	data property was an array with required keys.
Status	Pass
Severity	High

Test Case ID	EVM-008
Title	Constructor Handles Empty Array Data
Objective	Verify that passing an empty array to the constructor results in an empty array data property.
Preconditions	- Application running
Test Steps	1. Define $data = [] 2. Instantiate EstimateViewedMail with $data 3. Verify $mail->data is array and is empty
Test Data	- Input: []
Expected Result	- $mail->data is an empty array
Actual Result	data property was an empty array.
Status	Pass
Severity	Medium

Test Case ID	EVM-009
Title	Constructor Handles Null Data
Objective	Confirm EstimateViewedMail supports null data parameter in constructor.
Preconditions	- Application running
Test Steps	1. Instantiate EstimateViewedMail with null 2. Verify $mail->data is null
Test Data	- Input: null
Expected Result	- $mail->data is null
Actual Result	data property was null.
Status	Pass
Severity	High

Test Case ID	EVM-010
Title	Constructor Handles Complex Nested Data
Objective	Ensure constructor can accept and preserve complex nested arrays in data property.
Preconditions	- Application running
Test Steps	1. Define complex $data array with nested structure 2. Instantiate EstimateViewedMail with $data 3. Verify $mail->data equals $data
Test Data	- Input: ['estimate' => ['number' => 'EST-001', 'items' => [['name' => 'Item 1', 'price' => 100], ['name' => 'Item 2', 'price' => 200]]]]

Expected Result	- $mail->data equals provided nested array
Actual Result	data property matched nested array.
Status	Pass
Severity	High

Test Case ID	EVM-011
Title	Data Property is Public
Objective	Verify the data property in EstimateViewedMail class is publicly accessible.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Retrieve 'data' property using getProperty('data') 3. Verify isPublic() returns true
Test Data	- Property: data
Expected Result	- data property is public
Actual Result	Property was public as expected.
Status	Pass
Severity	High

Test Case ID	EVM-012
Title	Data Property Direct Access
Objective	Confirm direct access to data property is allowed and returns correct values.
Preconditions	- Application running
Test Steps	1. Define $data = ['test' => 'value'] 2. Instantiate EstimateViewedMail with $data 3. Access $mail->data['test'] directly and check value
Test Data	- Input: ['test' => 'value']
Expected Result	- $mail->data['test'] equals 'value'
Actual Result	Direct property access worked as expected.
Status	Pass
Severity	High

Test Case ID	EVM-013
Title	Data Property Can Be Modified After Instantiation
Objective	Ensure data property can be reassigned after object creation.
Preconditions	- Application running
Test Steps	1. Instantiate EstimateViewedMail with ['initial' => 'data'] 2. Modify $mail->data to ['modified' => 'data'] 3. Verify $mail->data equals ['modified' => 'data']
Test Data	- Initial: ['initial' => 'data'] - Modified: ['modified' => 'data']
Expected Result	- $mail->data is modifiable and matches new array
Actual Result	data property could be modified as expected.
Status	Pass
Severity	High

Test Case ID	EVM-014
Title	EstimateViewedMail Has __construct Method

Objective	Verify EstimateViewedMail defines the constructor method.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Check if method '__construct' exists
Test Data	- Method: __construct
Expected Result	- Method '__construct' exists
Actual Result	Constructor method was present.
Status	Pass
Severity	High

Test Case ID	EVM-015
Title	EstimateViewedMail Has build Method
Objective	Confirm EstimateViewedMail implements the build method.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Check for method 'build'
Test Data	- Method: build
Expected Result	- Method 'build' exists
Actual Result	Build method found in class.
Status	Pass
Severity	High

Test Case ID	EVM-016
Title	build Method is Public
Objective	Ensure the build method is publicly accessible.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Get method 'build' 3. Verify isPublic() is true
Test Data	- Method: build
Expected Result	- build method is public
Actual Result	Method was public.
Status	Pass
Severity	High

Test Case ID	EVM-017
Title	build Method has No Parameters
Objective	Verify build method does not require any parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Get method 'build' 3. Check getNumberOfParameters() == 0
Test Data	- Method: build
Expected Result	- build method has no parameters
Actual Result	Method had no parameters.
Status	Pass

Severity	High

Test Case ID	EVM-018
Title	build Method is Not Static
Objective	Ensure build method is not declared static.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Get method 'build' 3. Verify isStatic() returns false
Test Data	- Method: build
Expected Result	- build is not static
Actual Result	Method was non-static.
Status	Pass
Severity	High

Test Case ID	EVM-019
Title	build Method Returns Self
Objective	Confirm that build method returns the current instance of EstimateViewedMail.
Preconditions	- Application running
Test Steps	1. Define data array ['estimate_number' => 'EST-001'] 2. Instantiate EstimateViewedMail 3. Call $mail->build() 4. Verify returned object is identical ($mail === $result)
Test Data	- Input: ['estimate_number' => 'EST-001']
Expected Result	- build returns $mail object itself
Actual Result	build returned self-instance as expected.
Status	Pass
Severity	High

Test Case ID	EVM-020
Title	build Method Can Be Called Multiple Times
Objective	Ensure calling build multiple times yields same instance.
Preconditions	- Application running
Test Steps	1. Instantiate EstimateViewedMail with ['estimate_number' => 'EST-001'] 2. Call build() twice 3. Verify both calls return same object
Test Data	- Input: ['estimate_number' => 'EST-001']
Expected Result	- Both build() calls return original mail instance
Actual Result	Multiple calls returned same object.
Status	Pass
Severity	High

Test Case ID	EVM-021
Title	EstimateViewedMail Uses Queueable Trait
Objective	Verify that class uses Illuminate\Bus\Queueable trait.
Preconditions	- Application running

Test Steps	1. Use ReflectionClass on EstimateViewedMail 2. Retrieve list of trait names 3. Confirm 'Illuminate\Bus\Queueable' included
Test Data	- Trait: Illuminate\Bus\Queueable
Expected Result	- Trait is used by EstimateViewedMail
Actual Result	Trait used as required.
Status	Pass
Severity	High

Test Case ID	EVM-022
Title	EstimateViewedMail Uses SerializesModels Trait
Objective	Confirm EstimateViewedMail uses Illuminate\Queue\SerializesModels trait.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Retrieve trait names 3. Check for 'Illuminate\Queue\SerializesModels'
Test Data	- Trait: Illuminate\Queue\SerializesModels
Expected Result	- Trait present in class
Actual Result	Trait was used as expected.
Status	Pass
Severity	High

Test Case ID	EVM-023
Title	Multiple Instances Creation
Objective	Ensure multiple independent instances of EstimateViewedMail can be created.
Preconditions	- Application running
Test Steps	1. Instantiate $mail1 with ['data1' => 'value1'] 2. Instantiate $mail2 with ['data2' => 'value2'] 3. Verify both are EstimateViewedMail instances, and they are not the same object
Test Data	- Input: ['data1' => 'value1'], ['data2' => 'value2']
Expected Result	- Both objects are instances of EstimateViewedMail, and are distinct
Actual Result	Instances were created and were independent.
Status	Pass
Severity	High

Test Case ID	EVM-024
Title	EstimateViewedMail Can Be Cloned
Objective	Verify cloning preserves object data and yields distinct instance.
Preconditions	- Application running
Test Steps	1. Instantiate EstimateViewedMail with ['test' => 'data'] 2. Clone the instance 3. Confirm clone is EstimateViewedMail, and equals original but is different object
Test Data	- Input: ['test' => 'data']
Expected Result	- Cloned object is of same type, has same data, but is distinct
Actual Result	Clone met all conditions.
Status	Pass

Severity	High

Test Case ID	EVM-025
Title	Type Hinting with EstimateViewedMail
Objective	Ensure EstimateViewedMail can be used as type hint in PHP functions.
Preconditions	- Application running
Test Steps	1. Define function that type hints EstimateViewedMail 2. Instantiate EstimateViewedMail 3. Pass it to function 4. Check returned object is original
Test Data	- Input: ['test' => 'data']
Expected Result	- Function accepts and returns mail object
Actual Result	Type hint was accepted and function returned mail object.
Status	Pass
Severity	High

Test Case ID	EVM-026
Title	EstimateViewedMail Is Not Final
Objective	Verify class is extensible and not declared final.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Check if isFinal() returns false
Test Data	- Class: EstimateViewedMail
Expected Result	- Class is not final
Actual Result	Class was non-final.
Status	Pass
Severity	Medium

Test Case ID	EVM-027
Title	EstimateViewedMail Is Not An Interface
Objective	Verify EstimateViewedMail is a class and not interface.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Check if isInterface() returns false
Test Data	- Class: EstimateViewedMail
Expected Result	- isInterface() returns false
Actual Result	Class was not an interface.
Status	Pass
Severity	Medium

Test Case ID	EVM-028
Title	EstimateViewedMail Is Not A Trait
Objective	Ensure EstimateViewedMail is an actual class (not trait).
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Check isTrait() returns false

Test Data	- Class: EstimateViewedMail
Expected Result	- isTrait() returns false
Actual Result	Class was not a trait.
Status	Pass
Severity	Medium

Test Case ID	EVM-029
Title	EstimateViewedMail Class is Loaded
Objective	Verify PHP class loader finds EstimateViewedMail.
Preconditions	- Application running
Test Steps	1. Use class_exists() for EstimateViewedMail 2. Check result
Test Data	- Class: EstimateViewedMail
Expected Result	- class_exists() returns true
Actual Result	Class was loaded.
Status	Pass
Severity	High

Test Case ID	EVM-030
Title	EstimateViewedMail Uses Required Classes
Objective	Ensure required PHP imports are used.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get file name 2. Read file contents 3. Verify usage of 'use Illuminate\Bus\Queueable', 'use Illuminate\Mail\Mailable', and 'use Illuminate\Queue\SerializesModels'
Test Data	- Imports: Queueable, Mailable, SerializesModels
Expected Result	- All required classes are imported
Actual Result	Imports present as required.
Status	Pass
Severity	High

Test Case ID	EVM-031
Title	File Has Expected Structure
Objective	Ensure file structure for EstimateViewedMail class matches standards.
Preconditions	- Application running
Test Steps	1. Access file of EstimateViewedMail using ReflectionClass 2. Check for 'class EstimateViewedMail extends Mailable' 3. Check for 'public function __construct' 4. Check for 'public function build()'
Test Data	- File content
Expected Result	- File contains all structure elements
Actual Result	File had expected structure.
Status	Pass
Severity	Medium

Test Case ID	EVM-032

Title	Compact File Implementation
Objective	Verify EstimateViewedMail file is succinct (<1000 bytes).
Preconditions	- Application running
Test Steps	1. Get file contents via ReflectionClass 2. Measure size 3. Confirm size < 1000 bytes
Test Data	- File size in bytes
Expected Result	- File size less than 1000 bytes
Actual Result	File was under size limit.
Status	Pass
Severity	Low

Test Case ID	EVM-033
Title	Minimal Line Count
Objective	Confirm EstimateViewedMail file contains <50 lines.
Preconditions	- Application running
Test Steps	1. Get file using ReflectionClass 2. Split content into lines 3. Confirm line count < 50
Test Data	- Line count
Expected Result	- Line count less than 50
Actual Result	Line count was below threshold.
Status	Pass
Severity	Low

Test Case ID	EVM-034
Title	build Method Uses From Configuration
Objective	Verify build function (in EstimateViewedMail) uses sender address/name from config.
Preconditions	- Application running
Test Steps	1. Access file via ReflectionClass 2. Check for lines calling 'from(config('mail.from.address'), config('mail.from.name'))'
Test Data	- File content
Expected Result	- Sender is set from config
Actual Result	From address set as required.
Status	Pass
Severity	High

Test Case ID	EVM-035
Title	build Method Uses Markdown View
Objective	Ensure build method uses markdown view for rendering emails.
Preconditions	- Application running
Test Steps	1. Access file via ReflectionClass 2. Check for 'markdown(' and 'emails.viewed.estimate' in file
Test Data	- File content
Expected Result	- Markdown view is used

Actual Result	Markdown was utilized for email view.
Status	Pass
Severity	High

Test Case ID	EVM-036
Title	build Method Passes Data to View
Objective	Confirm $this->data is made available to the email view by build method.
Preconditions	- Application running
Test Steps	1. Access file and look for use of $this->data in build()
Test Data	- File content
Expected Result	- $this->data passed to view
Actual Result	Data available to view in build method.
Status	Pass
Severity	High

Test Case ID	EVM-037
Title	build Method Returns $this
Objective	Verify build method final statement returns self ($this).
Preconditions	- Application running
Test Steps	1. Access file via ReflectionClass 2. Search for 'return $this' in build method
Test Data	- File content
Expected Result	- build method returns $this
Actual Result	Method returned $this as expected.
Status	Pass
Severity	High

Test Case ID	EVM-038
Title	Constructor Assigns Parameter to Data Property (Implementation)
Objective	Ensure that implementation assigns constructor argument to data property.
Preconditions	- Application running
Test Steps	1. Access file for EstimateViewedMail 2. Verify presence of '$this->data = $data' in constructor
Test Data	- File content
Expected Result	- Assignment statement exists in constructor
Actual Result	Assignment present in code.
Status	Pass
Severity	High

Test Case ID	EVM-039
Title	Constructor Accepts Data Parameter
Objective	Validate that constructor defines and accepts a data parameter.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Retrieve __construct method parameters 3. Check that first parameter is named 'data'

Test Data	- Parameter name
Expected Result	- Parameter named 'data' found
Actual Result	Parameter named 'data' present.
Status	Pass
Severity	High

Test Case ID	EVM-040
Title	Constructor Has Documentation Block
Objective	Ensure __construct method is properly documented.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Get __construct method 3. Verify existence of doc comment
Test Data	- Doc comment presence
Expected Result	- Doc comment exists for constructor
Actual Result	Documentation was present.
Status	Pass
Severity	Low

Test Case ID	EVM-041
Title	build Method Has Documentation Block
Objective	Ensure build method is properly documented.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Get build method 3. Verify doc comment exists
Test Data	- Doc comment for build method
Expected Result	- Doc comment exists
Actual Result	Doc comment was present.
Status	Pass
Severity	Low

Test Case ID	EVM-042
Title	build Method Has Return Type Documentation
Objective	Confirm build method documentation includes @return statement.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for EstimateViewedMail 2. Get build method 3. Check doc comment contains '@return'
Test Data	- Doc comment
Expected Result	- '@return' is present
Actual Result	Return type was documented.
Status	Pass
Severity	Low

Test Case ID	EVM-043

Title	Data is Preserved Through Constructor
Objective	Validate that original data passed into constructor is preserved post-instantiation.
Preconditions	- Application running
Test Steps	1. Define $originalData = ['estimate_number' => 'EST-123', 'customer_name' => 'Test Customer', 'total' => 1500.50] 2. Instantiate EstimateViewedMail with $originalData 3. Confirm $mail->data equals $originalData 4. Verify each key matches expected value
Test Data	- Input: ['estimate_number' => 'EST-123', 'customer_name' => 'Test Customer', 'total' => 1500.50] - Expected: Same array, values as above
Expected Result	- $mail->data equals original, values match for each key
Actual Result	Data was preserved and keys matched expected values.
Status	Pass
Severity	High

Test Case ID	EVM-044
Title	Different Instances Have Independent Data
Objective	Ensure two EstimateViewedMail instances maintain independent data.
Preconditions	- Application running
Test Steps	1. Instantiate $mail1 with ['estimate' => 'EST-001'] 2. Instantiate $mail2 with ['estimate' => 'EST-002'] 3. Verify $mail1->data != $mail2->data 4. Check individual keys
Test Data	- Input: ['estimate' => 'EST-001'], ['estimate' => 'EST-002']
Expected Result	- $mail1->data and $mail2->data are different
Actual Result	Instances had independent data.
Status	Pass
Severity	High

Test Case ID	EVM-045
Title	EstimateViewedMail Inherits Mailable Methods
Objective	Verify EstimateViewedMail inherits standard Mailable methods.
Preconditions	- Application running
Test Steps	1. Instantiate EstimateViewedMail with ['test' => 'data'] 2. Check for methods 'to', 'subject', 'from'
Test Data	- Input: ['test' => 'data']
Expected Result	- All required methods are present
Actual Result	Methods were inherited as expected.
Status	Pass
Severity	High

Test Case ID	EVM-046

Title	EstimateViewedMail Can Use Mailable Features
Objective	Test that Mailable methods can be called (e.g., to, subject, from).
Preconditions	- Application running
Test Steps	1. Instantiate EstimateViewedMail with ['test' => 'data'] 2. Check if 'to' and 'subject' are callable
Test Data	- Input: ['test' => 'data']
Expected Result	- Methods are callable
Actual Result	Mailable features were callable.
Status	Pass
Severity	High

File: EstimatesRequest-Test.txt

Test Case ID	ER-001
Title	Instantiation of EstimatesRequest
Objective	Verify that the EstimatesRequest class can be instantiated successfully.
Preconditions	- Application running - Database seeded - PHP environment available
Test Steps	1. Instantiate the EstimatesRequest class. 2. Check the type of the created object.
Test Data	- Class: EstimatesRequest
Expected Result	- The object is an instance of EstimatesRequest.
Actual Result	- The object is an instance of EstimatesRequest.
Status	Pass
Severity	Medium

Test Case ID	ER-002
Title	EstimatesRequest extends FormRequest
Objective	Confirm that EstimatesRequest extends the FormRequest class.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate EstimatesRequest. 2. Check if the object is an instance of Illuminate\Foundation\Http\FormRequest.
Test Data	- Class: EstimatesRequest
Expected Result	- The object is an instance of FormRequest.
Actual Result	- The object is an instance of FormRequest.
Status	Pass
Severity	High

Test Case ID	ER-003
Title	EstimatesRequest namespace verification
Objective	Ensure EstimatesRequest class resides in the correct namespace.
Preconditions	- Application running - Database seeded
Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve its namespace.
Test Data	- Class: EstimatesRequest
Expected Result	- The namespace is 'Crater\Http\Requests'.
Actual Result	- The namespace is 'Crater\Http\Requests'.
Status	Pass
Severity	Low

Test Case ID	ER-004
Title	EstimatesRequest is not abstract
Objective	Ensure the EstimatesRequest class is not abstract.
Preconditions	- Application running

Test Steps	1. Reflect on EstimatesRequest class. 2. Check if the class is abstract.
Test Data	- Class: EstimatesRequest
Expected Result	- The class is not abstract.
Actual Result	- The class is not abstract.
Status	Pass
Severity	Medium

Test Case ID	ER-005
Title	EstimatesRequest class is instantiable
Objective	Verify the EstimatesRequest class can be instantiated.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Check if the class is instantiable.
Test Data	- Class: EstimatesRequest
Expected Result	- The class is instantiable.
Actual Result	- The class is instantiable.
Status	Pass
Severity	Medium

Test Case ID	ER-006
Title	EstimatesRequest has authorize method
Objective	Check that the authorize method exists in EstimatesRequest.
Preconditions	- Application running
Test Steps	1. Instantiate EstimatesRequest. 2. Check if 'authorize' method exists.
Test Data	- Method: authorize
Expected Result	- 'authorize' method exists.
Actual Result	- 'authorize' method exists.
Status	Pass
Severity	High

Test Case ID	ER-007
Title	EstimatesRequest has rules method
Objective	Check that the rules method exists in EstimatesRequest.
Preconditions	- Application running
Test Steps	1. Instantiate EstimatesRequest. 2. Check if 'rules' method exists.
Test Data	- Method: rules
Expected Result	- 'rules' method exists.
Actual Result	- 'rules' method exists.
Status	Pass
Severity	High

Test Case ID	ER-008
Title	EstimatesRequest has getEstimatePayload method

Objective	Verify existence of getEstimatePayload method in EstimatesRequest.
Preconditions	- Application running
Test Steps	1. Instantiate EstimatesRequest. 2. Check if 'getEstimatePayload' method exists.
Test Data	- Method: getEstimatePayload
Expected Result	- 'getEstimatePayload' method exists.
Actual Result	- 'getEstimatePayload' method exists.
Status	Pass
Severity	High

Test Case ID	ER-009
Title	Authorize method returns true
Objective	Verify that the authorize method of EstimatesRequest returns true.
Preconditions	- Application running
Test Steps	1. Instantiate EstimatesRequest. 2. Call authorize method.
Test Data	- Method: authorize
Expected Result	- authorize returns true.
Actual Result	- authorize returns true.
Status	Pass
Severity	High

Test Case ID	ER-010
Title	Authorize method is public
Objective	Verify that the authorize method is public.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve the authorize method. 3. Check if the method is public.
Test Data	- Method: authorize
Expected Result	- Method is public.
Actual Result	- Method is public.
Status	Pass
Severity	High

Test Case ID	ER-011
Title	Authorize method has no parameters
Objective	Ensure authorize method does not accept parameters.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve the authorize method. 3. Check the parameter count.
Test Data	- Method: authorize
Expected Result	- Method has zero parameters.
Actual Result	- Method has zero parameters.
Status	Pass

Severity	Medium

Test Case ID	ER-012
Title	Authorize method returns boolean
Objective	Verify that authorize method returns a boolean value.
Preconditions	- Application running
Test Steps	1. Instantiate EstimatesRequest. 2. Call the authorize method. 3. Check the returned value type.
Test Data	- Method: authorize
Expected Result	- Return type is boolean.
Actual Result	- Return type is boolean.
Status	Pass
Severity	High

Test Case ID	ER-013
Title	Rules method is public
Objective	Verify that the rules method of EstimatesRequest is public.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve the rules method. 3. Check if the method is public.
Test Data	- Method: rules
Expected Result	- Method is public.
Actual Result	- Method is public.
Status	Pass
Severity	High

Test Case ID	ER-014
Title	Rules method has no parameters
Objective	Check that rules method does not accept parameters.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve the rules method. 3. Check the parameter count.
Test Data	- Method: rules
Expected Result	- Method has zero parameters.
Actual Result	- Method has zero parameters.
Status	Pass
Severity	Medium

Test Case ID	ER-015
Title	getEstimatePayload method is public
Objective	Verify that getEstimatePayload method is public.
Preconditions	- Application running

Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve the getEstimatePayload method. 3. Check if the method is public.
Test Data	- Method: getEstimatePayload
Expected Result	- Method is public.
Actual Result	- Method is public.
Status	Pass
Severity	High

Test Case ID	ER-016
Title	getEstimatePayload method has no parameters
Objective	Ensure getEstimatePayload method does not accept parameters.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve the getEstimatePayload method. 3. Check the parameter count.
Test Data	- Method: getEstimatePayload
Expected Result	- Method has zero parameters.
Actual Result	- Method has zero parameters.
Status	Pass
Severity	Medium

Test Case ID	ER-017
Title	Methods are not static
Objective	Confirm that authorize, rules, and getEstimatePayload methods are not static.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve all three methods. 3. Check if any method is static.
Test Data	- Methods: authorize, rules, getEstimatePayload
Expected Result	- All methods are not static.
Actual Result	- All methods are not static.
Status	Pass
Severity	Medium

Test Case ID	ER-018
Title	Methods are not abstract
Objective	Confirm that authorize, rules, and getEstimatePayload methods are not abstract.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Retrieve all three methods. 3. Check if any method is abstract.
Test Data	- Methods: authorize, rules, getEstimatePayload
Expected Result	- All methods are not abstract.
Actual Result	- All methods are not abstract.
Status	Pass

Severity	Medium

Test Case ID	ER-019
Title	Multiple EstimatesRequest instances creation
Objective	Verify that multiple instances of EstimatesRequest can be created independently.
Preconditions	- Application running
Test Steps	1. Create first EstimatesRequest instance. 2. Create second EstimatesRequest instance. 3. Check both are instances of EstimatesRequest. 4. Ensure they are not the same object.
Test Data	- Instances: request1, request2
Expected Result	- Both are instances of EstimatesRequest. - They are not the same object.
Actual Result	- Both are instances of EstimatesRequest and are different objects.
Status	Pass
Severity	Low

Test Case ID	ER-020
Title	EstimatesRequest can be cloned
Objective	Verify that an EstimatesRequest instance can be cloned.
Preconditions	- Application running
Test Steps	1. Instantiate EstimatesRequest. 2. Clone the instance. 3. Check the type of the clone. 4. Ensure original and clone are not the same object.
Test Data	- Object: EstimatesRequest
Expected Result	- The clone is an instance of EstimatesRequest and distinct from the original.
Actual Result	- The clone is an instance of EstimatesRequest and distinct.
Status	Pass
Severity	Low

Test Case ID	ER-021
Title	Type hinting with EstimatesRequest
Objective	Ensure EstimatesRequest can be used in function type hints.
Preconditions	- Application running
Test Steps	1. Define a function with EstimatesRequest type hint. 2. Pass an EstimatesRequest instance to the function. 3. Return and check the result.
Test Data	- Function input: EstimatesRequest instance
Expected Result	- The function accepts and returns the correct object.
Actual Result	- Function accepts and returns the correct object.
Status	Pass
Severity	Low

Test Case ID	ER-022
Title	EstimatesRequest is not final
Objective	Confirm that the EstimatesRequest class is not final.

Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Check if the class is final.
Test Data	- Class: EstimatesRequest
Expected Result	- Class is not final.
Actual Result	- Class is not final.
Status	Pass
Severity	Medium

Test Case ID	ER-023
Title	EstimatesRequest is not an interface
Objective	Verify EstimatesRequest is a class, not an interface.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Check if it's an interface.
Test Data	- Class: EstimatesRequest
Expected Result	- Class is not an interface.
Actual Result	- Class is not an interface.
Status	Pass
Severity	Medium

Test Case ID	ER-024
Title	EstimatesRequest is not a trait
Objective	Verify EstimatesRequest is not a trait.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Check if it's a trait.
Test Data	- Class: EstimatesRequest
Expected Result	- Class is not a trait.
Actual Result	- Class is not a trait.
Status	Pass
Severity	Medium

Test Case ID	ER-025
Title	EstimatesRequest class is loaded
Objective	Ensure that the EstimatesRequest class is loaded in runtime.
Preconditions	- Application running
Test Steps	1. Check if EstimatesRequest class exists using class_exists.
Test Data	- Class: EstimatesRequest
Expected Result	- class_exists returns true.
Actual Result	- class_exists returns true.
Status	Pass
Severity	High

Test Case ID	ER-026

Title	EstimatesRequest uses required classes
Objective	Verify that EstimatesRequest imports all necessary classes via "use".
Preconditions	- Application running - File accessible
Test Steps	1. Reflect on EstimatesRequest class. 2. Read its file contents. 3. Verify presence of all required use statements.
Test Data	- Required imports: CompanySetting, Customer, Estimate, FormRequest, Rule.
Expected Result	- All required import statements present.
Actual Result	- All required import statements present.
Status	Pass
Severity	Medium

Test Case ID	ER-027
Title	EstimatesRequest file has expected structure
Objective	Ensure the EstimatesRequest file has essential class and method definitions.
Preconditions	- Application running - File accessible
Test Steps	1. Reflect on EstimatesRequest class. 2. Read its file contents. 3. Check for class declaration and public method definitions.
Test Data	- File: EstimatesRequest.php
Expected Result	- Declaration: 'class EstimatesRequest extends FormRequest' - Methods: 'public function authorize()', 'public function rules()', 'public function getEstimatePayload()' present.
Actual Result	- Class declaration and all method signatures present.
Status	Pass
Severity	High

Test Case ID	ER-028
Title	EstimatesRequest has comprehensive implementation
Objective	Verify that EstimatesRequest file has substantial implementation (>3000 bytes).
Preconditions	- Application running - File accessible
Test Steps	1. Reflect on EstimatesRequest class. 2. Read its file contents. 3. Check the file size in bytes.
Test Data	- File: EstimatesRequest.php
Expected Result	- File size is greater than 3000 bytes.
Actual Result	- File size is greater than 3000 bytes.
Status	Pass
Severity	Medium

Test Case ID	ER-029
Title	EstimatesRequest has reasonable line count
Objective	Verify that EstimatesRequest file has a reasonable number of lines (>100).

Preconditions	- Application running - File accessible
Test Steps	1. Reflect on EstimatesRequest class. 2. Read its file contents. 3. Count total lines.
Test Data	- File: EstimatesRequest.php
Expected Result	- Line count greater than 100.
Actual Result	- Line count greater than 100.
Status	Pass
Severity	Low

Test Case ID	ER-030
Title	Rules method contains validation logic
Objective	Confirm that the rules method specifies validation for required estimate fields.
Preconditions	- Application running - File accessible
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for validation rule keys.
Test Data	- Validation keys: estimate_date, expiry_date, customer_id, estimate_number, exchange_rate.
Expected Result	- All mentioned validation keys present.
Actual Result	- All mentioned validation keys present.
Status	Pass
Severity	High

Test Case ID	ER-031
Title	Rules method includes item validation
Objective	Validate that the rules method covers item and item fields.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for validation keys related to items.
Test Data	- Validation keys: items, items.*, items.*.name, items.*.quantity, items.*.price.
Expected Result	- All required item-related validation keys present.
Actual Result	- All required item-related validation keys present.
Status	Pass
Severity	High

Test Case ID	ER-032
Title	Rules method includes financial fields
Objective	Ensure rules method includes financial-related fields in validation.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for financial-related validation keys.
Test Data	- Validation keys: discount, discount_val, sub_total, total, tax.
Expected Result	- All financial fields present in validation.

Actual Result	- All financial fields present in validation.
Status	Pass
Severity	High

Test Case ID	ER-033
Title	Rules method uses Rule::unique
Objective	Verify unique validation is used via Rule::unique in rules method.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'Rule::unique'.
Test Data	- Validation method: Rule::unique
Expected Result	- 'Rule::unique' is present in rules.
Actual Result	- 'Rule::unique' is present in rules.
Status	Pass
Severity	High

Test Case ID	ER-034
Title	Rules method checks for PUT method
Objective	Ensure method-sensitive logic (PUT) in rules implementation.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'isMethod' and 'PUT'.
Test Data	- Keywords: isMethod, PUT
Expected Result	- Presence of 'isMethod' and 'PUT' logic.
Actual Result	- 'isMethod' and 'PUT' logic are present.
Status	Pass
Severity	High

Test Case ID	ER-035
Title	getEstimatePayload contains payload logic
Objective	Validate the presence of estimate data setup logic in getEstimatePayload.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for payload-related fields.
Test Data	- Payload keys: creator_id, status, company_id
Expected Result	- All keys are present in payload logic.
Actual Result	- All keys are present in payload logic.
Status	Pass
Severity	High

Test Case ID	ER-036
Title	getEstimatePayload uses CompanySetting
Objective	Ensure getEstimatePayload uses CompanySetting for configuration values.

Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'CompanySetting::getSetting'.
Test Data	- Method: CompanySetting::getSetting
Expected Result	- 'CompanySetting::getSetting' is used.
Actual Result	- 'CompanySetting::getSetting' is used.
Status	Pass
Severity	Medium

Test Case ID	ER-037
Title	getEstimatePayload calculates base values
Objective	Verify correct calculation of financial base values in getEstimatePayload.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'base_discount_val', 'base_sub_total', 'base_total', 'base_tax'.
Test Data	- Keys: base_discount_val, base_sub_total, base_total, base_tax
Expected Result	- All base value calculations are present.
Actual Result	- All base value calculations are present.
Status	Pass
Severity	High

Test Case ID	ER-038
Title	getEstimatePayload handles exchange rate
Objective	Confirm that exchange rate handling is implemented in payload method.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'exchange_rate'.
Test Data	- Key: exchange_rate
Expected Result	- exchange_rate logic is present.
Actual Result	- exchange_rate logic is present.
Status	Pass
Severity	High

Test Case ID	ER-039
Title	getEstimatePayload uses collect helper
Objective	Ensure getEstimatePayload uses Laravel's collect helper and associated methods.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'collect', 'except', 'merge', 'toArray'.
Test Data	- Methods: collect, except, merge, toArray
Expected Result	- All helper methods are used appropriately.
Actual Result	- All helper methods are used.

Status	Pass
Severity	Medium

Test Case ID	ER-040
Title	getEstimatePayload references Estimate statuses
Objective	Verify reference to Estimate status constants in getEstimatePayload.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'Estimate::STATUS_SENT' and 'Estimate::STATUS_DRAFT'.
Test Data	- Constants: Estimate::STATUS_SENT, Estimate::STATUS_DRAFT
Expected Result	- Both status constants are referenced.
Actual Result	- Both status constants are referenced.
Status	Pass
Severity	High

Test Case ID	ER-041
Title	getEstimatePayload checks for estimateSend
Objective	Confirm that getEstimatePayload checks for estimateSend flag in payload.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'estimateSend' logic.
Test Data	- Key: estimateSend
Expected Result	- estimateSend is checked.
Actual Result	- estimateSend is present and checked.
Status	Pass
Severity	High

Test Case ID	ER-042
Title	Rules method includes template_name validation
Objective	Verify template_name field is validated in rules method.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'template_name' in validation rules.
Test Data	- Key: template_name
Expected Result	- template_name is validated.
Actual Result	- template_name is validated.
Status	Pass
Severity	Medium

Test Case ID	ER-043
Title	Rules method validates item description
Objective	Ensure item descriptions are validated in rules method.
Preconditions	- Application running

Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'items.*.description'.
Test Data	- Key: items.*.description
Expected Result	- items.*.description is validated.
Actual Result	- items.*.description is validated.
Status	Pass
Severity	Medium

Test Case ID	ER-044
Title	Rules method checks currency matching
Objective	Validate that currency field has appropriate matching logic in rules.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'currency' in validation logic.
Test Data	- Key: currency
Expected Result	- currency field has matching logic/check.
Actual Result	- currency field has matching logic/check.
Status	Pass
Severity	High

Test Case ID	ER-045
Title	getEstimatePayload retrieves customer currency
Objective	Confirm that getEstimatePayload retrieves currency information from customer.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'Customer::find' and 'currency_id'.
Test Data	- Methods: Customer::find - Key: currency_id
Expected Result	- Customer currency_id is retrieved.
Actual Result	- Customer currency_id is retrieved.
Status	Pass
Severity	High

Test Case ID	ER-046
Title	getEstimatePayload uses tax_per_item setting
Objective	Verify getEstimatePayload uses company setting for tax per item.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'tax_per_item'.
Test Data	- Key: tax_per_item
Expected Result	- tax_per_item setting is used.
Actual Result	- tax_per_item setting is used.
Status	Pass

Severity	Medium

Test Case ID	ER-047
Title	getEstimatePayload uses discount_per_item setting
Objective	Verify getEstimatePayload uses company setting for discount per item.
Preconditions	- Application running
Test Steps	1. Reflect on EstimatesRequest class. 2. Read file contents. 3. Search for 'discount_per_item'.
Test Data	- Key: discount_per_item
Expected Result	- discount_per_item setting is used.
Actual Result	- discount_per_item setting is used.
Status	Pass
Severity	Medium

File: EventServiceProvider-Test.txt

Test Case ID	ESP-001
Title	Instantiation of EventServiceProvider
Objective	Verify that an instance of EventServiceProvider can be created.
Preconditions	- Application running - All dependencies (Laravel, Crater) installed - Database seeded
Test Steps	1. Invoke the app() helper to retrieve the application instance. 2. Create a new EventServiceProvider instance with the application. 3. Assert the created instance is of type EventServiceProvider.
Test Data	- app instance (default Laravel container) - EventServiceProvider
Expected Result	- The created object is an instance of EventServiceProvider.
Actual Result	- The EventServiceProvider instance is created and is of correct type.
Status	Pass
Severity	High

Test Case ID	ESP-002
Title	Inheritance from ServiceProvider
Objective	Ensure EventServiceProvider extends Laravel’s EventServiceProvider class.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Retrieve the application instance using app(). 2. Instantiate EventServiceProvider with application. 3. Assert the instance is of type Illuminate\Foundation\Support\Providers\EventServiceProvider.
Test Data	- app instance - EventServiceProvider
Expected Result	- EventServiceProvider instance is also an instance of Laravel’s ServiceProvider.
Actual Result	- Inheritance confirmed; EventServiceProvider is an instance of ServiceProvider.
Status	Pass
Severity	High

Test Case ID	ESP-003
Title	Correct Namespace Assignment
Objective	Verify that EventServiceProvider resides in the Crater\Providers namespace.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass for EventServiceProvider. 2. Get the namespace name via getNamespaceName(). 3. Assert that the namespace equals 'Crater\Providers'.
Test Data	- EventServiceProvider class
Expected Result	- Namespace retrieved matches 'Crater\Providers'.
Actual Result	- Namespace is correctly set to 'Crater\Providers'.
Status	Pass
Severity	Medium

Test Case ID	ESP-004
Title	Abstract Class Verification
Objective	Ensure EventServiceProvider is not abstract and can be instantiated.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass for EventServiceProvider. 2. Call isAbstract() method. 3. Assert the result is false.
Test Data	- EventServiceProvider class
Expected Result	- isAbstract() returns false.
Actual Result	- EventServiceProvider is not abstract.
Status	Pass
Severity	High

Test Case ID	ESP-005
Title	Instantiability of EventServiceProvider Class
Objective	Confirm that EventServiceProvider is instantiable.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass for EventServiceProvider. 2. Call isInstantiable() on the reflection object. 3. Assert the result is true.
Test Data	- EventServiceProvider class
Expected Result	- isInstantiable() returns true.
Actual Result	- EventServiceProvider is instantiable.
Status	Pass
Severity	High

Test Case ID	ESP-006
Title	Existence of Listen Property
Objective	Ensure EventServiceProvider has a 'listen' property.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass for EventServiceProvider. 2. Use hasProperty('listen') to check for the property. 3. Assert the result is true.
Test Data	- EventServiceProvider class
Expected Result	- hasProperty('listen') returns true.
Actual Result	- 'listen' property is present.
Status	Pass
Severity	High

Test Case ID	ESP-007
Title	'listen' Property Visibility
Objective	Verify that the 'listen' property is protected.
Preconditions	- Application running - EventServiceProvider class loaded

Test Steps	1. Use ReflectionClass for EventServiceProvider. 2. Get ReflectionProperty for 'listen'. 3. Assert property isProtected() returns true.
Test Data	- EventServiceProvider class
Expected Result	- 'listen' is protected.
Actual Result	- 'listen' property is confirmed as protected.
Status	Pass
Severity	High

Test Case ID	ESP-008
Title	Event Mappings Defined in Listen Property
Objective	Check that 'listen' property is a non-empty array containing event mappings.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Use ReflectionClass to access 'listen'. 3. Set property accessible and get its value. 4. Assert value is an array and not empty.
Test Data	- EventServiceProvider instance
Expected Result	- 'listen' is a non-empty array of event mappings.
Actual Result	- 'listen' contains event mappings as an array.
Status	Pass
Severity	High

Test Case ID	ESP-009
Title	Mapping of UpdateFinished Event
Objective	Verify that UpdateFinished event is mapped in the 'listen' property.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Use ReflectionClass to access 'listen' and make it accessible. 3. Assert array key exists for UpdateFinished::class.
Test Data	- EventServiceProvider instance - UpdateFinished::class
Expected Result	- 'listen' array contains key UpdateFinished::class.
Actual Result	- UpdateFinished event is mapped in 'listen'.
Status	Pass
Severity	High

Test Case ID	ESP-010
Title	Mapping of Registered Event
Objective	Verify that Registered event is mapped in the 'listen' property.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Use ReflectionClass to access 'listen' property. 3. Assert array key exists for Registered::class.
Test Data	- EventServiceProvider instance - Registered::class

Expected Result	- 'listen' contains key Registered::class.
Actual Result	- Registered event is mapped in 'listen'.
Status	Pass
Severity	High

Test Case ID	ESP-011
Title	Listeners Array for UpdateFinished
Objective	Confirm UpdateFinished event maps to an array with 8 listeners.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Access 'listen' property using ReflectionClass. 3. Retrieve value and check UpdateFinished::class mapping. 4. Assert that it is an array and contains 8 elements.
Test Data	- EventServiceProvider instance
Expected Result	- UpdateFinished maps to an array with 8 listeners.
Actual Result	- 8 listeners present for UpdateFinished.
Status	Pass
Severity	High

Test Case ID	ESP-012
Title	Version110 Listener for UpdateFinished
Objective	Ensure Version110 listener is present for UpdateFinished event.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Access 'listen' property using ReflectionClass. 3. Check that UpdateFinished::class listeners array contains Version110 class.
Test Data	- EventServiceProvider instance - \Crater\Listeners\Updates\v1\Version110::class
Expected Result	- Version110 listener is included for UpdateFinished event.
Actual Result	- Version110 listener is present.
Status	Pass
Severity	High

Test Case ID	ESP-013
Title	Inclusion of All v2 Version Listeners for UpdateFinished
Objective	Verify all v2 listeners are present for UpdateFinished event mapping.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Make 'listen' accessible via ReflectionClass. 3. Assert all v2 listener classes are present in UpdateFinished::class mapping.
Test Data	- \Crater\Listeners\Updates\v2\Version200, Version201, Version202, Version210
Expected Result	- All specified v2 listeners included.
Actual Result	- All v2 listeners present.
Status	Pass

Severity	High

Test Case ID	ESP-014
Title	Inclusion of All v3 Version Listeners for UpdateFinished
Objective	Confirm all v3 listeners are present for UpdateFinished event.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Access 'listen' property. 3. Assert all v3 listener classes are present.
Test Data	- \Crater\Listeners\Updates\v3\Version300, Version310, Version311
Expected Result	- All specified v3 listeners included.
Actual Result	- All v3 listeners present.
Status	Pass
Severity	High

Test Case ID	ESP-015
Title	Single Listener for Registered Event
Objective	Confirm Registered event maps to array containing one listener.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Access 'listen' property. 3. Assert array mapped to Registered::class has count of 1.
Test Data	- EventServiceProvider instance
Expected Result	- Registered event maps to array with one listener.
Actual Result	- Registered event has exactly one listener.
Status	Pass
Severity	High

Test Case ID	ESP-016
Title	SendEmailVerificationNotification Listener for Registered Event
Objective	Verify SendEmailVerificationNotification is mapped as listener for Registered event.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Access 'listen' property. 3. Assert Registered::class listeners contains SendEmailVerificationNotification class.
Test Data	- \Illuminate\Auth\Listeners\SendEmailVerificationNotification::class
Expected Result	- Listener is present for Registered event.
Actual Result	- SendEmailVerificationNotification listener is present.
Status	Pass
Severity	High

Test Case ID	ESP-017
Title	Existence of Boot Method

Objective	Validate boot method exists in EventServiceProvider.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Verify method 'boot' exists via hasMethod('boot').
Test Data	- EventServiceProvider class
Expected Result	- boot method exists in EventServiceProvider.
Actual Result	- boot method is present.
Status	Pass
Severity	High

Test Case ID	ESP-018
Title	Public Visibility of Boot Method
Objective	Ensure boot method is publicly accessible.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Retrieve 'boot' method. 3. Assert method is public.
Test Data	- EventServiceProvider class
Expected Result	- boot method is public.
Actual Result	- boot method has public visibility.
Status	Pass
Severity	High

Test Case ID	ESP-019
Title	No Parameters for Boot Method
Objective	Confirm boot method does not require parameters.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Retrieve 'boot' method. 3. Assert getNumberOfParameters() == 0.
Test Data	- EventServiceProvider class
Expected Result	- boot method has zero parameters.
Actual Result	- boot method does not accept parameters.
Status	Pass
Severity	High

Test Case ID	ESP-020
Title	Boot Method Non-static Verification
Objective	Confirm boot method is not static.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Retrieve 'boot' method. 3. Assert isStatic() returns false.
Test Data	- EventServiceProvider class

Expected Result	- boot is an instance method and not static.
Actual Result	- boot method is correctly non-static.
Status	Pass
Severity	High

Test Case ID	ESP-021
Title	Parent Boot Call in Boot Method
Objective	Verify boot method calls parent::boot().
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Read file content. 3. Assert 'parent::boot()' string is present.
Test Data	- EventServiceProvider file content
Expected Result	- boot method contains call to parent::boot().
Actual Result	- parent::boot() is called in boot method.
Status	Pass
Severity	High

Test Case ID	ESP-022
Title	Instantiating Multiple Instances
Objective	Confirm multiple distinct EventServiceProvider objects can be created.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate two EventServiceProvider objects. 2. Assert both are instances. 3. Assert they are not the same instance.
Test Data	- app instance
Expected Result	- Two distinct, valid instances are created.
Actual Result	- Multiple distinct instances created successfully.
Status	Pass
Severity	High

Test Case ID	ESP-023
Title	Cloning EventServiceProvider Instance
Objective	Verify that EventServiceProvider can be cloned, resulting in a distinct object.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Clone the instance. 3. Assert the clone is an instance and not the same as original.
Test Data	- app instance
Expected Result	- Clone is a new instance, not identical to original.
Actual Result	- Clone successfully generated and is a separate object.
Status	Pass
Severity	High

Test Case ID	ESP-024
Title	EventServiceProvider Type Hint Usability
Objective	Ensure EventServiceProvider can be used in function type hints.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Define a function with EventServiceProvider type-hinted parameter. 2. Pass EventServiceProvider instance. 3. Assert returned value matches passed instance.
Test Data	- EventServiceProvider instance
Expected Result	- Type hinted instance is accepted and returned.
Actual Result	- Type hinting works as expected in function parameters.
Status	Pass
Severity	High

Test Case ID	ESP-025
Title	Non-finality of EventServiceProvider Class
Objective	Check that EventServiceProvider is not marked as final.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Call isFinal() and assert false.
Test Data	- EventServiceProvider class
Expected Result	- isFinal() returns false.
Actual Result	- EventServiceProvider class is not final.
Status	Pass
Severity	High

Test Case ID	ESP-026
Title	EventServiceProvider Not an Interface
Objective	Validate EventServiceProvider is a class, not an interface.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Call isInterface() and assert false.
Test Data	- EventServiceProvider class
Expected Result	- isInterface() returns false.
Actual Result	- EventServiceProvider is not an interface.
Status	Pass
Severity	High

Test Case ID	ESP-027
Title	EventServiceProvider Not a Trait
Objective	Confirm EventServiceProvider is a class, not a trait.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Call isTrait() and assert false.

Test Data	- EventServiceProvider class
Expected Result	- isTrait() returns false.
Actual Result	- EventServiceProvider is not a trait.
Status	Pass
Severity	High

Test Case ID	ESP-028
Title	Class Loading Verification
Objective	Check EventServiceProvider is loaded in PHP runtime.
Preconditions	- Application running - EventServiceProvider class defined and autoloaded
Test Steps	1. Use class_exists() for EventServiceProvider::class. 2. Assert the result is true.
Test Data	- Class name
Expected Result	- class_exists returns true.
Actual Result	- Class is loaded successfully.
Status	Pass
Severity	High

Test Case ID	ESP-029
Title	Required Classes Usage Verification
Objective	Ensure EventServiceProvider uses essential classes.
Preconditions	- Application running - EventServiceProvider file accessible
Test Steps	1. Read EventServiceProvider class file content. 2. Assert presence of 'use' statements for UpdateFinished, Registered, EventServiceProvider.
Test Data	- EventServiceProvider file
Expected Result	- All required 'use' statements found.
Actual Result	- Required classes are used via import statements.
Status	Pass
Severity	High

Test Case ID	ESP-030
Title	Import of All Version Listeners
Objective	Verify all version listeners are imported by EventServiceProvider.
Preconditions	- Application running - EventServiceProvider file accessible
Test Steps	1. Read EventServiceProvider file content. 2. Assert presence of import statements for Version110, Version200, Version300.
Test Data	- EventServiceProvider file
Expected Result	- Imports exist for all listeners: v1, v2, v3.
Actual Result	- All version listener imports found.
Status	Pass
Severity	High

Test Case ID	ESP-031
Title	File Structure Validation
Objective	Ensure EventServiceProvider file structure includes expected definitions.
Preconditions	- Application running - EventServiceProvider file accessible
Test Steps	1. Read EventServiceProvider file. 2. Assert content includes 'class EventServiceProvider extends ServiceProvider', 'protected $listen', and 'public function boot()'.
Test Data	- EventServiceProvider file
Expected Result	- File contains class header, listen property, boot method.
Actual Result	- Structure as expected is present.
Status	Pass
Severity	High

Test Case ID	ESP-032
Title	Compact Implementation Verification
Objective	Check that EventServiceProvider file size is less than 2000 bytes.
Preconditions	- Application running - EventServiceProvider file accessible
Test Steps	1. Read EventServiceProvider file. 2. Assert strlen(file contents) < 2000.
Test Data	- EventServiceProvider file content
Expected Result	- File size is less than 2000 bytes.
Actual Result	- File meets compact size requirement (<2000 bytes).
Status	Pass
Severity	Medium

Test Case ID	ESP-033
Title	Reasonable Line Count in File
Objective	Ensure implementation file line count is less than 100.
Preconditions	- Application running - EventServiceProvider file accessible
Test Steps	1. Read file contents. 2. Split file into lines, count them. 3. Assert line count < 100.
Test Data	- Number of lines in EventServiceProvider file
Expected Result	- Line count is under 100.
Actual Result	- File has fewer than 100 lines.
Status	Pass
Severity	Medium

Test Case ID	ESP-034
Title	Documentation for Listen Property
Objective	Validate the presence of documentation comments for 'listen' property.
Preconditions	- Application running - EventServiceProvider class loaded

Test Steps	1. Use ReflectionClass for EventServiceProvider. 2. Access 'listen' property and getDocComment(). 3. Assert doc comment exists (not false).
Test Data	- EventServiceProvider class
Expected Result	- 'listen' property has documentation.
Actual Result	- Documentation for 'listen' property is present.
Status	Pass
Severity	Medium

Test Case ID	ESP-035
Title	Documentation for Boot Method
Objective	Verify boot method is documented.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Get 'boot' method and its doc comment. 3. Assert doc comment exists.
Test Data	- EventServiceProvider class
Expected Result	- boot method has documentation.
Actual Result	- boot method documentation found.
Status	Pass
Severity	Medium

Test Case ID	ESP-036
Title	Boot Method Return Type Documentation
Objective	Ensure boot method doc comment includes '@return'.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Get boot method doc comment. 3. Assert doc comment @return exists.
Test Data	- boot method doc comment
Expected Result	- '@return' found in boot method documentation.
Actual Result	- '@return' present in doc comment.
Status	Pass
Severity	Medium

Test Case ID	ESP-037
Title	Listener Version Order Verification
Objective	Confirm UpdateFinished event listeners are ordered by version.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Access listen property mapping for UpdateFinished. 3. Assert that the listeners array has the specified listeners in the correct index order.
Test Data	- List of listeners for UpdateFinished event

Expected Result	- listeners[0]: v1\Version110 - listeners[1]: v2\Version200 - listeners[7]: v3\Version311
Actual Result	- Listener order matches expectation.
Status	Pass
Severity	High

Test Case ID	ESP-038
Title	Exact Number of Event Mappings in Listen Property
Objective	Ensure 'listen' property contains exactly two event mappings.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Access 'listen' property. 3. Assert count($listen) == 2.
Test Data	- EventServiceProvider instance
Expected Result	- 'listen' array has 2 entries.
Actual Result	- 'listen' property has exactly 2 mappings.
Status	Pass
Severity	High

Test Case ID	ESP-039
Title	Event Class Key Verification for Listen Property
Objective	Confirm 'listen' property keys are UpdateFinished and Registered event classes.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Access 'listen' property keys. 3. Assert keys include UpdateFinished::class and Registered::class.
Test Data	- Keys of 'listen' array
Expected Result	- Keys contain both event classes.
Actual Result	- Keys correctly include UpdateFinished and Registered.
Status	Pass
Severity	High

Test Case ID	ESP-040
Title	Parent Class Verification
Objective	Ensure EventServiceProvider’s parent is EventServiceProvider from Laravel.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Use ReflectionClass. 2. Get parent class. 3. Assert parent class name is 'Illuminate\Foundation\Support\Providers\EventServiceProvider'.
Test Data	- EventServiceProvider class definition
Expected Result	- Parent class is EventServiceProvider from Laravel.
Actual Result	- Parent class is correct.
Status	Pass

Severity	High

Test Case ID	ESP-041
Title	Laravel ServiceProvider Inheritance
Objective	Confirm EventServiceProvider inherits from Laravel’s ServiceProvider.
Preconditions	- Application running - EventServiceProvider class loaded
Test Steps	1. Instantiate EventServiceProvider. 2. Assert it is instance of \Illuminate\Support\ServiceProvider.
Test Data	- EventServiceProvider instance
Expected Result	- Instance is of type \Illuminate\Support\ServiceProvider.
Actual Result	- Inheritance from Laravel ServiceProvider confirmed.
Status	Pass
Severity	High

File: ExchangeRateLog-Test.txt

Test Case ID	ERL-001
Title	Instantiate ExchangeRateLog
Objective	Verify that the ExchangeRateLog model can be instantiated.
Preconditions	- Application running - ExchangeRateLog class available
Test Steps	1. Instantiate a new ExchangeRateLog object.
Test Data	- None
Expected Result	- The created object is an instance of ExchangeRateLog.
Actual Result	Object of ExchangeRateLog is created and confirmed to be instance of correct class.
Status	Pass
Severity	Medium

Test Case ID	ERL-002
Title	ExchangeRateLog extends Model
Objective	Verify that ExchangeRateLog is a subclass of Illuminate\Database\Eloquent\Model.
Preconditions	- Application running - ExchangeRateLog and Model classes available
Test Steps	1. Instantiate ExchangeRateLog. 2. Check if object is instance of Model.
Test Data	- None
Expected Result	- ExchangeRateLog object is an instance of Illuminate\Database\Eloquent\Model.
Actual Result	ExchangeRateLog is confirmed as subtype of Model.
Status	Pass
Severity	Medium

Test Case ID	ERL-003
Title	Correct Namespace for ExchangeRateLog
Objective	Verify that ExchangeRateLog is declared under the "Crater\Models" namespace.
Preconditions	- Application running - ExchangeRateLog class available
Test Steps	1. Reflect ExchangeRateLog class. 2. Retrieve its namespace.
Test Data	- None
Expected Result	- Namespace is "Crater\Models".
Actual Result	Namespace identified as "Crater\Models".
Status	Pass
Severity	Low

Test Case ID	ERL-004
Title	ExchangeRateLog is not abstract
Objective	Ensure that ExchangeRateLog is not an abstract class.

Preconditions	- Application running - ExchangeRateLog class available
Test Steps	1. Reflect ExchangeRateLog class. 2. Check if it is abstract.
Test Data	- None
Expected Result	- isAbstract returns false.
Actual Result	ExchangeRateLog is confirmed to be non-abstract.
Status	Pass
Severity	Medium

Test Case ID	ERL-005
Title	ExchangeRateLog is instantiable
Objective	Verify that ExchangeRateLog class is instantiable.
Preconditions	- Application running - ExchangeRateLog class available
Test Steps	1. Reflect ExchangeRateLog class. 2. Check if isInstantiable returns true.
Test Data	- None
Expected Result	- isInstantiable returns true.
Actual Result	Class is instantiable.
Status	Pass
Severity	Medium

Test Case ID	ERL-006
Title	Guarded Properties Existence
Objective	Ensure ExchangeRateLog declares guarded properties.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Retrieve guarded properties.
Test Data	- None
Expected Result	- getGuarded returns ['id'].
Actual Result	Guarded properties are ['id'].
Status	Pass
Severity	Medium

Test Case ID	ERL-007
Title	Guarded Property prevents 'id' Mass Assignment
Objective	Ensure 'id' field is listed and prevented from mass assignment.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Retrieve guarded array.
Test Data	- None
Expected Result	- Guarded array contains 'id'.
Actual Result	'id' is present in guarded array.
Status	Pass
Severity	High

Test Case ID	ERL-008
Title	'id' Cannot be Mass Assigned
Objective	Ensure 'id' is not settable via mass assignment.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Use fill(['id' => 999, 'exchange_rate' => 1.5]). 3. Check log->id and log->exchange_rate.
Test Data	- fill: ['id' => 999, 'exchange_rate' => 1.5]
Expected Result	- log->id is null. - log->exchange_rate is 1.5.
Actual Result	'id' remains null; 'exchange_rate' set to 1.5.
Status	Pass
Severity	High

Test Case ID	ERL-009
Title	exchange_rate cast to float
Objective	Ensure exchange_rate attribute casts to float data type.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = 1.23. 3. Check type and value of exchange_rate.
Test Data	- exchange_rate = 1.23
Expected Result	- exchange_rate is float and equals 1.23.
Actual Result	exchange_rate is float and equals 1.23.
Status	Pass
Severity	High

Test Case ID	ERL-010
Title	exchange_rate casts string to float
Objective	Validate string value assignment is cast to float for exchange_rate.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = "1.5". 3. Check type and value.
Test Data	- exchange_rate = "1.5"
Expected Result	- exchange_rate is float and equals 1.5.
Actual Result	exchange_rate correctly cast to 1.5 float.
Status	Pass
Severity	High

Test Case ID	ERL-011
Title	exchange_rate casts integer to float
Objective	Ensure integer assignment is cast to float for exchange_rate.
Preconditions	- Application running

Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = 2. 3. Check type and value.
Test Data	- exchange_rate = 2
Expected Result	- exchange_rate is float and equals 2.0.
Actual Result	exchange_rate is float and 2.0.
Status	Pass
Severity	Medium

Test Case ID	ERL-012
Title	exchange_rate handles null assignment
Objective	Validate proper behavior when exchange_rate is set to null.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = null. 3. Check attribute value.
Test Data	- exchange_rate = null
Expected Result	- exchange_rate is null.
Actual Result	exchange_rate is null.
Status	Pass
Severity	Medium

Test Case ID	ERL-013
Title	exchange_rate casts invalid string to zero
Objective	Check that an invalid string assigned to exchange_rate is cast to float zero.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = "invalid". 3. Check type and value.
Test Data	- exchange_rate = "invalid"
Expected Result	- exchange_rate is float and equals 0.0.
Actual Result	exchange_rate is float and 0.0.
Status	Pass
Severity	Medium

Test Case ID	ERL-014
Title	currency method existence
Objective	Confirm that ExchangeRateLog has a currency method.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Verify currency method exists.
Test Data	- None
Expected Result	- currency method exists on ExchangeRateLog.
Actual Result	Method 'currency' found.
Status	Pass
Severity	Medium

Test Case ID	ERL-015
Title	currency relationship returns BelongsTo
Objective	Ensure currency method returns a BelongsTo relation.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Call currency(). 3. Check returned type.
Test Data	- None
Expected Result	- Returns instance of BelongsTo.
Actual Result	currency returns BelongsTo.
Status	Pass
Severity	Medium

Test Case ID	ERL-016
Title	currency relationship targets Currency model
Objective	Ensure relationship is to Currency model.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Call currency() and get related. 3. Check related model type.
Test Data	- None
Expected Result	- Related is instance of Currency.
Actual Result	Related model is Currency.
Status	Pass
Severity	Medium

Test Case ID	ERL-017
Title	currency relationship uses currency_id foreign key
Objective	Confirm currency relationship uses currency_id as foreign key.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Call currency(). 3. Get foreign key name.
Test Data	- None
Expected Result	- Foreign key is 'currency_id'.
Actual Result	Foreign key name is currency_id.
Status	Pass
Severity	High

Test Case ID	ERL-018
Title	currency relationship uses id as owner key
Objective	Confirm owner key for currency relation is 'id'.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Call currency(). 3. Get owner key name.
Test Data	- None

Expected Result	- Owner key name is 'id'.
Actual Result	Owner key confirmed as 'id'.
Status	Pass
Severity	High

Test Case ID	ERL-019
Title	company method existence
Objective	Confirm the presence of company method in ExchangeRateLog.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Verify company method existence.
Test Data	- None
Expected Result	- company method exists.
Actual Result	Method 'company' found.
Status	Pass
Severity	Medium

Test Case ID	ERL-020
Title	company relationship returns BelongsTo
Objective	Validate company method returns BelongsTo relation.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Call company(). 3. Check relation type.
Test Data	- None
Expected Result	- Returns BelongsTo instance.
Actual Result	company returns BelongsTo relation.
Status	Pass
Severity	Medium

Test Case ID	ERL-021
Title	company relationship targets Company model
Objective	Ensure company relation is defined to Company.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Call company() and get related. 3. Verify related model type.
Test Data	- None
Expected Result	- Related is Company model.
Actual Result	Related is Company.
Status	Pass
Severity	Medium

Test Case ID	ERL-022
Title	company relationship uses company_id foreign key
Objective	Confirm company relationship foreign key is company_id.

Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Call company(). 3. Get foreign key.
Test Data	- None
Expected Result	- Foreign key is 'company_id'.
Actual Result	Foreign key set as 'company_id'.
Status	Pass
Severity	High

Test Case ID	ERL-023
Title	company relationship uses id as owner key
Objective	Confirm owner key for company relation is 'id'.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Call company(). 3. Get owner key.
Test Data	- None
Expected Result	- Owner key is 'id'.
Actual Result	Owner key confirmed as 'id'.
Status	Pass
Severity	High

Test Case ID	ERL-024
Title	addExchangeRateLog method existence
Objective	Ensure addExchangeRateLog static method exists in ExchangeRateLog.
Preconditions	- Application running
Test Steps	1. Check class methods for addExchangeRateLog.
Test Data	- None
Expected Result	- Method addExchangeRateLog exists.
Actual Result	Method exists.
Status	Pass
Severity	High

Test Case ID	ERL-025
Title	addExchangeRateLog is static
Objective	Confirm addExchangeRateLog is a static method.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Get addExchangeRateLog method. 3. Check method isStatic flag.
Test Data	- None
Expected Result	- isStatic true.
Actual Result	Method is static.
Status	Pass
Severity	High

Test Case ID	ERL-026
Title	addExchangeRateLog is public
Objective	Verify addExchangeRateLog method is publicly accessible.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Get addExchangeRateLog method. 3. Check isPublic.
Test Data	- None
Expected Result	- isPublic true.
Actual Result	Method is public.
Status	Pass
Severity	High

Test Case ID	ERL-027
Title	addExchangeRateLog accepts one parameter
Objective	Ensure method signature matches requirement for one parameter.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Get addExchangeRateLog method. 3. Check number of parameters.
Test Data	- None
Expected Result	- Number of parameters is 1.
Actual Result	Method signature has one parameter.
Status	Pass
Severity	High

Test Case ID	ERL-028
Title	Uses HasFactory trait
Objective	Confirm ExchangeRateLog uses Illuminate\Database\Eloquent\Factories\HasFactory trait.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Get trait names.
Test Data	- None
Expected Result	- Traits collection contains HasFactory.
Actual Result	HasFactory trait found.
Status	Pass
Severity	Medium

Test Case ID	ERL-029
Title	Multiple instances can be created
Objective	Validate independent creation of multiple ExchangeRateLog objects.
Preconditions	- Application running
Test Steps	1. Create $log1 and $log2. 2. Confirm both are ExchangeRateLog instances. 3. Confirm objects are not the same.

Test Data	- None
Expected Result	- Both objects are ExchangeRateLog and not identical.
Actual Result	Multiple independent instances confirmed.
Status	Pass
Severity	Medium

Test Case ID	ERL-030
Title	ExchangeRateLog cloning
Objective	Confirm ExchangeRateLog objects can be cloned.
Preconditions	- Application running
Test Steps	1. Create ExchangeRateLog with exchange_rate 1.5. 2. Clone object. 3. Confirm clone is separate and has same exchange_rate.
Test Data	- exchange_rate = 1.5
Expected Result	- Cloned object is ExchangeRateLog, distinct from original, same exchange_rate value.
Actual Result	Cloned successfully, values match, objects distinct.
Status	Pass
Severity	Medium

Test Case ID	ERL-031
Title	Can use ExchangeRateLog in type hints
Objective	Validate type hinting with ExchangeRateLog object.
Preconditions	- Application running
Test Steps	1. Define function with ExchangeRateLog parameter. 2. Pass log instance to function. 3. Confirm returned object matches input.
Test Data	- None
Expected Result	- Function receives and returns correct object.
Actual Result	Type-hinting works, returned as input.
Status	Pass
Severity	Medium

Test Case ID	ERL-032
Title	ExchangeRateLog is not final
Objective	Confirm the class is not declared final.
Preconditions	- Application running
Test Steps	1. Reflect class. 2. Check isFinal.
Test Data	- None
Expected Result	- isFinal false.
Actual Result	Class is not final.
Status	Pass
Severity	Low

Test Case ID	ERL-033

Title	ExchangeRateLog is not an interface
Objective	Validate ExchangeRateLog is not an interface declaration.
Preconditions	- Application running
Test Steps	1. Reflect class. 2. Check isInterface.
Test Data	- None
Expected Result	- isInterface false.
Actual Result	Not an interface.
Status	Pass
Severity	Low

Test Case ID	ERL-034
Title	ExchangeRateLog is not a trait
Objective	Confirm ExchangeRateLog is a class, not a trait.
Preconditions	- Application running
Test Steps	1. Reflect class. 2. Check isTrait.
Test Data	- None
Expected Result	- isTrait false.
Actual Result	Not a trait.
Status	Pass
Severity	Low

Test Case ID	ERL-035
Title	ExchangeRateLog class is loaded
Objective	Validate class existence and is loaded at runtime.
Preconditions	- Application running
Test Steps	1. Check class_exists for ExchangeRateLog.
Test Data	- None
Expected Result	- class_exists returns true.
Actual Result	Class loaded and available.
Status	Pass
Severity	Medium

Test Case ID	ERL-036
Title	Required Classes Imported
Objective	Ensure necessary imports are present in ExchangeRateLog file.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Read file contents. 3. Check for class imports.
Test Data	- None
Expected Result	- Imports for HasFactory and Model exist.
Actual Result	Imports confirmed.
Status	Pass

Severity	Medium

Test Case ID	ERL-037
Title	File Structure Contains Expected Elements
Objective	Validate key structural parts exist in ExchangeRateLog file.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Obtain file contents. 3. Check for class definition, guarded, casts.
Test Data	- None
Expected Result	- class ExchangeRateLog extends Model present - protected $guarded present - protected $casts present
Actual Result	All expected structure found.
Status	Pass
Severity	Low

Test Case ID	ERL-038
Title	File Size is Compact
Objective	Confirm file size under 1500 bytes for compact implementation.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Get file content length.
Test Data	- None
Expected Result	- File size < 1500 bytes.
Actual Result	File compact, less than 1500 bytes.
Status	Pass
Severity	Low

Test Case ID	ERL-039
Title	Minimal Lines in File
Objective	Ensure code contains less than 60 lines.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Count line breaks.
Test Data	- None
Expected Result	- Line count < 60.
Actual Result	Line count meets minimum requirement.
Status	Pass
Severity	Low

Test Case ID	ERL-040
Title	All Relationship Methods are Public
Objective	Validate public accessibility of relationship methods currency and company.
Preconditions	- Application running

Test Steps	1. Reflect ExchangeRateLog. 2. Get method info for currency and company. 3. Check both are public.
Test Data	- None
Expected Result	- Both methods are public.
Actual Result	Both confirmed as public.
Status	Pass
Severity	High

Test Case ID	ERL-041
Title	All Relationship Methods are not Static
Objective	Ensure currency and company methods are non-static.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Check isStatic on currency and company.
Test Data	- None
Expected Result	- isStatic is false for both.
Actual Result	Neither method is static.
Status	Pass
Severity	High

Test Case ID	ERL-042
Title	Relationship Methods Have No Parameters
Objective	Validate currency and company relations have zero parameters.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Get number of parameters for currency and company.
Test Data	- None
Expected Result	- Both methods have zero parameters.
Actual Result	No parameters for relationships.
Status	Pass
Severity	Medium

Test Case ID	ERL-043
Title	Set and Get exchange_rate Attribute
Objective	Validate proper get/set of exchange_rate on model instance.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = 1.75. 3. Retrieve exchange_rate.
Test Data	- exchange_rate = 1.75
Expected Result	- exchange_rate is 1.75.
Actual Result	Set and get successful.
Status	Pass
Severity	High

Test Case ID	ERL-044
Title	Set and Get company_id Attribute
Objective	Validate ability to set/get company_id.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set company_id = 5. 3. Retrieve company_id.
Test Data	- company_id = 5
Expected Result	- company_id is 5.
Actual Result	company_id correctly set and retrieved.
Status	Pass
Severity	Medium

Test Case ID	ERL-045
Title	Set and Get currency_id Attribute
Objective	Validate ability to set/get currency_id.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set currency_id = 10. 3. Retrieve currency_id.
Test Data	- currency_id = 10
Expected Result	- currency_id is 10.
Actual Result	currency_id set to 10.
Status	Pass
Severity	Medium

Test Case ID	ERL-046
Title	Set and Get base_currency_id Attribute
Objective	Validate ability to set/get base_currency_id.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set base_currency_id = 15. 3. Retrieve base_currency_id.
Test Data	- base_currency_id = 15
Expected Result	- base_currency_id is 15.
Actual Result	base_currency_id correctly set.
Status	Pass
Severity	Medium

Test Case ID	ERL-047
Title	Casts Configured Properly
Objective	Validate model casts array includes float for exchange_rate.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Get casts array.
Test Data	- None
Expected Result	- 'exchange_rate' key exists and is 'float'.

Actual Result	exchange_rate cast configuration present.
Status	Pass
Severity	Medium

Test Case ID	ERL-048
Title	Decimal String Casts Properly to Float
Objective	Validate casting decimal string for exchange_rate.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = "1.234567". 3. Check type and value.
Test Data	- exchange_rate = "1.234567"
Expected Result	- exchange_rate is float and 1.234567.
Actual Result	exchange_rate correctly cast as float.
Status	Pass
Severity	High

Test Case ID	ERL-049
Title	Negative values cast to float
Objective	Validate negative values cast correctly for exchange_rate.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = -0.5. 3. Check type and value.
Test Data	- exchange_rate = -0.5
Expected Result	- exchange_rate is float and equals -0.5.
Actual Result	Negative value cast correctly.
Status	Pass
Severity	High

Test Case ID	ERL-050
Title	Zero value handled correctly
Objective	Validate zero assignment for exchange_rate.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = 0. 3. Check value.
Test Data	- exchange_rate = 0
Expected Result	- exchange_rate is 0.0.
Actual Result	exchange_rate is 0.0.
Status	Pass
Severity	High

Test Case ID	ERL-051
Title	Very small float value is handled
Objective	Validate precision handling of small float for exchange_rate.

Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = 0.0001. 3. Check type and value.
Test Data	- exchange_rate = 0.0001
Expected Result	- exchange_rate is float and equals 0.0001.
Actual Result	Small value preserved.
Status	Pass
Severity	Medium

Test Case ID	ERL-052
Title	Very large float value is handled
Objective	Validate handling large float assignments for exchange_rate.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Set exchange_rate = 999999.99. 3. Check type and value.
Test Data	- exchange_rate = 999999.99
Expected Result	- exchange_rate is float and equals 999999.99.
Actual Result	Large value assigned correctly.
Status	Pass
Severity	Medium

Test Case ID	ERL-053
Title	addExchangeRateLog uses CompanySetting
Objective	Confirm CompanySetting usage in addExchangeRateLog.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Read file contents for CompanySetting usage.
Test Data	- None
Expected Result	- file contains 'CompanySetting::getSetting'
Actual Result	CompanySetting usage found.
Status	Pass
Severity	High

Test Case ID	ERL-054
Title	addExchangeRateLog calls create
Objective	Ensure addExchangeRateLog calls self::create internally.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Search file for 'self::create'.
Test Data	- None
Expected Result	- File contains 'self::create'.
Actual Result	Method calls 'self::create'.
Status	Pass
Severity	High

Test Case ID	ERL-055
Title	addExchangeRateLog maps model properties
Objective	Validate property mapping inside addExchangeRateLog.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Read file content for mapping of exchange_rate, company_id, base_currency_id, currency_id.
Test Data	- None
Expected Result	- Mapping for four properties present.
Actual Result	All model properties mapped.
Status	Pass
Severity	High

Test Case ID	ERL-056
Title	addExchangeRateLog uses model parameter properties
Objective	Confirm use of $model parameter properties inside addExchangeRateLog.
Preconditions	- Application running
Test Steps	1. Reflect ExchangeRateLog. 2. Read file content for $model->exchange_rate, $model->company_id, $model->currency_id.
Test Data	- None
Expected Result	- Use of all mentioned properties found.
Actual Result	All required $model properties used.
Status	Pass
Severity	High

Test Case ID	ERL-057
Title	Data preserved through constructor
Objective	Confirm constructor attribute assignment is correctly stored.
Preconditions	- Application running
Test Steps	1. Define $data array with exchange_rate, company_id, currency_id, base_currency_id. 2. Instantiate ExchangeRateLog with $data. 3. Validate attribute assignments.
Test Data	- $data = ['exchange_rate' => 1.5, 'company_id' => 1, 'currency_id' => 2, 'base_currency_id' => 3]
Expected Result	- exchange_rate = 1.5 - company_id = 1 - currency_id = 2 - base_currency_id = 3
Actual Result	All constructor properties are assigned correctly.
Status	Pass
Severity	High

Test Case ID	ERL-058
Title	Data independence in instances
Objective	Confirm multiple instances manage independent data.

Preconditions	- Application running
Test Steps	1. Instantiate log1 with exchange_rate 1.0. 2. Instantiate log2 with exchange_rate 2.0. 3. Confirm properties are independent.
Test Data	- log1: exchange_rate = 1.0 - log2: exchange_rate = 2.0
Expected Result	- log1 not equal to log2 - log1 exchange_rate = 1.0 - log2 exchange_rate = 2.0
Actual Result	Instances are independent and data is isolated.
Status	Pass
Severity	High

Test Case ID	ERL-059
Title	Inherited methods from Model
Objective	Validate ExchangeRateLog inherits Model methods: save, fill, toArray.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Check for method existence.
Test Data	- None
Expected Result	- Methods save, fill, toArray exist.
Actual Result	All inherited methods present.
Status	Pass
Severity	High

Test Case ID	ERL-060
Title	Can use Model features
Objective	Validate Model features are usable on ExchangeRateLog.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateLog. 2. Check method is_callable for fill and toArray.
Test Data	- None
Expected Result	- fill and toArray are callable.
Actual Result	Model features usable.
Status	Pass
Severity	High

File: ExchangeRateLogCollectionAndResource-Test.txt

Test Case ID	ERLCAR-001
Title	Instantiation of ExchangeRateLogCollection
Objective	Verify that ExchangeRateLogCollection can be successfully instantiated.
Preconditions	- Application running - ExchangeRateLogCollection and dependencies are autoloaded
Test Steps	1. Create a new ExchangeRateLogCollection with an empty Collection. 2. Assert the type of the created object.
Test Data	- Input: new ExchangeRateLogCollection(new Collection([])) - Expected value: Instance of ExchangeRateLogCollection
Expected Result	- The instantiated object is an instance of ExchangeRateLogCollection.
Actual Result	- The object is successfully created as an instance of ExchangeRateLogCollection.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-002
Title	Inheritance from ResourceCollection
Objective	Ensure ExchangeRateLogCollection extends ResourceCollection.
Preconditions	- Application running - ExchangeRateLogCollection and ResourceCollection classes available
Test Steps	1. Create a new ExchangeRateLogCollection with an empty Collection. 2. Assert that the object is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Test Data	- Input: new ExchangeRateLogCollection(new Collection([])) - Expected value: Instance of ResourceCollection
Expected Result	- The object is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	- The object is an instance of ResourceCollection as expected.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-003
Title	Namespace Validation
Objective	Confirm ExchangeRateLogCollection is in the correct namespace.
Preconditions	- Application running - ExchangeRateLogCollection class available
Test Steps	1. Reflect on the ExchangeRateLogCollection class. 2. Retrieve the namespace name. 3. Compare namespace with expected value.
Test Data	- Input: ReflectionClass(ExchangeRateLogCollection::class) - Expected value: Namespace is 'Crater\Http\Resources'
Expected Result	- The namespace of ExchangeRateLogCollection is 'Crater\Http\Resources'.
Actual Result	- Namespace is correctly set to 'Crater\Http\Resources'.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-004

Title	Availability of toArray Method
Objective	Verify ExchangeRateLogCollection has a public toArray method implemented.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Instantiate ExchangeRateLogCollection. 2. Use method_exists to check for 'toArray'.
Test Data	- Input: method_exists($collection, 'toArray') - Expected value: true
Expected Result	- The class has a method named toArray.
Actual Result	- The method toArray exists.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-005
Title	Public Accessibility of toArray Method
Objective	Ensure the toArray method in ExchangeRateLogCollection is public.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Retrieve the toArray method details. 3. Assert the access modifier is public.
Test Data	- Input: $method->isPublic() - Expected value: true
Expected Result	- The toArray method is public.
Actual Result	- The toArray method is public.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-006
Title	toArray Method Parameter Validation
Objective	Verify that the toArray method accepts one parameter named 'request'.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Retrieve parameters for the toArray method. 3. Validate count and name of parameter.
Test Data	- Input: $parameters = $method->getParameters() - Expected value: 1 parameter named 'request'
Expected Result	- The toArray method accepts 1 parameter with the name 'request'.
Actual Result	- The toArray method accepts a parameter named 'request'.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-007
Title	Handling Empty Collections
Objective	Verify the toArray method handles empty collections gracefully.
Preconditions	- Application running - ExchangeRateLogCollection and Request classes loaded

Test Steps	1. Instantiate Request and ExchangeRateLogCollection with an empty collection. 2. Call toArray with the Request. 3. Assert returned array is empty.
Test Data	- Input: new Collection([]), new Request() - Expected value: []
Expected Result	- The result is an empty array.
Actual Result	- The toArray method returns an empty array.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-008
Title	Transformation of Single Exchange Rate Resource
Objective	Verify toArray method transforms a single ExchangeRateLogResource correctly.
Preconditions	- Application running - ExchangeRateLogResource and ExchangeRateLogCollection classes loaded
Test Steps	1. Create a single dummy exchange rate log entry (id=1, exchange_rate=1.5). 2. Wrap it in ExchangeRateLogResource. 3. Pass into ExchangeRateLogCollection, call toArray. 4. Assert the output is an array of count 1 with correct id.
Test Data	- Input: $log = createDummyExchangeRateLog(1, 1.5) - Expected value: array with ['id' => 1]
Expected Result	- Output array count is 1. - Output contains key 'id' with value 1.
Actual Result	- Output is array of 1 item with 'id' = 1.
Status	Pass
Severity	High

Test Case ID	ERLCAR-009
Title	Transformation of Multiple Exchange Rate Resources
Objective	Validate toArray method correctly transforms multiple ExchangeRateLogResource objects.
Preconditions	- Application running - ExchangeRateLogResource and ExchangeRateLogCollection classes loaded
Test Steps	1. Create two dummy exchange rate logs (id=1, rate=1.5; id=2, rate=2.0). 2. Wrap each log in ExchangeRateLogResource. 3. Pass both into ExchangeRateLogCollection, call toArray. 4. Assert the output is array of count 2 with correct ids.
Test Data	- Input: - Log1: id=1, exchange_rate=1.5 - Log2: id=2, exchange_rate=2.0 - Expected value: Array with first item id=1, second item id=2
Expected Result	- Output array count is 2. - Output[0]['id'] = 1 - Output[1]['id'] = 2
Actual Result	- Output is array with two items with correct ids.
Status	Pass
Severity	High

Test Case ID	ERLCAR-010
Title	Required Fields in Transformed Items
Objective	Ensure each transformed item contains all required fields.
Preconditions	- Application running - ExchangeRateLogResource and ExchangeRateLogCollection classes loaded
Test Steps	1. Create dummy exchange rate log. 2. Wrap in ExchangeRateLogResource. 3. Pass into ExchangeRateLogCollection, call toArray. 4. Assert presence of keys: 'id', 'exchange_rate', 'company_id', 'currency_id', 'base_currency_id'.
Test Data	- Input: id=1, exchange_rate=1.5 - Expected keys: id, exchange_rate, company_id, currency_id, base_currency_id
Expected Result	- Transformed item contains required keys.
Actual Result	- Output contains all required fields.
Status	Pass
Severity	High

Test Case ID	ERLCAR-011
Title	Handling Large Collections
Objective	Validate proper transformation and performance with large exchange rate collections.
Preconditions	- Application running - ExchangeRateLogResource, ExchangeRateLogCollection, and Request classes loaded
Test Steps	1. Create 50 dummy exchange rate log resources (ids 1-50, exchange_rate increments). 2. Pass all into ExchangeRateLogCollection, call toArray. 3. Assert output array count is 50. 4. Verify first item id=1, last item id=50.
Test Data	- Input: Resources with id=1..50, exchange_rate=i*0.1 - Expected values: array count=50, first item id=1, last item id=50
Expected Result	- Output is array of 50 items with correct ids.
Actual Result	- Correct count and id values in output.
Status	Pass
Severity	High

Test Case ID	ERLCAR-012
Title	toArray Delegation to Parent
Objective	Confirm toArray method in ExchangeRateLogCollection delegates to parent ResourceCollection.
Preconditions	- Application running - ExchangeRateLogCollection class source accessible
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Read source file. 3. Search for 'parent::toArray' in file content.
Test Data	- Source file content - Expected substring: 'parent::toArray'
Expected Result	- Source contains 'parent::toArray'.
Actual Result	- Delegation to parent method is present.

Status	Pass
Severity	Medium

Test Case ID	ERLCAR-013
Title	Parent Class Verification
Objective	Verify ExchangeRateLogCollection's parent class is ResourceCollection.
Preconditions	- Application running - ExchangeRateLogCollection and parent class available
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Retrieve parent class info. 3. Assert parent class is Illuminate\Http\Resources\Json\ResourceCollection.
Test Data	- Input: $reflection->getParentClass() - Expected value: ResourceCollection
Expected Result	- Parent class is ResourceCollection.
Actual Result	- Parent class verified as ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-014
Title	Multiple Instance Creation
Objective	Validate ability to create multiple independent ExchangeRateLogCollection instances.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Create two instances of ExchangeRateLogCollection. 2. Assert both are instances of ExchangeRateLogCollection. 3. Assert instances are not the same object.
Test Data	- Input: new ExchangeRateLogCollection(new Collection([])) - Expected values: Two distinct instances.
Expected Result	- Both objects are instances of ExchangeRateLogCollection and are not identical.
Actual Result	- Multiple distinct instances were successfully created.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-015
Title	Object Cloning
Objective	Test that ExchangeRateLogCollection instance can be cloned properly.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Create an ExchangeRateLogCollection instance. 2. Clone the instance. 3. Assert clone is instance of ExchangeRateLogCollection. 4. Assert original and clone are not the same object.
Test Data	- Input: $clone = clone $collection - Expected values: $clone is new and same type
Expected Result	- Clone is a valid object and distinct from original.
Actual Result	- Cloning operation works; cloned object is valid.
Status	Pass

Severity	Low

Test Case ID	ERLCAR-016
Title	Type Hint Compatibility
Objective	Ensure ExchangeRateLogCollection supports type hinting in functions.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Define a function that type-hints ExchangeRateLogCollection. 2. Pass an instance of ExchangeRateLogCollection to function. 3. Assert returned value matches input.
Test Data	- Input: function (ExchangeRateLogCollection $collection) - Expected value: Passed object is returned
Expected Result	- Type-hint is supported; input and output are equal.
Actual Result	- Type hinting works as expected.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-017
Title	Class is Not Abstract
Objective	Verify ExchangeRateLogCollection class is concrete, not abstract.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Check isAbstract property.
Test Data	- Input: $reflection->isAbstract() - Expected value: false
Expected Result	- Class is not abstract.
Actual Result	- Class confirmed as concrete.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-018
Title	Class is Not Final
Objective	Confirm ExchangeRateLogCollection is not a final class.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Check isFinal property.
Test Data	- Input: $reflection->isFinal() - Expected value: false
Expected Result	- Class is not declared as final.
Actual Result	- Class is not final.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-019
Title	Class Is Not Interface
Objective	Validate ExchangeRateLogCollection is not implemented as an interface.

Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Check isInterface property.
Test Data	- Input: $reflection->isInterface() - Expected value: false
Expected Result	- Class is not an interface.
Actual Result	- Class is not implemented as interface.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-020
Title	Class Is Not Trait
Objective	Ensure ExchangeRateLogCollection class is not a trait.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Check isTrait property.
Test Data	- Input: $reflection->isTrait() - Expected value: false
Expected Result	- Class is not a trait.
Actual Result	- Class is not a trait.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-021
Title	Class Load Verification
Objective	Confirm ExchangeRateLogCollection class is defined and loaded.
Preconditions	- Application running
Test Steps	1. Use class_exists to check for ExchangeRateLogCollection.
Test Data	- Input: class_exists(ExchangeRateLogCollection::class) - Expected value: true
Expected Result	- Class is loaded.
Actual Result	- Class exists as expected.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-022
Title	ResourceCollection Import Usage
Objective	Verify correct import statement for ResourceCollection in ExchangeRateLogCollection file.
Preconditions	- Application running - Source file accessible
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Get file contents. 3. Search for 'use Illuminate\Http\Resources\Json\ResourceCollection'.
Test Data	- Source file content - Expected substring: 'use Illuminate\Http\Resources\Json\ResourceCollection'

Expected Result	- File contains correct import statement.
Actual Result	- Import statement found in source.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-023
Title	toArray Method Not Static
Objective	Ensure toArray method on ExchangeRateLogCollection is not static.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Retrieve toArray method. 3. Check isStatic property.
Test Data	- Input: $method->isStatic() - Expected value: false
Expected Result	- toArray is not static.
Actual Result	- Method confirmed as non-static.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-024
Title	toArray Method Not Abstract
Objective	Confirm toArray method is a concrete method.
Preconditions	- Application running - ExchangeRateLogCollection class loaded
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Retrieve toArray method. 3. Check isAbstract property.
Test Data	- Input: $method->isAbstract() - Expected value: false
Expected Result	- toArray is concrete.
Actual Result	- Method is not abstract.
Status	Pass
Severity	Medium

Test Case ID	ERLCAR-025
Title	Exchange Rate Data Integrity
Objective	Ensure exchange rate data is preserved through transformation.
Preconditions	- Application running - ExchangeRateLogResource, ExchangeRateLogCollection, and Request classes loaded
Test Steps	1. Create a dummy exchange rate log entry (id=42, exchange_rate=3.14). 2. Wrap in ExchangeRateLogResource. 3. Pass into ExchangeRateLogCollection, call toArray. 4. Assert output id=42 and exchange_rate=3.14.
Test Data	- Input: id=42, exchange_rate=3.14 - Expected values: id=42, exchange_rate=3.14
Expected Result	- Transformed output matches original data.
Actual Result	- Data integrity preserved; output values are correct.

Status	Pass
Severity	High

Test Case ID	ERLCAR-026
Title	Multiple Exchange Rates Handling
Objective	Validate correct handling of different exchange rates.
Preconditions	- Application running - ExchangeRateLogResource, ExchangeRateLogCollection, and Request classes loaded
Test Steps	1. Create two dummy logs (id=1, rate=1.0; id=2, rate=2.5). 2. Wrap each in ExchangeRateLogResource. 3. Pass both into ExchangeRateLogCollection, call toArray. 4. Assert the output array contains correct exchange_rate values.
Test Data	- Input: [(id=1, rate=1.0), (id=2, rate=2.5)] - Expected values: output[0]['exchange_rate']=1.0, output[1]['exchange_rate']=2.5
Expected Result	- Each item retains its correct exchange_rate value.
Actual Result	- Exchange rates in the output are as expected.
Status	Pass
Severity	High

Test Case ID	ERLCAR-027
Title	Source File Size Verification
Objective	Ensure ExchangeRateLogCollection file is concise (<1000 bytes).
Preconditions	- Application running - Source file accessible
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Get file content. 3. Check file size in bytes.
Test Data	- Input: file size - Expected value: <1000 bytes
Expected Result	- File size is less than 1000 bytes.
Actual Result	- Source file is under the limit.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-028
Title	Minimal Line Count Verification
Objective	Ensure source file is minimal and maintainable (<30 lines).
Preconditions	- Application running - Source file accessible
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Get file content. 3. Count lines in file.
Test Data	- Input: line count - Expected value: <30
Expected Result	- File has minimal line count.
Actual Result	- Source file has less than 30 lines.
Status	Pass

Severity	Low

Test Case ID	ERLCAR-029
Title	Comprehensive Field Transformation
Objective	Validate that transformed items include all defined fields.
Preconditions	- Application running - ExchangeRateLogResource, ExchangeRateLogCollection, and Request classes loaded
Test Steps	1. Create a dummy exchange rate log. 2. Wrap in ExchangeRateLogResource. 3. Pass to ExchangeRateLogCollection and call toArray. 4. Assert transformed item includes: id, exchange_rate, company_id, currency_id, base_currency_id.
Test Data	- Input: id=1, exchange_rate=1.5 - Expected keys: id, exchange_rate, company_id, currency_id, base_currency_id
Expected Result	- Transformed output contains all listed fields.
Actual Result	- All required fields are present.
Status	Pass
Severity	High

Test Case ID	ERLCAR-030
Title	Valid Array Structure Verification
Objective	Verify that the transformed collection is a valid PHP array.
Preconditions	- Application running - ExchangeRateLogResource, ExchangeRateLogCollection, and Request classes loaded
Test Steps	1. Create dummy exchange rate log. 2. Wrap in ExchangeRateLogResource. 3. Pass to ExchangeRateLogCollection, call toArray. 4. Assert output is an array and is_array returns true.
Test Data	- Input: id=1, exchange_rate=1.5 - Expected value: array structure
Expected Result	- Output structure is a valid PHP array.
Actual Result	- Output is a valid array.
Status	Pass
Severity	High

Test Case ID	ERLCAR-031
Title	Documentation Presence for toArray Method
Objective	Validate that the toArray method has a docblock.
Preconditions	- Application running - Source file accessible
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Retrieve the toArray method. 3. Get the doc comment and verify it exists.
Test Data	- Input: $method->getDocComment() - Expected value: docblock present
Expected Result	- toArray method has a docblock.
Actual Result	- Documentation is present.

Status	Pass
Severity	Low

Test Case ID	ERLCAR-032
Title	Return Type Documentation in toArray Docblock
Objective	Check that toArray method docblock includes '@return' annotation.
Preconditions	- Application running - Source file accessible
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Retrieve the toArray method docblock. 3. Assert the existence of '@return' in docblock.
Test Data	- Input: docblock string - Expected value: '@return' present
Expected Result	- '@return' annotation found in docblock.
Actual Result	- Docblock contains '@return'.
Status	Pass
Severity	Low

Test Case ID	ERLCAR-033
Title	Parameter Documentation in toArray Docblock
Objective	Ensure '@param' annotation exists in documentation for toArray method.
Preconditions	- Application running - Source file accessible
Test Steps	1. Reflect on ExchangeRateLogCollection. 2. Retrieve the toArray method docblock. 3. Assert the existence of '@param' in docblock.
Test Data	- Input: docblock string - Expected value: '@param' present
Expected Result	- '@param' annotation found in docblock.
Actual Result	- Docblock contains '@param'.
Status	Pass
Severity	Low

File: ExchangeRateLogRequest-Test.txt

Test Case ID	ERLR-001
Title	Instantiation of ExchangeRateLogRequest
Objective	Verify that the ExchangeRateLogRequest class can be instantiated.
Preconditions	- Application running - PHP environment loaded - Required classes autoloaded
Test Steps	1. Attempt to instantiate the ExchangeRateLogRequest class. 2. Assert that the created object is an instance of ExchangeRateLogRequest.
Test Data	- Input: Instantiation of ExchangeRateLogRequest with no parameters - Expected value: Object is of type ExchangeRateLogRequest
Expected Result	ExchangeRateLogRequest instance is successfully created and is recognized as ExchangeRateLogRequest.
Actual Result	ExchangeRateLogRequest object is instantiated correctly.
Status	Pass
Severity	Medium

Test Case ID	ERLR-002
Title	Parent Class of ExchangeRateLogRequest
Objective	Ensure that ExchangeRateLogRequest extends the FormRequest class.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Instantiate ExchangeRateLogRequest. 2. Assert that it is an instance of Illuminate\Foundation\Http\FormRequest.
Test Data	- Input: Instantiation of ExchangeRateLogRequest - Expected value: Object is instance of FormRequest
Expected Result	ExchangeRateLogRequest is an instance of Illuminate\Foundation\Http\FormRequest.
Actual Result	ExchangeRateLogRequest object is an instance of FormRequest.
Status	Pass
Severity	Medium

Test Case ID	ERLR-003
Title	Namespace of ExchangeRateLogRequest
Objective	Verify that ExchangeRateLogRequest is defined in the correct namespace.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass on ExchangeRateLogRequest. 2. Assert that the namespace is 'Crater\Http\Requests'.
Test Data	- Input: ExchangeRateLogRequest class reflection - Expected value: Namespace equals 'Crater\Http\Requests'
Expected Result	ExchangeRateLogRequest is located in 'Crater\Http\Requests' namespace.
Actual Result	Namespace returned is 'Crater\Http\Requests'.
Status	Pass
Severity	Low

Test Case ID	ERLR-004

Title	Abstract Status of ExchangeRateLogRequest
Objective	Ensure that ExchangeRateLogRequest is not abstract.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to check abstract status. 2. Assert isAbstract() returns false.
Test Data	- Input: ReflectionClass of ExchangeRateLogRequest - Expected value: isAbstract() returns false
Expected Result	ExchangeRateLogRequest is not abstract.
Actual Result	isAbstract() is false.
Status	Pass
Severity	Low

Test Case ID	ERLR-005
Title	Instantiability of ExchangeRateLogRequest
Objective	Confirm that ExchangeRateLogRequest is instantiable.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass on ExchangeRateLogRequest. 2. Assert isInstantiable() returns true.
Test Data	- Input: ReflectionClass of ExchangeRateLogRequest - Expected value: isInstantiable() returns true
Expected Result	ExchangeRateLogRequest is instantiable.
Actual Result	isInstantiable() returns true.
Status	Pass
Severity	Medium

Test Case ID	ERLR-006
Title	Existence of authorize() Method
Objective	Validate that ExchangeRateLogRequest contains an 'authorize' method.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Instantiate ExchangeRateLogRequest. 2. Use method_exists to check for 'authorize'.
Test Data	- Input: ExchangeRateLogRequest object - Expected value: 'authorize' method exists
Expected Result	'authorize' method is present in ExchangeRateLogRequest.
Actual Result	Method exists as expected.
Status	Pass
Severity	High

Test Case ID	ERLR-007
Title	Existence of rules() Method
Objective	Ensure the presence of the 'rules' method in ExchangeRateLogRequest.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Instantiate ExchangeRateLogRequest. 2. Use method_exists to check for 'rules'.

Test Data	- Input: ExchangeRateLogRequest object - Expected value: 'rules' method exists
Expected Result	'rules' method exists in ExchangeRateLogRequest.
Actual Result	Method exists as expected.
Status	Pass
Severity	High

Test Case ID	ERLR-008
Title	Existence of getExchangeRateLogPayload() Method
Objective	Confirm that ExchangeRateLogRequest has the 'getExchangeRateLogPayload' method.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Instantiate ExchangeRateLogRequest. 2. Use method_exists to check for 'getExchangeRateLogPayload'.
Test Data	- Input: ExchangeRateLogRequest object - Expected value: 'getExchangeRateLogPayload' method exists
Expected Result	'getExchangeRateLogPayload' method exists.
Actual Result	Method exists as expected.
Status	Pass
Severity	High

Test Case ID	ERLR-009
Title	Return Value of authorize() Method
Objective	Validate that the 'authorize' method returns true.
Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call the 'authorize' method. 2. Assert the return value is true.
Test Data	- Input: Call to authorize() on ExchangeRateLogRequest object - Expected value: true
Expected Result	'authorize' returns true.
Actual Result	Method returns true.
Status	Pass
Severity	High

Test Case ID	ERLR-010
Title	Public Visibility of authorize() Method
Objective	Ensure 'authorize' method is public.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass on ExchangeRateLogRequest. 2. Get 'authorize' method via reflection. 3. Assert isPublic() returns true.
Test Data	- Input: Reflection of the authorize method - Expected value: isPublic() true
Expected Result	'authorize' is public.
Actual Result	Method is public.

Status	Pass
Severity	High

Test Case ID	ERLR-011
Title	authorize() Method Parameter Count
Objective	Confirm 'authorize' method has zero parameters.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass on ExchangeRateLogRequest. 2. Get 'authorize' method via reflection. 3. Assert getNumberOfParameters() returns 0.
Test Data	- Input: Reflection of authorize() method - Expected value: 0 parameters
Expected Result	'authorize' has no parameters.
Actual Result	Method has no parameters.
Status	Pass
Severity	Medium

Test Case ID	ERLR-012
Title	authorize() Method Return Type
Objective	Confirm 'authorize' method returns a boolean value.
Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call 'authorize' method. 2. Assert result is of boolean type.
Test Data	- Input: Call to authorize() - Expected value: boolean
Expected Result	Method returns a boolean value.
Actual Result	Return type is boolean.
Status	Pass
Severity	High

Test Case ID	ERLR-013
Title	Public Visibility of rules() Method
Objective	Ensure 'rules' method is public.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass on ExchangeRateLogRequest. 2. Get 'rules' method via reflection. 3. Assert isPublic() returns true.
Test Data	- Input: Reflection of rules() method - Expected value: isPublic() true
Expected Result	'rules' is public.
Actual Result	Method is public.
Status	Pass
Severity	High

Test Case ID	ERLR-014

Title	rules() Method Parameter Count
Objective	Confirm 'rules' method has zero parameters.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass on ExchangeRateLogRequest. 2. Get 'rules' method via reflection. 3. Assert getNumberOfParameters() returns 0.
Test Data	- Input: Reflection of rules() method - Expected value: 0 parameters
Expected Result	'rules' has no parameters.
Actual Result	Method has no parameters.
Status	Pass
Severity	Medium

Test Case ID	ERLR-015
Title	Return Type of rules() Method
Objective	Ensure 'rules' method returns an array.
Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call rules() method. 2. Assert the return value is an array.
Test Data	- Input: Call to rules() - Expected value: array
Expected Result	Method returns an array.
Actual Result	Return type is array.
Status	Pass
Severity	High

Test Case ID	ERLR-016
Title	exchange_rate Validation in rules() Method
Objective	Confirm 'exchange_rate' rule exists and is required in the validation array.
Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call rules() method of ExchangeRateLogRequest. 2. Check array for 'exchange_rate' key. 3. Assert that validation contains 'required'.
Test Data	- Input: Output from rules() - Expected values: 'exchange_rate' key present; 'required' is in rule array
Expected Result	exchange_rate is a required validation rule.
Actual Result	exchange_rate exists and contains 'required'.
Status	Pass
Severity	High

Test Case ID	ERLR-017
Title	currency_id Validation in rules() Method
Objective	Confirm 'currency_id' rule exists and is required in the validation array.
Preconditions	- Application running - ExchangeRateLogRequest instantiated

Test Steps	1. Call rules() method of ExchangeRateLogRequest. 2. Check array for 'currency_id' key. 3. Assert that validation contains 'required'.
Test Data	- Input: Output from rules() - Expected values: 'currency_id' key present; 'required' is in rule array
Expected Result	currency_id is a required validation rule.
Actual Result	currency_id exists and contains 'required'.
Status	Pass
Severity	High

Test Case ID	ERLR-018
Title	Validation Rule Count in rules() Method
Objective	Ensure rules() returns exactly two validation rules.
Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call rules() method. 2. Count number of keys in returned array.
Test Data	- Input: Output from rules() - Expected value: count equals 2
Expected Result	rules() returns exactly two validation rules.
Actual Result	Two validation rules returned.
Status	Pass
Severity	Medium

Test Case ID	ERLR-019
Title	Public Visibility of getExchangeRateLogPayload() Method
Objective	Confirm 'getExchangeRateLogPayload' method is public.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass on ExchangeRateLogRequest. 2. Get 'getExchangeRateLogPayload' via reflection. 3. Assert isPublic() returns true.
Test Data	- Input: Reflection of getExchangeRateLogPayload method - Expected value: isPublic() true
Expected Result	Method is public.
Actual Result	Method is public.
Status	Pass
Severity	High

Test Case ID	ERLR-020
Title	getExchangeRateLogPayload() Method Parameter Count
Objective	Confirm 'getExchangeRateLogPayload' method has zero parameters.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass on ExchangeRateLogRequest. 2. Get 'getExchangeRateLogPayload' via reflection. 3. Assert getNumberOfParameters() returns 0.
Test Data	- Input: Reflection of getExchangeRateLogPayload method - Expected value: 0 parameters

Expected Result	Method has no parameters.
Actual Result	Method has no parameters.
Status	Pass
Severity	Medium

Test Case ID	ERLR-021
Title	Static Status of Methods
Objective	Ensure that authorize, rules, and getExchangeRateLogPayload are not static methods.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to get method info. 2. Check isStatic() for each method.
Test Data	- Input: Reflection of methods - Expected value: isStatic() false for all
Expected Result	Methods are not static.
Actual Result	All tested methods are not static.
Status	Pass
Severity	Medium

Test Case ID	ERLR-022
Title	Abstract Status of Methods
Objective	Ensure that authorize, rules, and getExchangeRateLogPayload are not abstract methods.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to get methods. 2. Check isAbstract() for each method.
Test Data	- Input: Reflection of methods - Expected value: isAbstract() false for all
Expected Result	Methods are not abstract.
Actual Result	All tested methods are not abstract.
Status	Pass
Severity	Medium

Test Case ID	ERLR-023
Title	Multiple Instances Creation
Objective	Confirm that multiple instances of ExchangeRateLogRequest can be created and are unique.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Create two instances of ExchangeRateLogRequest. 2. Assert both are instances of ExchangeRateLogRequest. 3. Assert both instances are not the same object.
Test Data	- Input: Creation of two instances - Expected values: Instances are distinct and of correct type
Expected Result	Two unique instances are created successfully.
Actual Result	Instances created and are distinct.
Status	Pass

Severity	Medium

Test Case ID	ERLR-024
Title	Cloning ExchangeRateLogRequest
Objective	Confirm that an ExchangeRateLogRequest object can be cloned and is unique.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Instantiate ExchangeRateLogRequest. 2. Clone the object. 3. Assert clone is instance of ExchangeRateLogRequest and different from original.
Test Data	- Input: Clone operation - Expected values: Cloned object is same type, not same instance
Expected Result	Clone is a unique ExchangeRateLogRequest instance.
Actual Result	Cloning produces a distinct instance.
Status	Pass
Severity	Low

Test Case ID	ERLR-025
Title	Type Hint Usage of ExchangeRateLogRequest
Objective	Confirm that ExchangeRateLogRequest can be used in PHP type hints.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Define a function accepting ExchangeRateLogRequest as type hint. 2. Pass an ExchangeRateLogRequest object. 3. Assert function receives and returns the same object.
Test Data	- Input: Function with type hint and valid object - Expected value: Object is returned correctly
Expected Result	Function type hint works with ExchangeRateLogRequest.
Actual Result	Type hint works as expected.
Status	Pass
Severity	Low

Test Case ID	ERLR-026
Title	Final Status of ExchangeRateLogRequest
Objective	Confirm that ExchangeRateLogRequest class is not final.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to check isFinal(). 2. Assert isFinal() returns false.
Test Data	- Input: ReflectionClass of ExchangeRateLogRequest - Expected value: isFinal() false
Expected Result	Class is not final.
Actual Result	Class is not final.
Status	Pass
Severity	Low

Test Case ID	ERLR-027

Title	Interface Status of ExchangeRateLogRequest
Objective	Ensure ExchangeRateLogRequest is not an interface.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to check isInterface(). 2. Assert isInterface() returns false.
Test Data	- Input: ReflectionClass of ExchangeRateLogRequest - Expected value: isInterface() false
Expected Result	Class is not an interface.
Actual Result	Class is not an interface.
Status	Pass
Severity	Low

Test Case ID	ERLR-028
Title	Trait Status of ExchangeRateLogRequest
Objective	Confirm that ExchangeRateLogRequest is not a trait.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to check isTrait(). 2. Assert isTrait() returns false.
Test Data	- Input: ReflectionClass of ExchangeRateLogRequest - Expected value: isTrait() false
Expected Result	Class is not a trait.
Actual Result	Class is not a trait.
Status	Pass
Severity	Low

Test Case ID	ERLR-029
Title	Class Load Status
Objective	Verify ExchangeRateLogRequest class is loaded in the runtime.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use class_exists to check ExchangeRateLogRequest class. 2. Assert function returns true.
Test Data	- Input: class_exists(ExchangeRateLogRequest::class) - Expected value: true
Expected Result	Class is loaded.
Actual Result	Class exists.
Status	Pass
Severity	High

Test Case ID	ERLR-030
Title	Use Statement Validation for Required Classes
Objective	Confirm the file imports Crater\Models\CompanySetting and FormRequest via use statements.
Preconditions	- Application running - PHP environment loaded

Test Steps	1. Get file contents of ExchangeRateLogRequest. 2. Search for 'use Crater\Models\CompanySetting'. 3. Search for 'use Illuminate\Foundation\Http\FormRequest'.
Test Data	- Input: File content - Expected values: Both use statements are present
Expected Result	Required use statements are present.
Actual Result	Both imports found.
Status	Pass
Severity	Medium

Test Case ID	ERLR-031
Title	File Structure of ExchangeRateLogRequest
Objective	Confirm class and method structure in source file matches expectations.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents of ExchangeRateLogRequest. 2. Check for class declaration extending FormRequest. 3. Check for 'authorize', 'rules', 'getExchangeRateLogPayload' methods.
Test Data	- Input: File content - Expected values: Class and method signatures present
Expected Result	Class and required methods are defined as expected.
Actual Result	Class structure aligns with requirements.
Status	Pass
Severity	High

Test Case ID	ERLR-032
Title	File Size of ExchangeRateLogRequest
Objective	Confirm that the size of the ExchangeRateLogRequest file is less than 2000 bytes.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Check byte size. 3. Assert size < 2000 bytes.
Test Data	- Input: File content - Expected value: Size < 2000 bytes
Expected Result	File size is compact.
Actual Result	File size is less than 2000 bytes.
Status	Pass
Severity	Low

Test Case ID	ERLR-033
Title	File Line Count of ExchangeRateLogRequest
Objective	Ensure line count in the file is less than 70 lines.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Count lines. 3. Assert line count < 70.

Test Data	- Input: File content - Expected value: Line count < 70
Expected Result	File line count is below threshold.
Actual Result	File line count is less than 70.
Status	Pass
Severity	Low

Test Case ID	ERLR-034
Title	Validation Logic in rules() Implementation
Objective	Confirm rules() method contains logic for 'exchange_rate', 'currency_id', and 'required'.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for 'exchange_rate', 'currency_id', and 'required' in rules().
Test Data	- Input: File content - Expected values: Strings present in rules() method
Expected Result	Validation logic is present in rules().
Actual Result	All required validation statements exist.
Status	Pass
Severity	High

Test Case ID	ERLR-035
Title	CompanySetting Usage in getExchangeRateLogPayload()
Objective	Verify that getExchangeRateLogPayload uses CompanySetting::getSetting.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for 'CompanySetting::getSetting' in getExchangeRateLogPayload.
Test Data	- Input: File content - Expected value: Use of CompanySetting::getSetting
Expected Result	getExchangeRateLogPayload uses CompanySetting::getSetting.
Actual Result	Method calls CompanySetting::getSetting.
Status	Pass
Severity	Medium

Test Case ID	ERLR-036
Title	currency_id Checked in getExchangeRateLogPayload()
Objective	Confirm that getExchangeRateLogPayload checks currency_id.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for '$this->currency_id' in getExchangeRateLogPayload.
Test Data	- Input: File content - Expected value: '$this->currency_id' is used
Expected Result	getExchangeRateLogPayload checks currency_id.
Actual Result	currency_id is checked within method.
Status	Pass

Severity	Medium

Test Case ID	ERLR-037
Title	Header Method Usage in getExchangeRateLogPayload()
Objective	Confirm getExchangeRateLogPayload uses $this->header method.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for '$this->header' in method.
Test Data	- Input: File content - Expected value: Usage of $this->header
Expected Result	getExchangeRateLogPayload uses header method.
Actual Result	header method is used.
Status	Pass
Severity	Low

Test Case ID	ERLR-038
Title	Use of validated() Method in getExchangeRateLogPayload()
Objective	Confirm getExchangeRateLogPayload uses $this->validated().
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for '$this->validated()' in method.
Test Data	- Input: File content - Expected value: Usage of validated()
Expected Result	getExchangeRateLogPayload uses validated() method.
Actual Result	validated() method is used.
Status	Pass
Severity	Low

Test Case ID	ERLR-039
Title	Use of Laravel Collection Helpers in getExchangeRateLogPayload()
Objective	Confirm method uses 'collect', 'merge', and 'toArray'.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for 'collect', 'merge', 'toArray' in method.
Test Data	- Input: File content - Expected value: All helpers are used in method
Expected Result	Method utilizes Laravel collection helpers.
Actual Result	Helpers collect, merge, toArray are used.
Status	Pass
Severity	Low

Test Case ID	ERLR-040
Title	Inclusion of company_id in Payload
Objective	Verify that getExchangeRateLogPayload includes company_id property.

Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for 'company_id' in payload building logic.
Test Data	- Input: File content - Expected value: 'company_id' included
Expected Result	Payload contains company_id.
Actual Result	company_id is present in payload.
Status	Pass
Severity	Medium

Test Case ID	ERLR-041
Title	Inclusion of base_currency_id in Payload
Objective	Confirm that getExchangeRateLogPayload includes base_currency_id property.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for 'base_currency_id' in payload logic.
Test Data	- Input: File content - Expected value: 'base_currency_id' included
Expected Result	Payload contains base_currency_id.
Actual Result	base_currency_id is present in payload.
Status	Pass
Severity	Medium

Test Case ID	ERLR-042
Title	Conditional Logic in getExchangeRateLogPayload
Objective	Ensure getExchangeRateLogPayload method uses conditional logic.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for 'if (' in method.
Test Data	- Input: File content - Expected value: Conditional logic present
Expected Result	Conditional statements exist in method.
Actual Result	Conditional logic found.
Status	Pass
Severity	Low

Test Case ID	ERLR-043
Title	Use of Strict Comparison in getExchangeRateLogPayload
Objective	Ensure strict comparison ('!==') is used in method logic.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Get file contents. 2. Search for '!==' in method.
Test Data	- Input: File content - Expected value: '!==' found

Expected Result	Method uses strict comparison.
Actual Result	Strict comparison ('!==') is used.
Status	Pass
Severity	Low

Test Case ID	ERLR-044
Title	Documentation for authorize() Method
Objective	Confirm that authorize() method contains doc comment.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to get authorize() method. 2. Get doc comment. 3. Assert doc comment exists.
Test Data	- Input: ReflectionMethod of authorize() - Expected value: Doc comment is present
Expected Result	Method is documented.
Actual Result	Doc comment exists.
Status	Pass
Severity	Low

Test Case ID	ERLR-045
Title	Documentation for rules() Method
Objective	Confirm that rules() method contains doc comment.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to get rules() method. 2. Get doc comment. 3. Assert doc comment exists.
Test Data	- Input: ReflectionMethod of rules() - Expected value: Doc comment is present
Expected Result	Method is documented.
Actual Result	Doc comment exists.
Status	Pass
Severity	Low

Test Case ID	ERLR-046
Title	Return Type Documentation in authorize() Method
Objective	Verify @return annotation exists in authorize() doc comment.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to get authorize() method. 2. Get doc comment. 3. Assert '@return' string is present.
Test Data	- Input: Doc comment string - Expected value: '@return' annotation present
Expected Result	Return type documented in authorize() doc comment.
Actual Result	'@return' annotation found.
Status	Pass

Severity	Low

Test Case ID	ERLR-047
Title	Return Type Documentation in rules() Method
Objective	Verify @return annotation exists in rules() doc comment.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to get rules() method. 2. Get doc comment. 3. Assert '@return' string is present.
Test Data	- Input: Doc comment string - Expected value: '@return' annotation present
Expected Result	Return type documented in rules() doc comment.
Actual Result	'@return' annotation found.
Status	Pass
Severity	Low

Test Case ID	ERLR-048
Title	exchange_rate Rule Structure
Objective	Ensure 'exchange_rate' validation rule is an array.
Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call rules() method. 2. Assert 'exchange_rate' key is array.
Test Data	- Input: Output from rules()['exchange_rate'] - Expected value: Array type
Expected Result	Validation rule is array.
Actual Result	Rule is array.
Status	Pass
Severity	High

Test Case ID	ERLR-049
Title	currency_id Rule Structure
Objective	Ensure 'currency_id' validation rule is an array.
Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call rules() method. 2. Assert 'currency_id' key is array.
Test Data	- Input: Output from rules()['currency_id'] - Expected value: Array type
Expected Result	Validation rule is array.
Actual Result	Rule is array.
Status	Pass
Severity	High

Test Case ID	ERLR-050
Title	exchange_rate Validation Rule Count
Objective	Confirm 'exchange_rate' rule has exactly one validation.

Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call rules() method. 2. Count elements of rules()['exchange_rate'].
Test Data	- Input: Output from rules()['exchange_rate'] - Expected value: 1
Expected Result	exchange_rate rule array has one element.
Actual Result	Rule array has one item.
Status	Pass
Severity	High

Test Case ID	ERLR-051
Title	currency_id Validation Rule Count
Objective	Confirm 'currency_id' rule has exactly one validation.
Preconditions	- Application running - ExchangeRateLogRequest instantiated
Test Steps	1. Call rules() method. 2. Count elements of rules()['currency_id'].
Test Data	- Input: Output from rules()['currency_id'] - Expected value: 1
Expected Result	currency_id rule array has one element.
Actual Result	Rule array has one item.
Status	Pass
Severity	High

Test Case ID	ERLR-052
Title	Parent Class of ExchangeRateLogRequest
Objective	Confirm parent class is Illuminate\Foundation\Http\FormRequest.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Use ReflectionClass to get parent class. 2. Assert parent class is FormRequest.
Test Data	- Input: Reflection information - Expected value: Parent class name
Expected Result	Parent is FormRequest.
Actual Result	Parent is FormRequest.
Status	Pass
Severity	High

Test Case ID	ERLR-053
Title	Request Inheritance Verification
Objective	Verify ExchangeRateLogRequest is instance of Illuminate\Http\Request.
Preconditions	- Application running - PHP environment loaded
Test Steps	1. Instantiate ExchangeRateLogRequest. 2. Assert instance of Request.
Test Data	- Input: ExchangeRateLogRequest object - Expected value: Instance of Request

Expected Result	Object is instance of Illuminate\Http\Request.
Actual Result	Object is instance of Request.
Status	Pass
Severity	High

File: ExchangeRateProvider-Test.txt

Test Case ID	ERP-001
Title	Instantiation of ExchangeRateProvider
Objective	Verify that ExchangeRateProvider can be instantiated as an object.
Preconditions	- Application running - Database seeded - ExchangeRateProvider model is defined
Test Steps	1. Instantiate a new object of ExchangeRateProvider. 2. Check the type of the instantiated object.
Test Data	- None
Expected Result	- The object is an instance of ExchangeRateProvider.
Actual Result	- The object is correctly instantiated as ExchangeRateProvider.
Status	Pass
Severity	Medium

Test Case ID	ERP-002
Title	ExchangeRateProvider Extends Model
Objective	Verify ExchangeRateProvider inherits from Laravel's Model class.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Check if the instance is of type Illuminate\Database\Eloquent\Model.
Test Data	- None
Expected Result	- The object is an instance of Illuminate\Database\Eloquent\Model.
Actual Result	- ExchangeRateProvider correctly extends Model.
Status	Pass
Severity	Medium

Test Case ID	ERP-003
Title	ExchangeRateProvider Namespace Verification
Objective	Verify the model is in the correct namespace.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection on ExchangeRateProvider class. 2. Retrieve its declared namespace.
Test Data	- None
Expected Result	- Namespace is 'Crater\Models'.
Actual Result	- Namespace is 'Crater\Models'.
Status	Pass
Severity	Low

Test Case ID	ERP-004
Title	ExchangeRateProvider Is Not Abstract
Objective	Verify that ExchangeRateProvider is not an abstract class.
Preconditions	- Application running - ExchangeRateProvider class file available

Test Steps	1. Use reflection to check if ExchangeRateProvider is abstract.
Test Data	- None
Expected Result	- isAbstract() returns false.
Actual Result	- ExchangeRateProvider is confirmed not abstract.
Status	Pass
Severity	Low

Test Case ID	ERP-005
Title	ExchangeRateProvider Is Instantiable
Objective	Verify that ExchangeRateProvider can be instantiated.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection to check if ExchangeRateProvider is instantiable.
Test Data	- None
Expected Result	- isInstantiable() returns true.
Actual Result	- ExchangeRateProvider is instantiable.
Status	Pass
Severity	Medium

Test Case ID	ERP-006
Title	Guarded Properties Include 'id'
Objective	Verify that the 'guarded' array contains 'id'.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Retrieve and inspect guarded property.
Test Data	- None
Expected Result	- getGuarded() returns ['id'].
Actual Result	- Guarded property correctly set to ['id'].
Status	Pass
Severity	Medium

Test Case ID	ERP-007
Title	Prevent Mass Assignment of 'id'
Objective	Validate that 'id' is not mass assignable.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Attempt to fill 'id' using mass assignment. 3. Check that 'id' remains null, while other attributes are assigned.
Test Data	- Fill data: ['id' => 999, 'driver' => 'test']
Expected Result	- $provider->id is null. - $provider->driver is 'test'.
Actual Result	- id is null; driver is 'test'.
Status	Pass
Severity	Medium

Test Case ID	ERP-008
Title	Cast 'currencies' Attribute to Array
Objective	Confirm 'currencies' is cast to an array.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Retrieve casts array from model. 3. Check existence and value of 'currencies' key.
Test Data	- None
Expected Result	- 'currencies' exists in casts. - 'currencies' is set to 'array'.
Actual Result	- Casts array includes 'currencies' as 'array'.
Status	Pass
Severity	Medium

Test Case ID	ERP-009
Title	Cast 'driver_config' Attribute to Array
Objective	Confirm 'driver_config' is cast to an array.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Retrieve casts array from model. 3. Check existence and value of 'driver_config' key.
Test Data	- None
Expected Result	- 'driver_config' exists in casts. - 'driver_config' is 'array'.
Actual Result	- Casts array includes 'driver_config' as 'array'.
Status	Pass
Severity	Medium

Test Case ID	ERP-010
Title	Cast 'active' Attribute to Boolean
Objective	Confirm 'active' is cast to boolean.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Retrieve casts array. 3. Check existence and value of 'active'.
Test Data	- None
Expected Result	- 'active' exists in casts. - 'active' is 'boolean'.
Actual Result	- Casts array includes 'active' as 'boolean'.
Status	Pass
Severity	Medium

Test Case ID	ERP-011
Title	'company' Method Existence
Objective	Verify 'company' method exists in ExchangeRateProvider.

Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Check if 'company' method exists.
Test Data	- None
Expected Result	- method_exists($provider, 'company') returns true.
Actual Result	- Method 'company' is present.
Status	Pass
Severity	Medium

Test Case ID	ERP-012
Title	'company' Relationship Type is BelongsTo
Objective	Confirm company relationship returns a BelongsTo relation.
Preconditions	- Application running - ExchangeRateProvider and Company models defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Invoke 'company' relationship. 3. Check relation type.
Test Data	- None
Expected Result	- Relation is instance of BelongsTo.
Actual Result	- company() returns a BelongsTo relation.
Status	Pass
Severity	Medium

Test Case ID	ERP-013
Title	'company' Relationship Targets Company Model
Objective	Ensure relationship points to Company model.
Preconditions	- Application running - ExchangeRateProvider and Company models defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Invoke 'company' relationship. 3. Check related model type.
Test Data	- None
Expected Result	- getRelated() returns Company instance.
Actual Result	- Relation targets Company model.
Status	Pass
Severity	Medium

Test Case ID	ERP-014
Title	'company' Relationship Foreign Key is 'company_id'
Objective	Ensure foreign key of relationship is 'company_id'.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Invoke company relationship. 3. Verify foreign key name.
Test Data	- None
Expected Result	- getForeignKeyName() returns 'company_id'.

Actual Result	- Foreign key is 'company_id'.
Status	Pass
Severity	Medium

Test Case ID	ERP-015
Title	'company' Relationship Owner Key is 'id'
Objective	Ensure owner key for relationship is 'id'.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Invoke company relationship. 3. Verify owner key name.
Test Data	- None
Expected Result	- getOwnerKeyName() returns 'id'.
Actual Result	- Owner key is 'id'.
Status	Pass
Severity	Medium

Test Case ID	ERP-016
Title	'setCurrenciesAttribute' Method Exists
Objective	Confirm the existence of 'setCurrenciesAttribute'.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Check if 'setCurrenciesAttribute' exists.
Test Data	- None
Expected Result	- method_exists($provider, 'setCurrenciesAttribute') returns true.
Actual Result	- Method exists as required.
Status	Pass
Severity	Medium

Test Case ID	ERP-017
Title	'setCurrenciesAttribute' JSON Encodes Array
Objective	Verify setter encodes arrays as JSON.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set currencies attribute with array ['USD', 'EUR', 'GBP']. 3. Inspect internal attributes.
Test Data	- Currencies array: ['USD', 'EUR', 'GBP']
Expected Result	- attributes['currencies'] is json encode of array.
Actual Result	- Attribute is '{"USD","EUR","GBP"}'.
Status	Pass
Severity	Medium

Test Case ID	ERP-018
Title	'setCurrenciesAttribute' Handles Empty Array

Objective	Verify setter encodes empty arrays as JSON.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set currencies attribute with []. 3. Inspect internal attributes.
Test Data	- Currencies array: []
Expected Result	- attributes['currencies'] is '[]'.
Actual Result	- Attribute is '[]'.
Status	Pass
Severity	Medium

Test Case ID	ERP-019
Title	'setCurrenciesAttribute' Handles Single Currency
Objective	Verify setter encodes single-item arrays as JSON.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set currencies attribute with ['USD']. 3. Inspect internal attributes.
Test Data	- Currencies array: ['USD']
Expected Result	- attributes['currencies'] is '["USD"]'.
Actual Result	- Attribute is '["USD"]'.
Status	Pass
Severity	Medium

Test Case ID	ERP-020
Title	'setDriverConfigAttribute' Method Existence
Objective	Confirm existence of 'setDriverConfigAttribute'.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Check if 'setDriverConfigAttribute' exists.
Test Data	- None
Expected Result	- Method exists on model.
Actual Result	- Method present.
Status	Pass
Severity	Medium

Test Case ID	ERP-021
Title	'setDriverConfigAttribute' JSON Encodes Array
Objective	Verify setter encodes config arrays as JSON.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set driver_config attribute with array ['api_key' => 'test123', 'base_url' => 'https://api.example.com']. 3. Inspect internal attributes.

Test Data	- Config array: ['api_key' => 'test123', 'base_url' => 'https://api.example.com']
Expected Result	- attributes['driver_config'] is JSON of config.
Actual Result	- Attribute is '{"api_key":"test123","base_url":"https://api.example.com"}'.
Status	Pass
Severity	Medium

Test Case ID	ERP-022
Title	'setDriverConfigAttribute' Handles Empty Config
Objective	Verify setter encodes empty arrays as JSON.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set driver_config attribute with []. 3. Inspect internal attributes.
Test Data	- Config array: []
Expected Result	- attributes['driver_config'] is '[]'.
Actual Result	- Attribute is '[]'.
Status	Pass
Severity	Medium

Test Case ID	ERP-023
Title	'setDriverConfigAttribute' Handles Complex Config
Objective	Verify setter encodes complex config arrays as JSON.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set driver_config attribute with array ['type' => 'PREMIUM', 'url' => ..., 'timeout' => 30, 'retry' => true]. 3. Inspect internal attributes.
Test Data	- Config array: ['type' => 'PREMIUM', 'url' => 'https://api.example.com', 'timeout' => 30, 'retry' => true]
Expected Result	- attributes['driver_config'] is JSON encoding of config.
Actual Result	- Attribute is JSON encoded as expected.
Status	Pass
Severity	Medium

Test Case ID	ERP-024
Title	'getCurrencyConverterUrl' Static Method Existence
Objective	Confirm 'getCurrencyConverterUrl' static method exists.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Check if static method exists in ExchangeRateProvider.
Test Data	- None
Expected Result	- Static method exists.
Actual Result	- Static method present.
Status	Pass
Severity	Medium

Test Case ID	ERP-025
Title	'getCurrencyConverterUrl' is Static
Objective	Confirm 'getCurrencyConverterUrl' method is static.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection to inspect 'getCurrencyConverterUrl' method. 2. Verify if method is static.
Test Data	- None
Expected Result	- Method is static.
Actual Result	- Method is static.
Status	Pass
Severity	Medium

Test Case ID	ERP-026
Title	'getCurrencyConverterUrl' Returns PREMIUM URL
Objective	Confirm correct URL for type 'PREMIUM'.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Call getCurrencyConverterUrl(['type' => 'PREMIUM']).
Test Data	- Input data: ['type' => 'PREMIUM'] - Expected value: 'https://api.currconv.com'
Expected Result	- URL is 'https://api.currconv.com'.
Actual Result	- Correct URL returned.
Status	Pass
Severity	High

Test Case ID	ERP-027
Title	'getCurrencyConverterUrl' Returns PREPAID URL
Objective	Confirm correct URL for type 'PREPAID'.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Call getCurrencyConverterUrl(['type' => 'PREPAID']).
Test Data	- Input data: ['type' => 'PREPAID'] - Expected value: 'https://prepaid.currconv.com'
Expected Result	- URL is 'https://prepaid.currconv.com'.
Actual Result	- PREPAID URL returned.
Status	Pass
Severity	High

Test Case ID	ERP-028
Title	'getCurrencyConverterUrl' Returns FREE URL
Objective	Confirm correct URL for type 'FREE'.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Call getCurrencyConverterUrl(['type' => 'FREE']).

Test Data	- Input data: ['type' => 'FREE'] - Expected value: 'https://free.currconv.com'
Expected Result	- URL is 'https://free.currconv.com'.
Actual Result	- FREE URL returned.
Status	Pass
Severity	High

Test Case ID	ERP-029
Title	'getCurrencyConverterUrl' Returns DEDICATED URL
Objective	Confirm correct URL for type 'DEDICATED' is from input data.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Call getCurrencyConverterUrl(['type' => 'DEDICATED', 'url' => 'https://mycustom.currconv.com']).
Test Data	- Input data: ['type' => 'DEDICATED', 'url' => 'https://mycustom.currconv.com'] - Expected value: 'https://mycustom.currconv.com'
Expected Result	- URL is 'https://mycustom.currconv.com'.
Actual Result	- DEDICATED URL returned.
Status	Pass
Severity	High

Test Case ID	ERP-030
Title	'getCurrencyConverterUrl' Handles Different DEDICATED URLs
Objective	Verify function returns different DEDICATED URLs based on input.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Call getCurrencyConverterUrl with a unique 'DEDICATED' URL in data.
Test Data	- Input data: ['type' => 'DEDICATED', 'url' => 'https://another-custom.example.com'] - Expected value: 'https://another-custom.example.com'
Expected Result	- URL matches input value.
Actual Result	- Input DEDICATED URL is returned.
Status	Pass
Severity	High

Test Case ID	ERP-031
Title	'getCurrencyConverterUrl' Returns Null for Unknown Type
Objective	Function handles unknown type gracefully.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Call getCurrencyConverterUrl(['type' => 'UNKNOWN']).
Test Data	- Input data: ['type' => 'UNKNOWN']
Expected Result	- Returns null.
Actual Result	- Null returned as expected.
Status	Pass
Severity	High

Test Case ID	ERP-032
Title	'getCurrencyConverterUrl' Returns Null for Invalid Type
Objective	Function handles invalid type gracefully.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Call getCurrencyConverterUrl(['type' => 'INVALID']).
Test Data	- Input data: ['type' => 'INVALID']
Expected Result	- Returns null.
Actual Result	- Null returned.
Status	Pass
Severity	High

Test Case ID	ERP-033
Title	ExchangeRateProvider Has All Required Methods
Objective	Ensure all critical methods exist for model functionality.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Check existence of methods: company, setCurrenciesAttribute, setDriverConfigAttribute, scopeWhereCompany, updateFromRequest.
Test Data	- None
Expected Result	- All listed methods exist.
Actual Result	- All required methods present.
Status	Pass
Severity	High

Test Case ID	ERP-034
Title	ExchangeRateProvider Has Static Utility Methods
Objective	Ensure key static utility methods exist.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Check existence of static methods: createFromRequest, checkActiveCurrencies, checkExchangeRateProviderStatus, getCurrencyConverterUrl.
Test Data	- None
Expected Result	- All static methods exist.
Actual Result	- Static utility methods present.
Status	Pass
Severity	High

Test Case ID	ERP-035
Title	ExchangeRateProvider Uses HasFactory Trait
Objective	Confirm the model uses HasFactory trait.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Use reflection to get list of used traits. 2. Check for Illuminate\Database\Eloquent\Factories\HasFactory.

Test Data	- None
Expected Result	- HasFactory trait is present.
Actual Result	- Trait is present.
Status	Pass
Severity	Medium

Test Case ID	ERP-036
Title	Multiple Instances Can Be Created
Objective	Verify multiple instances of ExchangeRateProvider behave independently.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Create two instances. 2. Check both are instances of ExchangeRateProvider. 3. Confirm they are not the same object.
Test Data	- None
Expected Result	- Both are instances of ExchangeRateProvider. - They are not equal.
Actual Result	- Independent instances created.
Status	Pass
Severity	Medium

Test Case ID	ERP-037
Title	ExchangeRateProvider Can Be Cloned
Objective	Verify model supports object cloning.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Create an instance with data. 2. Clone the instance. 3. Check clone is a new ExchangeRateProvider object.
Test Data	- Initial data: ['driver' => 'test']
Expected Result	- Cloned object is instance of ExchangeRateProvider and not the same as original.
Actual Result	- Cloning works as expected.
Status	Pass
Severity	Medium

Test Case ID	ERP-038
Title	ExchangeRateProvider Type Hinting Works
Objective	Confirm ExchangeRateProvider can be type-hinted in functions.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Define a function accepting ExchangeRateProvider. 2. Pass an instance to function and verify returned instance.
Test Data	- None
Expected Result	- Instance is passed and returned correctly.
Actual Result	- Type hinting works.
Status	Pass
Severity	Low

Test Case ID	ERP-039
Title	ExchangeRateProvider Is Not Final
Objective	Confirm model class is not final.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection to check if class is final.
Test Data	- None
Expected Result	- isFinal() returns false.
Actual Result	- Model is not final.
Status	Pass
Severity	Low

Test Case ID	ERP-040
Title	ExchangeRateProvider Is Not an Interface
Objective	Confirm model class is not an interface.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection to check if class is interface.
Test Data	- None
Expected Result	- isInterface() returns false.
Actual Result	- Model is not an interface.
Status	Pass
Severity	Low

Test Case ID	ERP-041
Title	ExchangeRateProvider Is Not a Trait
Objective	Confirm model class is not a trait.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection to check if class is trait.
Test Data	- None
Expected Result	- isTrait() returns false.
Actual Result	- Model is not a trait.
Status	Pass
Severity	Low

Test Case ID	ERP-042
Title	ExchangeRateProvider Class Is Loaded
Objective	Confirm class is loaded by PHP.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Check if class_exists(ExchangeRateProvider::class) returns true.
Test Data	- None
Expected Result	- class_exists returns true.
Actual Result	- Class is loaded.

Status	Pass
Severity	Medium

Test Case ID	ERP-043
Title	ExchangeRateProvider Uses Required Classes in Imports
Objective	Validate all necessary class imports exist in file.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Read content of ExchangeRateProvider file. 2. Check for specific import statements.
Test Data	- Required imports: ExchangeRateProviderRequest, HasFactory, Model, Http
Expected Result	- All required imports present in file.
Actual Result	- All import statements found.
Status	Pass
Severity	High

Test Case ID	ERP-044
Title	ExchangeRateProvider File Structure Is As Expected
Objective	Validate file contains critical structure elements.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Read file content. 2. Check for presence of class definition, protected $guarded, protected $casts.
Test Data	- None
Expected Result	- File contains necessary structural elements.
Actual Result	- Structure is correct.
Status	Pass
Severity	Medium

Test Case ID	ERP-045
Title	ExchangeRateProvider Has Reasonable Line Count
Objective	Ensure code file meets minimum complexity.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Count lines in file. 2. Check if greater than 100.
Test Data	- None
Expected Result	- Line count exceeds 100.
Actual Result	- Sufficient lines in file.
Status	Pass
Severity	Low

Test Case ID	ERP-046
Title	Set and Get 'driver' Attribute
Objective	Verify ability to set and retrieve driver attribute value.

Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set driver to 'currency_freak'. 3. Retrieve driver value.
Test Data	- Set value: 'currency_freak'
Expected Result	- driver attribute is 'currency_freak'.
Actual Result	- Attribute value matches.
Status	Pass
Severity	Medium

Test Case ID	ERP-047
Title	Set and Get 'active' Attribute
Objective	Confirm active attribute can be set and retrieved.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set active = true. 3. Retrieve value.
Test Data	- Set value: true
Expected Result	- active attribute is true.
Actual Result	- Value is true.
Status	Pass
Severity	Medium

Test Case ID	ERP-048
Title	'active' Attribute Casts to Boolean
Objective	Validate type casting for 'active' attribute.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set active = 1. 3. Check returned value and type.
Test Data	- Set value: 1
Expected Result	- active attribute is true. - Type is boolean.
Actual Result	- Value is true (boolean).
Status	Pass
Severity	Medium

Test Case ID	ERP-049
Title	Set and Get 'company_id' Attribute
Objective	Verify ability to set and retrieve company_id.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set company_id = 5. 3. Retrieve value.
Test Data	- Set value: 5

Expected Result	- company_id attribute is 5.
Actual Result	- company_id is set to 5.
Status	Pass
Severity	Medium

Test Case ID	ERP-050
Title	'setCurrenciesAttribute' Is Public
Objective	Ensure method visibility is public.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection to inspect method. 2. Check if visibility is public.
Test Data	- None
Expected Result	- Method has public visibility.
Actual Result	- Public method confirmed.
Status	Pass
Severity	Medium

Test Case ID	ERP-051
Title	'setDriverConfigAttribute' Is Public
Objective	Ensure visibility of method is public.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection to inspect setDriverConfigAttribute. 2. Check method visibility.
Test Data	- None
Expected Result	- Method is public.
Actual Result	- Method is public.
Status	Pass
Severity	Medium

Test Case ID	ERP-052
Title	'getCurrencyConverterUrl' Is Public
Objective	Confirm method is public.
Preconditions	- Application running - ExchangeRateProvider class file available
Test Steps	1. Use reflection to inspect getCurrencyConverterUrl. 2. Check method visibility.
Test Data	- None
Expected Result	- Method is public.
Actual Result	- Public method confirmed.
Status	Pass
Severity	Medium

Test Case ID	ERP-053
Title	'currencies' Setter Preserves Data Integrity

Objective	Check if setter/getter for 'currencies' maintain data integrity.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set currencies to ['USD', 'EUR', 'GBP', 'JPY']. 3. Retrieve and verify value.
Test Data	- Currencies: ['USD', 'EUR', 'GBP', 'JPY']
Expected Result	- Retrieved value matches input array.
Actual Result	- Data preserved.
Status	Pass
Severity	Medium

Test Case ID	ERP-054
Title	'driver_config' Setter Preserves Data Integrity
Objective	Check if setter/getter for 'driver_config' maintain data integrity.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Set driver_config to ['type' => 'PREMIUM', 'timeout' => 30]. 3. Retrieve and verify value.
Test Data	- Config: ['type' => 'PREMIUM', 'timeout' => 30]
Expected Result	- Retrieved value matches input.
Actual Result	- Data is preserved.
Status	Pass
Severity	Medium

Test Case ID	ERP-055
Title	Different Instances Have Independent Data
Objective	Confirm two model instances maintain separate attribute values.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate provider1 with ['driver' => 'currency_freak']. 2. Instantiate provider2 with ['driver' => 'currency_layer']. 3. Verify driver values differ and match inputs.
Test Data	- provider1: 'currency_freak' - provider2: 'currency_layer'
Expected Result	- Values are independent and correct.
Actual Result	- Each instance has its own data.
Status	Pass
Severity	Medium

Test Case ID	ERP-056
Title	getCurrencyConverterUrl Handles All Valid Types
Objective	Ensure URLs returned for all known types are correct.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Loop through types: PREMIUM, PREPAID, FREE. 2. Call getCurrencyConverterUrl for each. 3. Verify returned URL.

Test Data	- ['type' => 'PREMIUM'] => 'https://api.currconv.com' - ['type' => 'PREPAID'] => 'https://prepaid.currconv.com' - ['type' => 'FREE'] => 'https://free.currconv.com'
Expected Result	- Each type returns corresponding URL.
Actual Result	- URLs for all types correct.
Status	Pass
Severity	High

Test Case ID	ERP-057
Title	getCurrencyConverterUrl Requires URL for DEDICATED Type
Objective	Confirm 'DEDICATED' type requires custom URL input.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Call getCurrencyConverterUrl(['type' => 'DEDICATED', 'url' => 'https://my-dedicated-server.com']). 2. Check returned value.
Test Data	- Input: ['type' => 'DEDICATED', 'url' => 'https://my-dedicated-server.com']
Expected Result	- URL is 'https://my-dedicated-server.com'.
Actual Result	- Custom DEDICATED URL is returned correctly.
Status	Pass
Severity	High

Test Case ID	ERP-058
Title	ExchangeRateProvider Inherits Model Methods
Objective	Ensure model inherits key Eloquent Model methods.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Check for methods: save, fill, toArray.
Test Data	- None
Expected Result	- All Model methods exist.
Actual Result	- All methods are present.
Status	Pass
Severity	High

Test Case ID	ERP-059
Title	ExchangeRateProvider Can Use Model Features
Objective	Confirm model functionality with core Model feature calls.
Preconditions	- Application running - ExchangeRateProvider model is defined
Test Steps	1. Instantiate ExchangeRateProvider. 2. Check if 'fill' and 'toArray' are callable.
Test Data	- None
Expected Result	- Both methods are callable.
Actual Result	- Methods are callable.
Status	Pass
Severity	High

File: ExchangeRateProviderCollection-Test.txt

Test Case ID	ERPC-001
Title	Instantiation of ExchangeRateProviderCollection
Objective	Verify that ExchangeRateProviderCollection can be properly instantiated.
Preconditions	- Application running - Required PHP classes are available - No exchange rate provider data required
Test Steps	1. Instantiate ExchangeRateProviderCollection with an empty Laravel Collection. 2. Check the type of the created object.
Test Data	- Input: new Collection([]) - Expected object class: ExchangeRateProviderCollection
Expected Result	- ExchangeRateProviderCollection object is created successfully. - Object is an instance of ExchangeRateProviderCollection.
Actual Result	ExchangeRateProviderCollection object was instantiated and verified as correct type.
Status	Pass
Severity	Medium

Test Case ID	ERPC-002
Title	ExchangeRateProviderCollection ResourceCollection Inheritance
Objective	Verify that ExchangeRateProviderCollection extends ResourceCollection.
Preconditions	- Application running - Required PHP classes are available
Test Steps	1. Instantiate ExchangeRateProviderCollection with an empty Laravel Collection. 2. Check if object is instance of \Illuminate\Http\Resources\Json\ResourceCollection.
Test Data	- Input: new Collection([]) - Expected parent class: ResourceCollection
Expected Result	- Object is an instance of \Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	ExchangeRateProviderCollection correctly extends ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	ERPC-003
Title	Correct Namespace of ExchangeRateProviderCollection
Objective	Ensure ExchangeRateProviderCollection is under the correct namespace.
Preconditions	- Application running - Classes correctly autoloaded
Test Steps	1. Use ReflectionClass to get namespace name of ExchangeRateProviderCollection. 2. Verify that the namespace is 'Crater\Http\Resources'.
Test Data	- Class: ExchangeRateProviderCollection
Expected Result	- Namespace is 'Crater\Http\Resources'.
Actual Result	Namespace found to be 'Crater\Http\Resources' as expected.
Status	Pass
Severity	Low

Test Case ID	ERPC-004
Title	toArray Method Presence
Objective	Verify that ExchangeRateProviderCollection implements a public toArray method.
Preconditions	- Application running - ExchangeRateProviderCollection class present
Test Steps	1. Instantiate ExchangeRateProviderCollection. 2. Check for the existence of 'toArray' method.
Test Data	- Class: ExchangeRateProviderCollection
Expected Result	- 'toArray' method exists.
Actual Result	Method 'toArray' was found.
Status	Pass
Severity	Medium

Test Case ID	ERPC-005
Title	toArray Method Public Accessibility
Objective	Confirm that the toArray method is public.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get method characteristics for 'toArray'. 2. Check if the method is public.
Test Data	- Method: toArray
Expected Result	- Method is publicly accessible.
Actual Result	Method 'toArray' is public.
Status	Pass
Severity	Medium

Test Case ID	ERPC-006
Title	toArray Method Argument Verification
Objective	Ensure that the toArray method accepts the correct request parameter.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to inspect the 'toArray' method parameters. 2. Confirm parameter named 'request' is present.
Test Data	- Class: ExchangeRateProviderCollection - Method: toArray
Expected Result	- Method has one parameter named 'request'.
Actual Result	Parameter 'request' found as expected.
Status	Pass
Severity	Medium

Test Case ID	ERPC-007
Title	Handling Empty Collection
Objective	Verify that toArray returns an empty array when given no providers.
Preconditions	- Application running
Test Steps	1. Create an empty Collection. 2. Pass it to ExchangeRateProviderCollection. 3. Call toArray with a new Request.

Test Data	- Collection: [] - Expected output: []
Expected Result	- Result is an empty array.
Actual Result	Result is an empty array as expected.
Status	Pass
Severity	Medium

Test Case ID	ERPC-008
Title	Single Provider Transformation
Objective	Ensure toArray transforms a single provider resource correctly.
Preconditions	- Application running - Dummy provider data available
Test Steps	1. Create a dummy exchange rate provider with id=1, driver='currency_freak'. 2. Wrap it in ExchangeRateProviderResource. 3. Create a collection with a single resource. 4. Call toArray with a new Request.
Test Data	- Provider: { id: 1, driver: 'currency_freak', ... } - Expected output: Array with one item with key 'id' as 1
Expected Result	- Result is an array with count 1 - Result[0] has key 'id' and its value is 1
Actual Result	Transformation produced expected single provider array.
Status	Pass
Severity	High

Test Case ID	ERPC-009
Title	Multiple Provider Transformation
Objective	Verify toArray can handle and transform multiple provider resources.
Preconditions	- Application running - Dummy provider data available
Test Steps	1. Create two dummy providers: - id=1, driver='currency_freak' - id=2, driver='currency_layer' 2. Wrap them in respective resources. 3. Create collection with both resources. 4. Call toArray with a new Request.
Test Data	- Providers: [{id:1}, {id:2}] - Expected output: Array of length 2, id values 1 and 2
Expected Result	- Result is array with count 2 - result[0]['id'] == 1 - result[1]['id'] == 2
Actual Result	Array contains both providers as expected.
Status	Pass
Severity	High

Test Case ID	ERPC-010
Title	Provider Required Fields in Transformed Item
Objective	Ensure each transformed provider item includes all required fields.
Preconditions	- Application running

Test Steps	1. Create a dummy provider (id=1). 2. Wrap in ExchangeRateProviderResource. 3. Pass to ExchangeRateProviderCollection and call toArray.
Test Data	- Provider fields: 'id', 'driver', 'key', 'active', 'currencies', 'driver_config' - Expected keys present
Expected Result	- Transformed item contains all listed keys.
Actual Result	Required fields present in transformed array item.
Status	Pass
Severity	High

Test Case ID	ERPC-011
Title	Transformation of Large Provider Collection
Objective	Verify toArray can handle and transform large provider collections (50 items).
Preconditions	- Application running
Test Steps	1. For i=1 to 50, create dummy provider with id=i. 2. Wrap each in ExchangeRateProviderResource. 3. Pass all resources into ExchangeRateProviderCollection. 4. Call toArray.
Test Data	- Providers: 50 items, id=1 to id=50
Expected Result	- Result count is 50 - result[0]['id'] == 1 - result[49]['id'] == 50
Actual Result	Transformation resulted in correct array with expected count and ids.
Status	Pass
Severity	High

Test Case ID	ERPC-012
Title	toArray Delegation to Parent ResourceCollection
Objective	Verify toArray implementation delegates to parent ResourceCollection.
Preconditions	- Application running
Test Steps	1. Reflect on ExchangeRateProviderCollection class source code. 2. Search for 'parent::toArray' usage.
Test Data	- Source file content
Expected Result	- File contains 'parent::toArray'
Actual Result	String 'parent::toArray' found indicating delegation.
Status	Pass
Severity	Medium

Test Case ID	ERPC-013
Title	ExchangeRateProviderCollection Parent Class Verification
Objective	Validate parent class is ResourceCollection.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get parent of ExchangeRateProviderCollection. 2. Check parent class name.
Test Data	- Class: ExchangeRateProviderCollection
Expected Result	- Parent class is \Illuminate\Http\Resources\Json\ResourceCollection
Actual Result	Parent class verified successfully.
Status	Pass

Severity	Medium

Test Case ID	ERPC-014
Title	Multiple Instances Support
Objective	Validate that multiple instances of ExchangeRateProviderCollection can co-exist.
Preconditions	- Application running
Test Steps	1. Create two separate instances of ExchangeRateProviderCollection. 2. Check both are instances of class. 3. Ensure the two objects are not identical.
Test Data	- Input: new Collection([])
Expected Result	- Both instances exist and are distinct.
Actual Result	Multiple instances created and verified.
Status	Pass
Severity	Low

Test Case ID	ERPC-015
Title	Cloning ExchangeRateProviderCollection
Objective	Ensure ExchangeRateProviderCollection can be cloned.
Preconditions	- Application running
Test Steps	1. Instantiate ExchangeRateProviderCollection. 2. Clone the instance. 3. Validate the cloned object is a distinct, valid instance.
Test Data	- Input: Collection([])
Expected Result	- Clone is ExchangeRateProviderCollection type but not the same instance.
Actual Result	Cloning succeeded and produced a distinct object.
Status	Pass
Severity	Low

Test Case ID	ERPC-016
Title	Type Hint Usability of ExchangeRateProviderCollection
Objective	Ensure ExchangeRateProviderCollection can be type hinted in PHP functions.
Preconditions	- Application running
Test Steps	1. Create a function accepting ExchangeRateProviderCollection type. 2. Pass an instance to the function. 3. Ensure function returns the same object.
Test Data	- Input: ExchangeRateProviderCollection object
Expected Result	- Object passes through function without type errors.
Actual Result	Type hint works, object returned as expected.
Status	Pass
Severity	Low

Test Case ID	ERPC-017
Title	Class Non-Abstract Verification
Objective	Confirm that ExchangeRateProviderCollection is not abstract.
Preconditions	- Application running

Test Steps	1. Use ReflectionClass on ExchangeRateProviderCollection. 2. Check isAbstract property.
Test Data	- Class: ExchangeRateProviderCollection
Expected Result	- isAbstract returns false.
Actual Result	Class verified as non-abstract.
Status	Pass
Severity	Low

Test Case ID	ERPC-018
Title	Class Non-Final Verification
Objective	Confirm that ExchangeRateProviderCollection is not declared as final.
Preconditions	- Application running
Test Steps	1. Reflect class and check isFinal property.
Test Data	- Class: ExchangeRateProviderCollection
Expected Result	- isFinal returns false.
Actual Result	Class is not marked as final.
Status	Pass
Severity	Low

Test Case ID	ERPC-019
Title	Not an Interface Verification
Objective	Verify ExchangeRateProviderCollection is not an interface.
Preconditions	- Application running
Test Steps	1. ReflectionClass on ExchangeRateProviderCollection. 2. Check isInterface property.
Test Data	- Class: ExchangeRateProviderCollection
Expected Result	- isInterface returns false.
Actual Result	Class correctly identified as not an interface.
Status	Pass
Severity	Low

Test Case ID	ERPC-020
Title	Not a Trait Verification
Objective	Confirm ExchangeRateProviderCollection is not a trait.
Preconditions	- Application running
Test Steps	1. ReflectionClass on ExchangeRateProviderCollection. 2. Check isTrait property.
Test Data	- Class: ExchangeRateProviderCollection
Expected Result	- isTrait returns false.
Actual Result	Class correctly detected as not a trait.
Status	Pass
Severity	Low

Test Case ID	ERPC-021
Title	Class Loaded Verification

Objective	Ensure ExchangeRateProviderCollection is loaded and accessible.
Preconditions	- Application running
Test Steps	1. Use class_exists() on ExchangeRateProviderCollection.
Test Data	- Class: ExchangeRateProviderCollection
Expected Result	- class_exists returns true.
Actual Result	Class loaded as expected.
Status	Pass
Severity	Medium

Test Case ID	ERPC-022
Title	ResourceCollection Import Usage
Objective	Validate that the file imports ResourceCollection.
Preconditions	- Application running
Test Steps	1. Reflect class source file. 2. Search for 'use Illuminate\Http\Resources\Json\ResourceCollection'.
Test Data	- File content
Expected Result	- Import statement present.
Actual Result	Import found in file.
Status	Pass
Severity	Medium

Test Case ID	ERPC-023
Title	toArray Method Is Not Static
Objective	Confirm toArray method implementation is not static.
Preconditions	- Application running
Test Steps	1. ReflectionClass to inspect method 'toArray'. 2. Check isStatic property.
Test Data	- Method: toArray
Expected Result	- isStatic returns false.
Actual Result	Method is not static as required.
Status	Pass
Severity	Medium

Test Case ID	ERPC-024
Title	toArray Method Is Not Abstract
Objective	Confirm toArray method implementation is not abstract.
Preconditions	- Application running
Test Steps	1. ReflectionClass to inspect method 'toArray'. 2. Check isAbstract property.
Test Data	- Method: toArray
Expected Result	- isAbstract returns false.
Actual Result	Method is implemented and not abstract.
Status	Pass
Severity	Medium

Test Case ID	ERPC-025
Title	Data Integrity Preservation
Objective	Verify exchange rate provider data is preserved in transformation.
Preconditions	- Application running
Test Steps	1. Create dummy provider (id=42, driver='open_exchange_rate'). 2. Wrap in resource. 3. Transform via ExchangeRateProviderCollection.
Test Data	- Provider: id=42, driver='open_exchange_rate'
Expected Result	- Transformed item has id=42 and driver='open_exchange_rate'.
Actual Result	Data integrity preserved in transformation.
Status	Pass
Severity	High

Test Case ID	ERPC-026
Title	Handles Different Provider Drivers
Objective	Ensure transformation preserves individual provider driver values.
Preconditions	- Application running
Test Steps	1. Create three providers with different drivers. 2. Transform collection and inspect output.
Test Data	- Provider drivers: 'currency_freak', 'currency_layer', 'open_exchange_rate'
Expected Result	- result[0]['driver'] == 'currency_freak' - result[1]['driver'] == 'currency_layer' - result[2]['driver'] == 'open_exchange_rate'
Actual Result	Drivers matched expectation.
Status	Pass
Severity	High

Test Case ID	ERPC-027
Title	File Size Is Concise
Objective	Ensure file size for ExchangeRateProviderCollection implementation is small.
Preconditions	- Application running
Test Steps	1. Reflect class source file. 2. Measure byte length. 3. Verify length < 1000 bytes.
Test Data	- File size threshold: 1000 bytes
Expected Result	- File size under 1000 bytes.
Actual Result	File measured at under 1000 bytes.
Status	Pass
Severity	Low

Test Case ID	ERPC-028
Title	Minimal Line Count
Objective	Confirm file line count is minimal (< 30 lines).
Preconditions	- Application running
Test Steps	1. Reflect class source file. 2. Count lines. 3. Verify count < 30.

Test Data	- Line count threshold: 30 lines
Expected Result	- File has less than 30 lines.
Actual Result	Line count is below threshold.
Status	Pass
Severity	Low

Test Case ID	ERPC-029
Title	Transformed Items Include All Provider Fields
Objective	Ensure each transformed item contains every provider field.
Preconditions	- Application running
Test Steps	1. Create dummy provider. 2. Transform via collection to array. 3. Verify all fields: id, driver, key, active, currencies, driver_config.
Test Data	- Provider fields expected in result
Expected Result	- Transformed array has all listed fields.
Actual Result	Fields present as expected.
Status	Pass
Severity	High

Test Case ID	ERPC-030
Title	Result Structure Is Valid Array
Objective	Verify that the transformation returns a valid array.
Preconditions	- Application running
Test Steps	1. Create dummy provider with typical values. 2. Transform and inspect result structure.
Test Data	- Provider: id=1, driver='currency_freak'
Expected Result	- Result is an array; is_array returns true
Actual Result	Result is a valid array.
Status	Pass
Severity	High

Test Case ID	ERPC-031
Title	toArray Method Has Documentation Comment
Objective	Ensure toArray method has PHP doc comment.
Preconditions	- Application running
Test Steps	1. ReflectionClass to get method's doc comment. 2. Check for existence of doc comment.
Test Data	- Method: toArray
Expected Result	- Doc comment is present.
Actual Result	Documentation comment found.
Status	Pass
Severity	Low

Test Case ID	ERPC-032
Title	toArray Method Return Type Documentation
Objective	Confirm that toArray method documentation includes '@return'.

Preconditions	- Application running
Test Steps	1. Get doc comment for 'toArray'. 2. Search for '@return' in doc comment.
Test Data	- Method doc comment
Expected Result	- '@return' tag present.
Actual Result	Return documentation tag present.
Status	Pass
Severity	Low

Test Case ID	ERPC-033
Title	toArray Method Parameter Documentation
Objective	Confirm that toArray method documentation includes '@param'.
Preconditions	- Application running
Test Steps	1. Get doc comment for 'toArray'. 2. Search for '@param' in doc comment.
Test Data	- Method doc comment
Expected Result	- '@param' tag present.
Actual Result	Parameter documentation tag found.
Status	Pass
Severity	Low

Test Case ID	ERPC-034
Title	Handles Active and Inactive Providers
Objective	Verify transformation correctly handles providers with different active statuses.
Preconditions	- Application running
Test Steps	1. Create two providers: - id=1, active=true - id=2, active=false 2. Transform via collection. 3. Inspect active statuses in result.
Test Data	- Providers with active: true, false
Expected Result	- result[0]['active'] == true - result[1]['active'] == false
Actual Result	Active statuses matched expected values.
Status	Pass
Severity	High

Test Case ID	ERPC-035
Title	Handles Different Currency Configurations
Objective	Ensure transformation maintains configured currencies for each provider.
Preconditions	- Application running
Test Steps	1. Create two providers with different currencies: - id=1, currencies=['USD', 'EUR'] - id=2, currencies=['GBP', 'JPY', 'AUD'] 2. Transform via collection. 3. Inspect currencies on each result.
Test Data	- Provider currencies: ['USD','EUR'] and ['GBP','JPY','AUD']

Expected Result	- result[0]['currencies'] == ['USD', 'EUR'] - result[1]['currencies'] == ['GBP', 'JPY', 'AUD']
Actual Result	Currency fields correctly handled.
Status	Pass
Severity	High

File: ExchangeRateProviderController-Test.txt

Test Case ID	ERPC-001
Title	Instantiate ExchangeRateProviderController
Objective	Verify that the ExchangeRateProviderController class can be instantiated.
Preconditions	- Application codebase available - Required dependencies are autoloaded
Test Steps	1. Attempt to instantiate an ExchangeRateProviderController object. 2. Assert that the instance is of type ExchangeRateProviderController.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- An object of type ExchangeRateProviderController is created successfully.
Actual Result	An object of type ExchangeRateProviderController was instantiated as expected.
Status	Pass
Severity	Medium

Test Case ID	ERPC-002
Title	Verify Controller Inheritance
Objective	Ensure ExchangeRateProviderController extends the base Controller class.
Preconditions	- Application codebase available - Required dependencies are autoloaded
Test Steps	1. Instantiate ExchangeRateProviderController. 2. Assert the instance is of type Crater\Http\Controllers\Controller.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- Instance is a subclass of Crater\Http\Controllers\Controller.
Actual Result	Instance is successfully recognized as a subclass of Controller.
Status	Pass
Severity	Medium

Test Case ID	ERPC-003
Title	Validate Namespace of Controller
Objective	Confirm the namespace of ExchangeRateProviderController.
Preconditions	- Application codebase available
Test Steps	1. Use ReflectionClass to inspect ExchangeRateProviderController. 2. Retrieve and check the namespace name.
Test Data	- Class: ExchangeRateProviderController - Expected Namespace: Crater\Http\Controllers\V1\Admin\ExchangeRate
Expected Result	- Namespace matches Crater\Http\Controllers\V1\Admin\ExchangeRate.
Actual Result	Namespace returned as Crater\Http\Controllers\V1\Admin\ExchangeRate.
Status	Pass
Severity	Low

Test Case ID	ERPC-004
Title	Check Abstract Status of Controller
Objective	Ensure ExchangeRateProviderController is not abstract.
Preconditions	- Application codebase available

Test Steps	1. Reflect on ExchangeRateProviderController. 2. Assert the class is not abstract.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- isAbstract returns false.
Actual Result	Class is confirmed to be non-abstract.
Status	Pass
Severity	Low

Test Case ID	ERPC-005
Title	Check if Controller is Instantiable
Objective	Ensure ExchangeRateProviderController can be instantiated via Reflection.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Assert the class is instantiable.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- isInstantiable returns true.
Actual Result	Class is instantiable.
Status	Pass
Severity	Low

Test Case ID	ERPC-006
Title	Validate Existence of index Method
Objective	Confirm that ExchangeRateProviderController implements an index method.
Preconditions	- Application codebase available
Test Steps	1. Instantiate ExchangeRateProviderController. 2. Check that method 'index' exists.
Test Data	- Method: index
Expected Result	- index method exists in the class.
Actual Result	index method is present in ExchangeRateProviderController.
Status	Pass
Severity	High

Test Case ID	ERPC-007
Title	Validate Existence of store Method
Objective	Confirm that ExchangeRateProviderController implements a store method.
Preconditions	- Application codebase available
Test Steps	1. Instantiate ExchangeRateProviderController. 2. Check that method 'store' exists.
Test Data	- Method: store
Expected Result	- store method exists in the class.
Actual Result	store method is present in ExchangeRateProviderController.
Status	Pass
Severity	High

Test Case ID	ERPC-008
Title	Validate Existence of show Method

Objective	Confirm that ExchangeRateProviderController implements a show method.
Preconditions	- Application codebase available
Test Steps	1. Instantiate ExchangeRateProviderController. 2. Check that method 'show' exists.
Test Data	- Method: show
Expected Result	- show method exists in the class.
Actual Result	show method is present in ExchangeRateProviderController.
Status	Pass
Severity	High

Test Case ID	ERPC-009
Title	Validate Existence of update Method
Objective	Confirm that ExchangeRateProviderController implements an update method.
Preconditions	- Application codebase available
Test Steps	1. Instantiate ExchangeRateProviderController. 2. Check that method 'update' exists.
Test Data	- Method: update
Expected Result	- update method exists in the class.
Actual Result	update method is present in ExchangeRateProviderController.
Status	Pass
Severity	High

Test Case ID	ERPC-010
Title	Validate Existence of destroy Method
Objective	Confirm that ExchangeRateProviderController implements a destroy method.
Preconditions	- Application codebase available
Test Steps	1. Instantiate ExchangeRateProviderController. 2. Check that method 'destroy' exists.
Test Data	- Method: destroy
Expected Result	- destroy method exists in the class.
Actual Result	destroy method is present in ExchangeRateProviderController.
Status	Pass
Severity	High

Test Case ID	ERPC-011
Title	Validate public access of index Method
Objective	Ensure index method is declared as public.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'index' method. 3. Check if method is public.
Test Data	- Method: index
Expected Result	- index method is public.
Actual Result	index method confirmed as public.
Status	Pass
Severity	High

Test Case ID	ERPC-012
Title	Validate index Method Parameter Type
Objective	Ensure index method accepts a Request parameter.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'index' method parameters. 3. Check parameter count and name.
Test Data	- Method: index - Parameter: request
Expected Result	- index method accepts one parameter named 'request'.
Actual Result	index method accepts one parameter named 'request'.
Status	Pass
Severity	High

Test Case ID	ERPC-013
Title	Validate index Method Non-static Status
Objective	Ensure index method is not static.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'index' method. 3. Assert method is not static.
Test Data	- Method: index
Expected Result	- index method is not static.
Actual Result	index method is not static.
Status	Pass
Severity	High

Test Case ID	ERPC-014
Title	Validate public access of store Method
Objective	Ensure store method is declared as public.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'store' method. 3. Check if method is public.
Test Data	- Method: store
Expected Result	- store method is public.
Actual Result	store method is public.
Status	Pass
Severity	High

Test Case ID	ERPC-015
Title	Validate store Method Parameter Type
Objective	Ensure store method accepts an ExchangeRateProviderRequest parameter named 'request'.
Preconditions	- Application codebase available

Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'store' method parameters. 3. Check parameter count and name.
Test Data	- Method: store - Parameter: request
Expected Result	- store method accepts one parameter named 'request'.
Actual Result	store method accepts one parameter named 'request'.
Status	Pass
Severity	High

Test Case ID	ERPC-016
Title	Validate store Method Non-static Status
Objective	Ensure store method is not static.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'store' method. 3. Assert method is not static.
Test Data	- Method: store
Expected Result	- store method is not static.
Actual Result	store method is not static.
Status	Pass
Severity	High

Test Case ID	ERPC-017
Title	Validate public access of show Method
Objective	Ensure show method is declared as public.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'show' method. 3. Check if method is public.
Test Data	- Method: show
Expected Result	- show method is public.
Actual Result	show method is public.
Status	Pass
Severity	High

Test Case ID	ERPC-018
Title	Validate show Method Parameter Type
Objective	Ensure show method accepts an ExchangeRateProvider parameter named 'exchangeRateProvider'.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'show' method parameters. 3. Check parameter count and name.
Test Data	- Method: show - Parameter: exchangeRateProvider
Expected Result	- show method accepts one parameter named 'exchangeRateProvider'.
Actual Result	show method accepts one parameter named 'exchangeRateProvider'.

Status	Pass
Severity	High

Test Case ID	ERPC-019
Title	Validate show Method Non-static Status
Objective	Ensure show method is not static.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'show' method. 3. Assert method is not static.
Test Data	- Method: show
Expected Result	- show method is not static.
Actual Result	show method is not static.
Status	Pass
Severity	High

Test Case ID	ERPC-020
Title	Validate public access of update Method
Objective	Ensure update method is declared as public.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'update' method. 3. Check if method is public.
Test Data	- Method: update
Expected Result	- update method is public.
Actual Result	update method is public.
Status	Pass
Severity	High

Test Case ID	ERPC-021
Title	Validate update Method Parameters
Objective	Ensure update method accepts two parameters: 'request' and 'exchangeRateProvider'.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'update' method parameters. 3. Check parameter count and names.
Test Data	- Method: update - Parameters: request, exchangeRateProvider
Expected Result	- update method accepts two parameters: 'request' and 'exchangeRateProvider'.
Actual Result	update method parameters are named as expected.
Status	Pass
Severity	High

Test Case ID	ERPC-022
Title	Validate update Method Non-static Status

Objective	Ensure update method is not static.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'update' method. 3. Assert method is not static.
Test Data	- Method: update
Expected Result	- update method is not static.
Actual Result	update method is not static.
Status	Pass
Severity	High

Test Case ID	ERPC-023
Title	Validate public access of destroy Method
Objective	Ensure destroy method is declared as public.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'destroy' method. 3. Check if method is public.
Test Data	- Method: destroy
Expected Result	- destroy method is public.
Actual Result	destroy method is public.
Status	Pass
Severity	High

Test Case ID	ERPC-024
Title	Validate destroy Method Parameter
Objective	Ensure destroy method accepts an ExchangeRateProvider parameter named 'exchangeRateProvider'.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'destroy' method parameters. 3. Check parameter count and name.
Test Data	- Method: destroy - Parameter: exchangeRateProvider
Expected Result	- destroy method accepts one parameter named 'exchangeRateProvider'.
Actual Result	Parameter is correctly named and typed.
Status	Pass
Severity	High

Test Case ID	ERPC-025
Title	Validate destroy Method Non-static Status
Objective	Ensure destroy method is not static.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'destroy' method. 3. Assert method is not static.
Test Data	- Method: destroy
Expected Result	- destroy method is not static.

Actual Result	destroy method is not static.
Status	Pass
Severity	High

Test Case ID	ERPC-026
Title	Create Multiple Controller Instances
Objective	Ensure multiple instances of ExchangeRateProviderController can exist and are independent.
Preconditions	- Application codebase available
Test Steps	1. Instantiate two ExchangeRateProviderController objects. 2. Assert both are instances of ExchangeRateProviderController. 3. Assert objects are not identical.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- Two distinct instances are created.
Actual Result	Two instances created and are not equal.
Status	Pass
Severity	Low

Test Case ID	ERPC-027
Title	Clone Controller Instance
Objective	Verify that ExchangeRateProviderController object can be cloned.
Preconditions	- Application codebase available
Test Steps	1. Instantiate ExchangeRateProviderController. 2. Clone the object. 3. Assert the cloned object is of the same type but not identical.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- Cloned object is a new instance and different from the original.
Actual Result	Cloned instance matches expected criteria.
Status	Pass
Severity	Low

Test Case ID	ERPC-028
Title	Use Controller in Type Hinting
Objective	Ensure ExchangeRateProviderController can be used in PHP type hinting.
Preconditions	- Application codebase available
Test Steps	1. Define a function accepting ExchangeRateProviderController as parameter. 2. Instantiate ExchangeRateProviderController. 3. Pass to the function and return result. 4. Assert the result is the same object.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- Type hinting works without error.
Actual Result	Function returned same controller as passed.
Status	Pass
Severity	Low

Test Case ID	ERPC-029
Title	Check if Controller is Not Final

Objective	Ensure ExchangeRateProviderController class is not declared as final.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Check final status.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- isFinal returns false.
Actual Result	Class is not final as expected.
Status	Pass
Severity	Low

Test Case ID	ERPC-030
Title	Check if Controller is Not an Interface
Objective	Ensure ExchangeRateProviderController is not an interface.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Check isInterface.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- isInterface returns false.
Actual Result	Class is not an interface.
Status	Pass
Severity	Low

Test Case ID	ERPC-031
Title	Check if Controller is Not a Trait
Objective	Ensure ExchangeRateProviderController is not a trait.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Check isTrait.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- isTrait returns false.
Actual Result	Class is not a trait.
Status	Pass
Severity	Low

Test Case ID	ERPC-032
Title	Check if Controller Class is Loaded
Objective	Ensure ExchangeRateProviderController class is autoloaded and accessible.
Preconditions	- Application codebase available
Test Steps	1. Verify class_exists for ExchangeRateProviderController.
Test Data	- Class: ExchangeRateProviderController
Expected Result	- class_exists returns true.
Actual Result	class_exists returned true.
Status	Pass
Severity	Low

Test Case ID	ERPC-033
Title	Validate Required Imports in Controller
Objective	Confirm that ExchangeRateProviderController imports all required classes.
Preconditions	- Application codebase available
Test Steps	1. Reflect file name of ExchangeRateProviderController. 2. Read file contents. 3. Assert 'use' statements for all required classes exist.
Test Data	- Required imports: Controller, ExchangeRateProviderRequest, ExchangeRateProviderResource, ExchangeRateProvider, Illuminate\Http\Request
Expected Result	- File contains all specified 'use' statements.
Actual Result	All required imports were found.
Status	Pass
Severity	Medium

Test Case ID	ERPC-034
Title	Validate File Structure of Controller
Objective	Ensure file structure and method signatures match expectations.
Preconditions	- Application codebase available
Test Steps	1. Reflect file name of ExchangeRateProviderController. 2. Read file contents. 3. Check for class declaration and all public method signatures.
Test Data	- Expected structure: class declaration and five public methods.
Expected Result	- File matches expected structure.
Actual Result	File contained expected class declaration and methods.
Status	Pass
Severity	Medium

Test Case ID	ERPC-035
Title	Check Reasonable Line Count of Controller
Objective	Ensure controller file length is appropriate (between 50 and 200 lines).
Preconditions	- Application codebase available
Test Steps	1. Read ExchangeRateProviderController file. 2. Count lines. 3. Assert line count is >50 and <200.
Test Data	- Expected line count: >50, <200
Expected Result	- File has line count within specified bounds.
Actual Result	Line count was within valid range.
Status	Pass
Severity	Low

Test Case ID	ERPC-036
Title	Check for Documentation on index Method
Objective	Verify that index method has documentation comment.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'index' method. 3. Check getDocComment does not return false.

Test Data	- Method: index
Expected Result	- Documentation comment exists for index method.
Actual Result	Documentation found on index method.
Status	Pass
Severity	Medium

Test Case ID	ERPC-037
Title	Check for Documentation on store Method
Objective	Verify that store method has documentation comment.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'store' method. 3. Check getDocComment does not return false.
Test Data	- Method: store
Expected Result	- Documentation comment exists for store method.
Actual Result	Documentation found on store method.
Status	Pass
Severity	Medium

Test Case ID	ERPC-038
Title	Check for Documentation on show Method
Objective	Verify that show method has documentation comment.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'show' method. 3. Check getDocComment does not return false.
Test Data	- Method: show
Expected Result	- Documentation comment exists for show method.
Actual Result	Documentation found on show method.
Status	Pass
Severity	Medium

Test Case ID	ERPC-039
Title	Check for Documentation on update Method
Objective	Verify that update method has documentation comment.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'update' method. 3. Check getDocComment does not return false.
Test Data	- Method: update
Expected Result	- Documentation comment exists for update method.
Actual Result	Documentation found on update method.
Status	Pass
Severity	Medium

Test Case ID	ERPC-040

Title	Check for Documentation on destroy Method
Objective	Verify that destroy method has documentation comment.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get 'destroy' method. 3. Check getDocComment does not return false.
Test Data	- Method: destroy
Expected Result	- Documentation comment exists for destroy method.
Actual Result	Documentation found on destroy method.
Status	Pass
Severity	Medium

Test Case ID	ERPC-041
Title	Check Authorization in index Method
Objective	Ensure index method uses Laravel's authorize function.
Preconditions	- Application codebase available
Test Steps	1. Reflect file name of ExchangeRateProviderController. 2. Check file content for $this->authorize('viewAny', ExchangeRateProvider::class).
Test Data	- Authorization call in index method.
Expected Result	- Authorization code present in index method.
Actual Result	Authorization found in index method.
Status	Pass
Severity	High

Test Case ID	ERPC-042
Title	Check whereCompany Scope Usage in index Method
Objective	Ensure index method uses whereCompany scope for ExchangeRateProvider.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Ensure 'ExchangeRateProvider::whereCompany()' is present.
Test Data	- Method: index
Expected Result	- whereCompany scope used in index method.
Actual Result	whereCompany scope verified.
Status	Pass
Severity	High

Test Case ID	ERPC-043
Title	Check paginate Usage in index Method
Objective	Ensure pagination is implemented in index method.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for '->paginate($limit)' in index method.
Test Data	- Method: index
Expected Result	- Pagination is present in index method.
Actual Result	Pagination found as expected.

Status	Pass
Severity	High

Test Case ID	ERPC-044
Title	Check index Method Returns Resource Collection
Objective	Ensure index method returns an ExchangeRateProviderResource collection.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Ensure 'ExchangeRateProviderResource::collection' is used.
Test Data	- Method: index
Expected Result	- index returns a resource collection.
Actual Result	Resource collection verified.
Status	Pass
Severity	High

Test Case ID	ERPC-045
Title	Check checkActiveCurrencies Usage in store Method
Objective	Ensure store method uses checkActiveCurrencies method.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for 'ExchangeRateProvider::checkActiveCurrencies' in store method.
Test Data	- Method: store
Expected Result	- checkActiveCurrencies is called.
Actual Result	checkActiveCurrencies used in store method.
Status	Pass
Severity	High

Test Case ID	ERPC-046
Title	Check checkExchangeRateProviderStatus Usage in store Method
Objective	Ensure store method uses checkExchangeRateProviderStatus method.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for 'ExchangeRateProvider::checkExchangeRateProviderStatus' in store method.
Test Data	- Method: store
Expected Result	- checkExchangeRateProviderStatus is called.
Actual Result	Function present in store method.
Status	Pass
Severity	High

Test Case ID	ERPC-047
Title	Check createFromRequest Usage in store Method
Objective	Ensure store method uses createFromRequest static method.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for 'ExchangeRateProvider::createFromRequest' in store method.

Test Data	- Method: store
Expected Result	- createFromRequest is called.
Actual Result	createFromRequest found in store method.
Status	Pass
Severity	High

Test Case ID	ERPC-048
Title	Show Method Returns Resource
Objective	Ensure show method returns an instance of ExchangeRateProviderResource.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for 'new ExchangeRateProviderResource($exchangeRateProvider)' in show method.
Test Data	- Method: show
Expected Result	- show returns an ExchangeRateProviderResource.
Actual Result	Resource returned as expected.
Status	Pass
Severity	High

Test Case ID	ERPC-049
Title	Check checkUpdateActiveCurrencies Usage in update Method
Objective	Ensure update method calls checkUpdateActiveCurrencies.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for '$exchangeRateProvider->checkUpdateActiveCurrencies' in update method.
Test Data	- Method: update
Expected Result	- checkUpdateActiveCurrencies called in update method.
Actual Result	Method call present in update.
Status	Pass
Severity	High

Test Case ID	ERPC-050
Title	Check updateFromRequest Usage in update Method
Objective	Ensure update method calls updateFromRequest.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for '$exchangeRateProvider->updateFromRequest' in update method.
Test Data	- Method: update
Expected Result	- updateFromRequest called in update method.
Actual Result	updateFromRequest function found.
Status	Pass
Severity	High

Test Case ID	ERPC-051

Title	Check destroy Method Active Status Check
Objective	Ensure destroy method checks 'active' status of provider.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Look for '$exchangeRateProvider->active' in destroy method.
Test Data	- Method: destroy
Expected Result	- destroy checks active status before proceeding.
Actual Result	Active status check found.
Status	Pass
Severity	High

Test Case ID	ERPC-052
Title	Check delete Call in destroy Method
Objective	Ensure destroy method calls delete on ExchangeRateProvider.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for '$exchangeRateProvider->delete()' in destroy method.
Test Data	- Method: destroy
Expected Result	- delete() method is called.
Actual Result	delete() call present in destroy.
Status	Pass
Severity	High

Test Case ID	ERPC-053
Title	Check Success Response in destroy Method
Objective	Ensure destroy method returns success response on completion.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for "'success' => true" in destroy method.
Test Data	- Method: destroy
Expected Result	- Success response with 'success' => true is returned.
Actual Result	Success response found.
Status	Pass
Severity	High

Test Case ID	ERPC-054
Title	Ensure Authorization Used Across All Methods
Objective	Ensure that all CRUD methods use proper Laravel authorization.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for relevant '$this->authorize' calls for viewAny, create, view, update, and delete.
Test Data	- Authorization calls for CRUD
Expected Result	- All CRUD methods include appropriate authorization checks.
Actual Result	Authorization calls found in all relevant methods.
Status	Pass

Severity	High

Test Case ID	ERPC-055
Title	Store Method Handles "Currency used." Error
Objective	Ensure store method properly handles error "Currency used."
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for 'Currency used.' error handling in store method.
Test Data	- Error string: "Currency used."
Expected Result	- Store method returns/handles "Currency used." error when duplicate currency detected.
Actual Result	Error handling present in store method.
Status	Pass
Severity	High

Test Case ID	ERPC-056
Title	Update Method Handles "Currency used." Error
Objective	Ensure update method properly handles error "Currency used."
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Look for 'Currency used.' error handling in update method.
Test Data	- Error string: "Currency used." - Multiple instances checked
Expected Result	- Update method returns/handles "Currency used." error when duplicate currency detected.
Actual Result	Error handling present more than once as expected.
Status	Pass
Severity	High

Test Case ID	ERPC-057
Title	Destroy Method Handles "Provider Active." Error
Objective	Ensure destroy method handles active provider error "Provider Active."
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Look for 'Provider Active.' error handling in destroy method.
Test Data	- Error string: "Provider Active."
Expected Result	- "Provider Active." error returned when trying to delete an active provider.
Actual Result	Error handling present in destroy method.
Status	Pass
Severity	High

Test Case ID	ERPC-058
Title	Confirm Controller Parent Class is Controller
Objective	Ensure ExchangeRateProviderController has Controller as parent.
Preconditions	- Application codebase available

Test Steps	1. Reflect on ExchangeRateProviderController. 2. Get parent class. 3. Validate parent class name.
Test Data	- Expected parent: Crater\Http\Controllers\Controller
Expected Result	- Parent is Crater\Http\Controllers\Controller.
Actual Result	Parent class confirmed as Controller.
Status	Pass
Severity	High

Test Case ID	ERPC-059
Title	Verify Controller Has Exactly 5 Public Methods
Objective	Ensure ExchangeRateProviderController only exposes 5 public methods: index, store, show, update, destroy.
Preconditions	- Application codebase available
Test Steps	1. Reflect on ExchangeRateProviderController. 2. List all public methods. 3. Filter out inherited methods. 4. Count own public methods.
Test Data	- Expected public methods: 5
Expected Result	- Exactly 5 own public methods found.
Actual Result	Five public methods found as expected.
Status	Pass
Severity	High

Test Case ID	ERPC-060
Title	Check for respondJson Helper Usage
Objective	Confirm controller uses respondJson helper for API responses.
Preconditions	- Application codebase available
Test Steps	1. Read controller file contents. 2. Check for 'respondJson' usage.
Test Data	- Helper: respondJson
Expected Result	- Controller uses respondJson helper method.
Actual Result	respondJson found in controller.
Status	Pass
Severity	High

File: ExchangeRateProviderRequest-Test.txt

Test Case ID	ERPR-001
Title	Authorization Always Succeeds for ExchangeRateProviderRequest
Objective	Verify that the authorize() method of ExchangeRateProviderRequest always returns true, ensuring that any user is authorized to make the request.
Preconditions	- Application is running - Crater\Http\Requests\ExchangeRateProviderRequest class is available - No specific user authentication required
Test Steps	1. Instantiate a new ExchangeRateProviderRequest object. 2. Call the authorize() method on the object. 3. Verify that the method returns true.
Test Data	- Request object with no additional properties set
Expected Result	- The authorize() method returns true.
Actual Result	- The authorize() method returned true as expected.
Status	Pass
Severity	High

Test Case ID	ERPR-002
Title	Validation Rules Are Returned Correctly by rules() Method
Objective	Ensure that the rules() method of ExchangeRateProviderRequest returns the exact validation rules as defined.
Preconditions	- Application is running - Crater\Http\Requests\ExchangeRateProviderRequest class is available
Test Steps	1. Instantiate a new ExchangeRateProviderRequest object. 2. Call the rules() method on the object. 3. Compare the returned rules to the expected rules array.
Test Data	- Expected rules: 'driver' => ['required'], 'key' => ['required'], 'currencies' => ['nullable'], 'currencies.*' => ['nullable'], 'driver_config' => ['nullable'], 'active' => ['nullable', 'boolean']
Expected Result	- The rules() method returns an array exactly matching the expected validation rules.
Actual Result	- The rules() method returned the expected rules array.
Status	Pass
Severity	High

Test Case ID	ERPR-003
Title	Payload Includes Validated Data and company_id from Header
Objective	Verify that getExchangeRateProviderPayload() correctly merges validated request data with company_id from the request header.
Preconditions	- Application is running - Database seeded with test companies - Crater\Http\Requests\ExchangeRateProviderRequest class is available - Mocked request object supports validated() and header() methods

Test Steps	1. Mock ExchangeRateProviderRequest and make partial. 2. Set validated() to return: - driver: 'some_driver' - key: 'some_key' - currencies: ['USD', 'EUR'] - active: true 3. Set header('company') to return 123. 4. Call getExchangeRateProviderPayload(). 5. Assert that the returned payload matches the expected array.
Test Data	- validatedData: driver: 'some_driver' key: 'some_key' currencies: ['USD', 'EUR'] active: true - companyId: 123
Expected Result	- getExchangeRateProviderPayload() returns: ['driver' => 'some_driver', 'key' => 'some_key', 'currencies' => ['USD', 'EUR'], 'active' => true, 'company_id' => 123]
Actual Result	- Returned payload matched expected data including company_id.
Status	Pass
Severity	High

Test Case ID	ERPR-004
Title	Payload Handles Empty Validated Data Gracefully
Objective	Ensure getExchangeRateProviderPayload() returns only company_id if validated data is empty.
Preconditions	- Application is running - Crater\Http\Requests\ExchangeRateProviderRequest class is available - Mocked request object supports validated() and header() methods
Test Steps	1. Mock ExchangeRateProviderRequest and make partial. 2. Set validated() to return empty array. 3. Set header('company') to return 456. 4. Call getExchangeRateProviderPayload(). 5. Assert the returned payload only contains company_id.
Test Data	- validatedData: [] - companyId: 456
Expected Result	- getExchangeRateProviderPayload() returns: ['company_id' => 456]
Actual Result	- Returned payload contained only company_id as expected.
Status	Pass
Severity	High

Test Case ID	ERPR-005
Title	Payload Handles Null company_id from Header
Objective	Verify that getExchangeRateProviderPayload() includes company_id as null when header value is null.
Preconditions	- Application is running - Crater\Http\Requests\ExchangeRateProviderRequest class is available - Mocked request object supports validated() and header() methods

Test Steps	1. Mock ExchangeRateProviderRequest and make partial. 2. Set validated() to return: - driver: 'another_driver' - key: 'another_key' 3. Set header('company') to return null. 4. Call getExchangeRateProviderPayload(). 5. Assert the returned payload contains company_id as null.
Test Data	- validatedData: driver: 'another_driver' key: 'another_key' - companyId: null
Expected Result	- getExchangeRateProviderPayload() returns: ['driver' => 'another_driver', 'key' => 'another_key', 'company_id' => null]
Actual Result	- Returned payload included company_id as null as expected.
Status	Pass
Severity	High

Test Case ID	ERPR-006
Title	Payload Handles Complete Validated Data and Null company_id
Objective	Check that getExchangeRateProviderPayload() properly merges full validated data with null company_id.
Preconditions	- Application is running - Crater\Http\Requests\ExchangeRateProviderRequest class is available - Mocked request object supports validated() and header() methods
Test Steps	1. Mock ExchangeRateProviderRequest and make partial. 2. Set validated() to return: - driver: 'complete_driver' - key: 'complete_key' - currencies: ['AUD'] - driver_config: ['endpoint' => 'some_url'] - active: false 3. Set header('company') to return null. 4. Call getExchangeRateProviderPayload(). 5. Assert the returned payload matches full validated data with company_id null.
Test Data	- validatedData: driver: 'complete_driver' key: 'complete_key' currencies: ['AUD'] driver_config: ['endpoint' => 'some_url'] active: false - companyId: null
Expected Result	- getExchangeRateProviderPayload() returns: ['driver' => 'complete_driver', 'key' => 'complete_key', 'currencies' => ['AUD'], 'driver_config' => ['endpoint' => 'some_url'], 'active' => false, 'company_id' => null]
Actual Result	- Returned payload incorporated full validated data and company_id as null.
Status	Pass
Severity	High

File: Expense-Test.txt

Test Case ID	E-001
Title	ExpenseCollection instantiation
Objective	Verify that ExpenseCollection can be instantiated properly.
Preconditions	- Application running - ExpenseCollection class available - Illuminate Collection class available
Test Steps	1. Create an empty Illuminate\Support\Collection. 2. Instantiate ExpenseCollection with the empty collection.
Test Data	- Input: Empty Collection [] - Expected object: ExpenseCollection instance
Expected Result	ExpenseCollection is successfully instantiated and is an instance of ExpenseCollection.
Actual Result	ExpenseCollection is instantiated as expected.
Status	Pass
Severity	Medium

Test Case ID	E-002
Title	ExpenseCollection inheritance from ResourceCollection
Objective	Ensure ExpenseCollection extends Illuminate\Http\Resources\Json\ResourceCollection.
Preconditions	- Application running - ExpenseCollection and ResourceCollection classes available
Test Steps	1. Create an empty Illuminate\Support\Collection. 2. Instantiate ExpenseCollection with the collection. 3. Assert that the instance is of type ResourceCollection.
Test Data	- Input: Empty Collection [] - Expected object: ResourceCollection inheritance
Expected Result	ExpenseCollection instance is also a ResourceCollection.
Actual Result	ExpenseCollection is a ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	E-003
Title	ExpenseCollection correct namespace verification
Objective	Check that ExpenseCollection is in 'Crater\Http\Resources' namespace.
Preconditions	- Application running - ExpenseCollection class available
Test Steps	1. Use ReflectionClass to reflect ExpenseCollection. 2. Retrieve namespace from reflection. 3. Verify the namespace is 'Crater\Http\Resources'.
Test Data	- Input class: ExpenseCollection - Namespace: 'Crater\Http\Resources'
Expected Result	ExpenseCollection resides in 'Crater\Http\Resources'.
Actual Result	Namespace is correct as 'Crater\Http\Resources'.
Status	Pass
Severity	Low

Test Case ID	E-004
Title	ExpenseCollection handles empty collection in toArray
Objective	Confirm that toArray handles empty collections and returns empty array.
Preconditions	- Application running - ExpenseCollection class available
Test Steps	1. Create an empty Collection. 2. Instantiate ExpenseCollection. 3. Invoke toArray with a fresh Request. 4. Assert result is an empty array.
Test Data	- Input: Empty Collection [] - Expected output: []
Expected Result	toArray returns an empty array.
Actual Result	Empty array returned.
Status	Pass
Severity	Medium

Test Case ID	E-005
Title	ExpenseCollection transforms single ExpenseResource
Objective	Verify ExpenseCollection transforms a single ExpenseResource correctly.
Preconditions	- Application running - ExpenseResource and ExpenseCollection classes available
Test Steps	1. Create dummy expense with ID 1 and amount 100. 2. Wrap expense in ExpenseResource. 3. Create a collection with the resource. 4. Instantiate ExpenseCollection with this collection. 5. Call toArray with a fresh Request. 6. Check that output is an array of size 1. 7. Check output contains key 'id' and that 'id' equals 1.
Test Data	- Input: ExpenseResource wrapping expense {id: 1, amount: 100} - Expected output: [{ 'id': 1, ... }]
Expected Result	Array of size 1 with first element having key 'id' equal to 1.
Actual Result	One-element array with correct ID returned.
Status	Pass
Severity	High

Test Case ID	E-006
Title	ExpenseCollection transforms multiple ExpenseResources
Objective	Ensure ExpenseCollection transforms multiple ExpenseResource objects.
Preconditions	- Application running - ExpenseResource and ExpenseCollection classes available
Test Steps	1. Create dummy expense with ID 1, amount 100. 2. Create dummy expense with ID 2, amount 200. 3. Wrap each in ExpenseResource. 4. Add both to Collection. 5. Instantiate ExpenseCollection with collection. 6. Call toArray with a fresh Request. 7. Assert array of size 2, first 'id' is 1, second 'id' is 2.
Test Data	- Input: [{id: 1, amount: 100}, {id: 2, amount: 200}] - Expected output: [{'id': 1, ...}, {'id': 2, ...}]
Expected Result	Array of size 2: first element 'id' = 1, second 'id' = 2.
Actual Result	Array with two elements, IDs 1 and 2 as expected.

Status	Pass
Severity	High

Test Case ID	E-007
Title	ExpenseCollection delegates to parent toArray
Objective	Confirm that ExpenseCollection uses parent::toArray method in its implementation.
Preconditions	- Application running - ExpenseCollection class source accessible
Test Steps	1. Use ReflectionClass on ExpenseCollection. 2. Load source file with file_get_contents. 3. Search for 'parent::toArray' in file.
Test Data	- Source code: ExpenseCollection.php
Expected Result	'parent::toArray' is present in source file.
Actual Result	Found 'parent::toArray' in source file.
Status	Pass
Severity	Low

Test Case ID	E-008
Title	ExpenseResource instantiation
Objective	Verify that ExpenseResource can be instantiated.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense with ID 1, amount 100. 2. Instantiate ExpenseResource with expense.
Test Data	- Input: Expense with ID 1, amount 100
Expected Result	ExpenseResource is instantiated successfully.
Actual Result	ExpenseResource instantiated as expected.
Status	Pass
Severity	Medium

Test Case ID	E-009
Title	ExpenseResource inheritance from JsonResource
Objective	Ensure ExpenseResource extends JsonResource.
Preconditions	- Application running - ExpenseResource and JsonResource classes available
Test Steps	1. Create dummy expense. 2. Instantiate ExpenseResource. 3. Check type inheritance from JsonResource.
Test Data	- Input: ExpenseResource wrapping expense
Expected Result	ExpenseResource is an instance of JsonResource.
Actual Result	ExpenseResource is of type JsonResource.
Status	Pass
Severity	Medium

Test Case ID	E-010
Title	ExpenseResource correct namespace

Objective	Check that ExpenseResource is in correct namespace.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Use ReflectionClass on ExpenseResource. 2. Retrieve its namespace. 3. Assert namespace is 'Crater\Http\Resources'.
Test Data	- Input: ExpenseResource - Namespace: 'Crater\Http\Resources'
Expected Result	Namespace is 'Crater\Http\Resources'.
Actual Result	Namespace is correct.
Status	Pass
Severity	Low

Test Case ID	E-011
Title	ExpenseResource toArray returns all required fields
Objective	Verify toArray populates all required keys for an expense.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense with ID 1, amount 100. 2. Wrap in ExpenseResource. 3. Invoke toArray with a Request. 4. Ensure result array contains all expected keys.
Test Data	- Input: Expense with all dummy fields - Expected keys: id, expense_date, amount, notes, customer_id, attachment_receipt_url, attachment_receipt, attachment_receipt_meta, company_id, expense_category_id, creator_id, formatted_expense_date, formatted_created_at, exchange_rate, currency_id, base_amount, payment_method_id
Expected Result	All required keys are present in toArray output.
Actual Result	All expected keys found in output.
Status	Pass
Severity	High

Test Case ID	E-012
Title	ExpenseResource toArray returns correct id
Objective	Confirm toArray returns correct expense ID.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense with ID 42, amount 100. 2. Wrap in ExpenseResource. 3. Get output from toArray. 4. Assert output['id'] == 42.
Test Data	- Input: Expense with ID 42 - Expected id: 42
Expected Result	Returned 'id' equals 42.
Actual Result	Output id is 42.
Status	Pass
Severity	High

Test Case ID	E-013

Title	ExpenseResource toArray returns correct amount
Objective	Validate that 'amount' field is set correctly.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense with ID 1, amount 250.75. 2. Wrap in ExpenseResource. 3. Call toArray, check 'amount' = 250.75.
Test Data	- Input: Expense with amount 250.75 - Expected amount: 250.75
Expected Result	Output['amount'] is 250.75.
Actual Result	Output amount is correct.
Status	Pass
Severity	High

Test Case ID	E-014
Title	ExpenseResource toArray returns correct expense_date
Objective	Ensure 'expense_date' field matches input date.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense with expense_date '2024-01-01'. 2. Wrap in ExpenseResource. 3. Call toArray; verify 'expense_date' == '2024-01-01'.
Test Data	- Input: expense_date = '2024-01-01' - Expected value: '2024-01-01'
Expected Result	Returned 'expense_date' is '2024-01-01'.
Actual Result	Output expense_date matches.
Status	Pass
Severity	High

Test Case ID	E-015
Title	ExpenseResource toArray handles zero amount
Objective	Verify toArray output when amount is zero.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense with amount 0. 2. Wrap in ExpenseResource. 3. Call toArray; check output amount is 0.
Test Data	- Input: amount = 0
Expected Result	Output['amount'] is 0.
Actual Result	Amount 0 returned as expected.
Status	Pass
Severity	High

Test Case ID	E-016
Title	ExpenseResource toArray includes customer field
Objective	Ensure output includes 'customer' field.
Preconditions	- Application running - ExpenseResource class available

Test Steps	1. Create dummy expense. 2. Wrap in ExpenseResource. 3. Get output array. 4. Check for 'customer' key in array.
Test Data	- Input: Dummy expense
Expected Result	Output includes key 'customer'.
Actual Result	'customer' field present.
Status	Pass
Severity	Medium

Test Case ID	E-017
Title	ExpenseResource toArray includes expense_category field
Objective	Ensure output includes 'expense_category' field.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense. 2. Wrap in ExpenseResource. 3. Get output array. 4. Check for 'expense_category' key.
Test Data	- Input: Dummy expense
Expected Result	Output includes 'expense_category' field.
Actual Result	'expense_category' key present.
Status	Pass
Severity	Medium

Test Case ID	E-018
Title	ExpenseResource toArray includes creator field
Objective	Ensure output includes 'creator' field.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense. 2. Wrap in ExpenseResource. 3. Get output array. 4. Check for 'creator' key.
Test Data	- Input: Dummy expense
Expected Result	Output includes 'creator' field.
Actual Result	'creator' field present.
Status	Pass
Severity	Medium

Test Case ID	E-019
Title	ExpenseResource toArray includes fields field
Objective	Ensure output contains 'fields' key.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense. 2. Wrap in ExpenseResource. 3. Call toArray and check for 'fields'.
Test Data	- Input: Dummy expense

Expected Result	Output includes 'fields' field.
Actual Result	'fields' field found.
Status	Pass
Severity	Medium

Test Case ID	E-020
Title	ExpenseResource toArray includes company field
Objective	Ensure output includes 'company' field.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense. 2. Wrap in ExpenseResource. 3. Call toArray and check for 'company'.
Test Data	- Input: Dummy expense
Expected Result	Output includes 'company' field.
Actual Result	'company' field present.
Status	Pass
Severity	Medium

Test Case ID	E-021
Title	ExpenseResource toArray includes currency field
Objective	Ensure output includes 'currency' key.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense. 2. Wrap in ExpenseResource. 3. Call toArray, check for 'currency' key.
Test Data	- Input: Dummy expense
Expected Result	Output includes 'currency' field.
Actual Result	'currency' field found.
Status	Pass
Severity	Medium

Test Case ID	E-022
Title	ExpenseResource toArray includes payment_method field
Objective	Ensure 'payment_method' key exists in output.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Create dummy expense. 2. Wrap in ExpenseResource. 3. Call toArray, check for 'payment_method'.
Test Data	- Input: Dummy expense
Expected Result	'payment_method' key present in output.
Actual Result	'payment_method' field present.
Status	Pass
Severity	Medium

Test Case ID	E-023
Title	ExpenseRequest instantiation
Objective	Verify that an ExpenseRequest can be created.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Instantiate ExpenseRequest.
Test Data	- Input: Creation of new ExpenseRequest
Expected Result	Object is an instance of ExpenseRequest.
Actual Result	Instance created successfully.
Status	Pass
Severity	Medium

Test Case ID	E-024
Title	ExpenseRequest inheritance from FormRequest
Objective	Ensure ExpenseRequest extends FormRequest.
Preconditions	- Application running - ExpenseRequest and FormRequest classes available
Test Steps	1. Instantiate ExpenseRequest. 2. Assert it extends FormRequest.
Test Data	- Input: ExpenseRequest instance
Expected Result	ExpenseRequest is also a FormRequest.
Actual Result	Type assertion passes.
Status	Pass
Severity	Medium

Test Case ID	E-025
Title	ExpenseRequest correct namespace
Objective	Verify ExpenseRequest resides in 'Crater\Http\Requests'.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Reflect ExpenseRequest class. 2. Check namespace.
Test Data	- Input: ExpenseRequest - Namespace: 'Crater\Http\Requests'
Expected Result	Namespace matches 'Crater\Http\Requests'.
Actual Result	Namespace is correct.
Status	Pass
Severity	Low

Test Case ID	E-026
Title	ExpenseRequest has authorize method
Objective	Confirm ExpenseRequest defines authorize method.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Instantiate ExpenseRequest. 2. Check if 'authorize' method exists.
Test Data	- Input: ExpenseRequest instance

Expected Result	'authorize' method present.
Actual Result	authorize method exists.
Status	Pass
Severity	High

Test Case ID	E-027
Title	ExpenseRequest has rules method
Objective	Confirm rules() method exists.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Instantiate ExpenseRequest. 2. Check for 'rules' method existence.
Test Data	- Input: ExpenseRequest instance
Expected Result	'rules' method present.
Actual Result	rules method present.
Status	Pass
Severity	High

Test Case ID	E-028
Title	ExpenseRequest has getExpensePayload method
Objective	Ensure getExpensePayload is defined.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Instantiate ExpenseRequest. 2. Check for getExpensePayload method.
Test Data	- Input: ExpenseRequest instance
Expected Result	getExpensePayload method exists.
Actual Result	Method exists.
Status	Pass
Severity	High

Test Case ID	E-029
Title	ExpenseRequest authorize returns true
Objective	Confirm authorize method returns true.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Instantiate ExpenseRequest. 2. Call authorize(). 3. Check returned value.
Test Data	- Input: ExpenseRequest - Expected: true
Expected Result	authorize() returns true.
Actual Result	authorize returns true.
Status	Pass
Severity	High

Test Case ID	E-030

Title	ExpenseRequest authorize method is public
Objective	Ensure authorize method visibility is public.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Reflect ExpenseRequest class. 2. Get 'authorize' ReflectionMethod. 3. Check isPublic().
Test Data	- Input: ExpenseRequest Reflection
Expected Result	authorize is public.
Actual Result	authorize method is public.
Status	Pass
Severity	High

Test Case ID	E-031
Title	ExpenseRequest rules method is public
Objective	Ensure rules method visibility is public.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Reflect ExpenseRequest class. 2. Get 'rules' ReflectionMethod. 3. Check isPublic().
Test Data	- Input: ExpenseRequest Reflection
Expected Result	rules is public.
Actual Result	rules method is public.
Status	Pass
Severity	High

Test Case ID	E-032
Title	ExpenseRequest getExpensePayload method is public
Objective	Verify getExpensePayload method visibility is public.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Reflect ExpenseRequest class. 2. Find getExpensePayload ReflectionMethod. 3. Check isPublic().
Test Data	- Input: ExpenseRequest Reflection
Expected Result	getExpensePayload is public.
Actual Result	getExpensePayload method is public.
Status	Pass
Severity	High

Test Case ID	E-033
Title	ExpenseRequest rules validate expense_date as required
Objective	Confirm 'expense_date' rule includes 'required'.
Preconditions	- Application running - Source code accessible

Test Steps	1. Reflect ExpenseRequest class. 2. Read source file. 3. Search for "'expense_date' =>" and 'required'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Rule for 'expense_date' includes 'required'.
Actual Result	'required' is included for 'expense_date'.
Status	Pass
Severity	High

Test Case ID	E-034
Title	ExpenseRequest rules validate expense_category_id as required
Objective	Check 'expense_category_id' rule is 'required'.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Read ExpenseRequest source. 2. Search for "'expense_category_id' =>" and 'required'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Rule for 'expense_category_id' includes 'required'.
Actual Result	'required' is included.
Status	Pass
Severity	High

Test Case ID	E-035
Title	ExpenseRequest rules validate amount as required
Objective	Ensure 'amount' validation rule includes 'required'.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Read source for "'amount' =>" and 'required'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Rule includes 'required' for 'amount'.
Actual Result	Rule confirmed.
Status	Pass
Severity	High

Test Case ID	E-036
Title	ExpenseRequest rules validate currency_id as required
Objective	Confirm 'currency_id' validation includes 'required'.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Read source for "'currency_id' =>" and 'required'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Rule has 'required' for currency_id.
Actual Result	Rule is present.
Status	Pass
Severity	High

Test Case ID	E-037
Title	ExpenseRequest rules validate exchange_rate as nullable
Objective	Verify 'exchange_rate' rule contains 'nullable'.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Search source for "'exchange_rate' =>" and 'nullable'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Rule for 'exchange_rate' includes 'nullable'.
Actual Result	Rule confirmed.
Status	Pass
Severity	Medium

Test Case ID	E-038
Title	ExpenseRequest rules validate payment_method_id as nullable
Objective	Ensure 'payment_method_id' rule contains 'nullable'.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Search for "'payment_method_id' =>" and 'nullable'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Rule for 'payment_method_id' includes 'nullable'.
Actual Result	Rule is present.
Status	Pass
Severity	Medium

Test Case ID	E-039
Title	ExpenseRequest rules validate customer_id as nullable
Objective	Confirm 'customer_id' rule includes 'nullable'.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Search source file for "'customer_id' =>" and 'nullable'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Rule for 'customer_id' includes 'nullable'.
Actual Result	Rule included.
Status	Pass
Severity	Medium

Test Case ID	E-040
Title	ExpenseRequest rules validate notes as nullable
Objective	Ensure 'notes' rule contains 'nullable'.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Search source for "'notes' =>" and 'nullable'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Rule for 'notes' is 'nullable'.
Actual Result	Rule present.

Status	Pass
Severity	Low

Test Case ID	E-041
Title	ExpenseRequest rules include validation for attachment_receipt
Objective	Confirm attachment_receipt rule checks type, mimes, and max size.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Search for "'attachment_receipt' =>", 'file', mimes, and max.
Test Data	- Source: ExpenseRequest.php
Expected Result	Validation includes file, mimes (jpg,png,pdf,doc,docx,xls,xlsx,ppt,pptx), and max:20000.
Actual Result	Validation rule present.
Status	Pass
Severity	High

Test Case ID	E-042
Title	ExpenseRequest getExpensePayload uses CompanySetting
Objective	Ensure CompanySetting::getSetting is called in getExpensePayload.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Read source. 2. Search for 'CompanySetting::getSetting'.
Test Data	- Source: ExpenseRequest.php
Expected Result	getExpensePayload uses CompanySetting::getSetting.
Actual Result	Call is present in method.
Status	Pass
Severity	Medium

Test Case ID	E-043
Title	ExpenseRequest getExpensePayload uses collect helper
Objective	Confirm collect helper usage in getExpensePayload.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Read source. 2. Search for 'collect($this->validated())'.
Test Data	- Source: ExpenseRequest.php
Expected Result	Use of collect($this->validated()) in getExpensePayload.
Actual Result	collect helper called.
Status	Pass
Severity	Low

Test Case ID	E-044
Title	ExpenseRequest getExpensePayload merges additional fields
Objective	Confirm additional fields are merged in getExpensePayload.
Preconditions	- Application running - ExpenseRequest code accessible

Test Steps	1. Read source. 2. Search for merge(['creator_id', 'company_id', 'exchange_rate', 'base_amount', 'currency_id']).
Test Data	- Source: ExpenseRequest.php
Expected Result	Fields are merged in getExpensePayload.
Actual Result	Fields are merged as expected.
Status	Pass
Severity	High

Test Case ID	E-045
Title	ExpenseRequest getExpensePayload returns array
Objective	Ensure getExpensePayload finalizes by returning array.
Preconditions	- Application running - ExpenseRequest code accessible
Test Steps	1. Search source for '->toArray()' at end of getExpensePayload.
Test Data	- Source: ExpenseRequest.php
Expected Result	Method returns array using toArray().
Actual Result	Method returns array.
Status	Pass
Severity	High

Test Case ID	E-046
Title	All main classes are not abstract
Objective	Confirm ExpenseCollection, ExpenseResource, ExpenseRequest are not abstract.
Preconditions	- Application running - PHP ReflectionClass available
Test Steps	1. Reflect each class. 2. Check isAbstract() is false.
Test Data	- Input: ExpenseCollection, ExpenseResource, ExpenseRequest
Expected Result	None of the classes are abstract.
Actual Result	isAbstract() false for all classes.
Status	Pass
Severity	High

Test Case ID	E-047
Title	All main classes are loaded and available
Objective	Confirm ExpenseCollection, ExpenseResource, ExpenseRequest classes exist.
Preconditions	- Application running
Test Steps	1. Use class_exists on each class.
Test Data	- Input: ExpenseCollection, ExpenseResource, ExpenseRequest
Expected Result	class_exists returns true for all.
Actual Result	All classes loaded.
Status	Pass
Severity	High

Test Case ID	E-048
Title	ExpenseCollection toArray method has documentation
Objective	Verify that toArray method in ExpenseCollection is documented.
Preconditions	- Application running - ExpenseCollection class available
Test Steps	1. Reflect ExpenseCollection. 2. Get toArray ReflectionMethod. 3. Check for doc comment presence.
Test Data	- Input: ExpenseCollection toArray()
Expected Result	getDocComment() is not false.
Actual Result	Method is documented.
Status	Pass
Severity	Low

Test Case ID	E-049
Title	ExpenseResource toArray method has documentation
Objective	Ensure toArray method in ExpenseResource is documented.
Preconditions	- Application running - ExpenseResource class available
Test Steps	1. Reflect ExpenseResource. 2. Get toArray ReflectionMethod. 3. Check for doc comment.
Test Data	- Input: ExpenseResource toArray()
Expected Result	getDocComment() is not false.
Actual Result	Method documentation found.
Status	Pass
Severity	Low

Test Case ID	E-050
Title	ExpenseRequest authorize method has documentation
Objective	Ensure authorize method in ExpenseRequest is documented.
Preconditions	- Application running - ExpenseRequest class available
Test Steps	1. Reflect ExpenseRequest. 2. Get authorize ReflectionMethod. 3. Check for doc comment.
Test Data	- Input: ExpenseRequest authorize()
Expected Result	getDocComment() is not false.
Actual Result	Method is documented.
Status	Pass
Severity	Low

Test Case ID	E-051
Title	ExpenseRequest rules method has documentation
Objective	Ensure rules method in ExpenseRequest is documented.
Preconditions	- Application running - ExpenseRequest class available

Test Steps	1. Reflect ExpenseRequest. 2. Get rules ReflectionMethod. 3. Check for doc comment.
Test Data	- Input: ExpenseRequest rules()
Expected Result	getDocComment() is not false.
Actual Result	Method documentation found.
Status	Pass
Severity	Low

File: ExpenseCategoriesController-Test.txt

Test Case ID	ECC-001
Title	Instantiation of ExpenseCategoriesController
Objective	Verify that an instance of ExpenseCategoriesController can be created.
Preconditions	- Application running - Database seeded - Crater packages installed
Test Steps	1. Attempt to instantiate ExpenseCategoriesController. 2. Check if the created object is an instance of ExpenseCategoriesController.
Test Data	- Class: Crater\Http\Controllers\V1\Admin\Expense\ExpenseCategoriesController
Expected Result	- ExpenseCategoriesController instance is created successfully.
Actual Result	ExpenseCategoriesController instance created as expected.
Status	Pass
Severity	High

Test Case ID	ECC-002
Title	ExpenseCategoriesController extends Controller
Objective	Ensure that ExpenseCategoriesController inherits from the base Controller class.
Preconditions	- Application running - Database seeded
Test Steps	1. Instantiate ExpenseCategoriesController. 2. Check inheritance from Crater\Http\Controllers\Controller.
Test Data	- Class: ExpenseCategoriesController
Expected Result	- ExpenseCategoriesController is an instance of Controller.
Actual Result	ExpenseCategoriesController is confirmed to extend Controller.
Status	Pass
Severity	High

Test Case ID	ECC-003
Title	Namespace verification of ExpenseCategoriesController
Objective	Verify that ExpenseCategoriesController resides in the correct namespace.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on ExpenseCategoriesController. 2. Get and check the namespace name.
Test Data	- Namespace: Crater\Http\Controllers\V1\Admin\Expense
Expected Result	- The namespace is 'Crater\Http\Controllers\V1\Admin\Expense'.
Actual Result	Namespace correctly identified as 'Crater\Http\Controllers\V1\Admin\Expense'.
Status	Pass
Severity	Medium

Test Case ID	ECC-004
Title	ExpenseCategoriesController is not abstract
Objective	Ensure ExpenseCategoriesController is a concrete class, not abstract.

Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Verify isAbstract() returns false.
Test Data	- Class: ExpenseCategoriesController
Expected Result	- isAbstract() returns false.
Actual Result	Class is not abstract as expected.
Status	Pass
Severity	High

Test Case ID	ECC-005
Title	ExpenseCategoriesController is instantiable
Objective	Ensure that ExpenseCategoriesController can be instantiated.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Check isInstantiable() property.
Test Data	- Class: ExpenseCategoriesController
Expected Result	- isInstantiable() returns true.
Actual Result	Controller class is instantiable.
Status	Pass
Severity	High

Test Case ID	ECC-006
Title	Existence of 'index' method in ExpenseCategoriesController
Objective	Verify that 'index' method exists in ExpenseCategoriesController.
Preconditions	- Application running
Test Steps	1. Instantiate ExpenseCategoriesController. 2. Use method_exists() to confirm 'index' method.
Test Data	- Method: index
Expected Result	- 'index' method exists on ExpenseCategoriesController.
Actual Result	'index' method confirmed to exist.
Status	Pass
Severity	High

Test Case ID	ECC-007
Title	Existence of 'store' method in ExpenseCategoriesController
Objective	Verify that 'store' method exists.
Preconditions	- Application running
Test Steps	1. Instantiate ExpenseCategoriesController. 2. Use method_exists() to confirm the presence of 'store'.
Test Data	- Method: store
Expected Result	- 'store' method exists on ExpenseCategoriesController.
Actual Result	'store' method exists as expected.
Status	Pass
Severity	High

Test Case ID	ECC-008

Title	Existence of 'show' method in ExpenseCategoriesController
Objective	Verify that 'show' method exists in ExpenseCategoriesController.
Preconditions	- Application running
Test Steps	1. Instantiate ExpenseCategoriesController. 2. Use method_exists() to check for 'show'.
Test Data	- Method: show
Expected Result	- 'show' method exists on ExpenseCategoriesController.
Actual Result	'show' method exists as expected.
Status	Pass
Severity	High

Test Case ID	ECC-009
Title	Existence of 'update' method in ExpenseCategoriesController
Objective	Verify that 'update' method exists in ExpenseCategoriesController.
Preconditions	- Application running
Test Steps	1. Instantiate ExpenseCategoriesController. 2. Use method_exists() to check for 'update'.
Test Data	- Method: update
Expected Result	- 'update' method exists on ExpenseCategoriesController.
Actual Result	'update' method exists as expected.
Status	Pass
Severity	High

Test Case ID	ECC-010
Title	Existence of 'destroy' method in ExpenseCategoriesController
Objective	Verify that 'destroy' method exists in ExpenseCategoriesController.
Preconditions	- Application running
Test Steps	1. Instantiate ExpenseCategoriesController. 2. Use method_exists() to check for 'destroy'.
Test Data	- Method: destroy
Expected Result	- 'destroy' method exists on ExpenseCategoriesController.
Actual Result	'destroy' method exists as expected.
Status	Pass
Severity	High

Test Case ID	ECC-011
Title	'index' method is public
Objective	Ensure 'index' method has public visibility.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve 'index' method. 3. Check visibility using isPublic().
Test Data	- Method: index
Expected Result	- isPublic() returns true for 'index'.
Actual Result	'index' method is public.
Status	Pass

Severity	High

Test Case ID	ECC-012
Title	'index' method accepts Request parameter
Objective	Verify that 'index' method requires a Request parameter.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve parameters of 'index' method. 3. Check count and name of parameter.
Test Data	- Method: index - Parameter: request
Expected Result	- One parameter named 'request' is required.
Actual Result	Parameter 'request' accepted by index method.
Status	Pass
Severity	High

Test Case ID	ECC-013
Title	'index' method is not static
Objective	Ensure that the 'index' method is not declared static.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve 'index' method. 3. Check isStatic() value.
Test Data	- Method: index
Expected Result	- isStatic() returns false.
Actual Result	'index' method is not static.
Status	Pass
Severity	High

Test Case ID	ECC-014
Title	'store' method is public
Objective	Ensure 'store' method is declared public.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve 'store' method. 3. Validate with isPublic().
Test Data	- Method: store
Expected Result	- isPublic() returns true for 'store'.
Actual Result	'store' method is public.
Status	Pass
Severity	High

Test Case ID	ECC-015
Title	'store' method accepts ExpenseCategoryRequest parameter
Objective	Verify 'store' method requires an ExpenseCategoryRequest parameter.
Preconditions	- Application running

Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve parameters for 'store' method. 3. Check for single parameter named 'request'.
Test Data	- Method: store - Parameter: request (ExpenseCategoryRequest)
Expected Result	- 'store' method accepts one parameter: 'request'.
Actual Result	Correct parameter detected.
Status	Pass
Severity	High

Test Case ID	ECC-016
Title	'store' method is not static
Objective	Ensure 'store' method is not static.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve 'store' method. 3. Check isStatic().
Test Data	- Method: store
Expected Result	- isStatic() returns false.
Actual Result	'store' method is not static.
Status	Pass
Severity	High

Test Case ID	ECC-017
Title	'show' method is public
Objective	Ensure 'show' method is declared public.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve 'show' method. 3. Verify isPublic().
Test Data	- Method: show
Expected Result	- isPublic() returns true.
Actual Result	'show' method is public.
Status	Pass
Severity	High

Test Case ID	ECC-018
Title	'show' method accepts ExpenseCategory parameter
Objective	Ensure 'show' method requires an ExpenseCategory parameter.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve parameters for 'show' method. 3. Validate parameter name is 'category'.
Test Data	- Method: show - Parameter: category (ExpenseCategory)
Expected Result	- Parameter 'category' is accepted.
Actual Result	Parameter confirmed.
Status	Pass

Severity	High

Test Case ID	ECC-019
Title	'show' method is not static
Objective	Validate that 'show' method is not static.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Retrieve 'show' method. 3. Check isStatic().
Test Data	- Method: show
Expected Result	- isStatic() returns false.
Actual Result	'show' method is not static.
Status	Pass
Severity	High

Test Case ID	ECC-020
Title	'update' method is public
Objective	Validate public visibility of 'update' method.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Get 'update' method. 3. Confirm isPublic().
Test Data	- Method: update
Expected Result	- isPublic() is true.
Actual Result	'update' method is public.
Status	Pass
Severity	High

Test Case ID	ECC-021
Title	'update' method accepts two parameters
Objective	Ensure 'update' method requires two parameters: request and category.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Get parameters for 'update' method. 3. Confirm count is two and names are 'request' and 'category'.
Test Data	- Method: update - Parameters: request, category
Expected Result	- Two parameters: 'request' and 'category'.
Actual Result	Both parameters present as expected.
Status	Pass
Severity	High

Test Case ID	ECC-022
Title	'update' method is not static
Objective	Validate 'update' is not declared static.
Preconditions	- Application running

Test Steps	1. Reflect ExpenseCategoriesController. 2. Get 'update' method. 3. Confirm isStatic() value.
Test Data	- Method: update
Expected Result	- isStatic() returns false.
Actual Result	'update' is not static.
Status	Pass
Severity	High

Test Case ID	ECC-023
Title	'destroy' method is public
Objective	Confirm 'destroy' method public visibility.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Get 'destroy' method. 3. Check isPublic().
Test Data	- Method: destroy
Expected Result	- isPublic() returns true.
Actual Result	'destroy' method is public.
Status	Pass
Severity	High

Test Case ID	ECC-024
Title	'destroy' method accepts ExpenseCategory parameter
Objective	Confirm 'destroy' method accepts category parameter.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Get 'destroy' parameters. 3. Confirm single parameter 'category'.
Test Data	- Method: destroy - Parameter: category
Expected Result	- Parameter 'category' present.
Actual Result	Parameter present and correct.
Status	Pass
Severity	High

Test Case ID	ECC-025
Title	'destroy' method is not static
Objective	Validate 'destroy' is instance method.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Get 'destroy' method. 3. Check isStatic().
Test Data	- Method: destroy
Expected Result	- isStatic() returns false.
Actual Result	Method is not static.
Status	Pass

Severity	High

Test Case ID	ECC-026
Title	Multiple ExpenseCategoriesController instances can be created
Objective	Verify multiple instances can be created and are unique.
Preconditions	- Application running
Test Steps	1. Instantiate ExpenseCategoriesController twice. 2. Check that both are instances. 3. Ensure they are not the same object.
Test Data	- Two new ExpenseCategoriesController objects
Expected Result	- Each instance is unique and correctly typed.
Actual Result	Instances are unique and of correct type.
Status	Pass
Severity	Medium

Test Case ID	ECC-027
Title	ExpenseCategoriesController can be cloned
Objective	Verify that cloning the controller creates a new instance.
Preconditions	- Application running
Test Steps	1. Instantiate ExpenseCategoriesController. 2. Clone the object. 3. Ensure clone is a new, distinct instance.
Test Data	- ExpenseCategoriesController object
Expected Result	- Cloned object is distinct but same type.
Actual Result	Clone operation successful.
Status	Pass
Severity	Medium

Test Case ID	ECC-028
Title	ExpenseCategoriesController can be used in type hints
Objective	Confirm type hints compatibility for ExpenseCategoriesController.
Preconditions	- Application running
Test Steps	1. Create a function accepting ExpenseCategoriesController. 2. Pass instance to function. 3. Confirm type match on return.
Test Data	- Function accepting ExpenseCategoriesController
Expected Result	- Passed and returned instances match type.
Actual Result	Type hinting works as expected.
Status	Pass
Severity	Medium

Test Case ID	ECC-029
Title	ExpenseCategoriesController is not final
Objective	Confirm that ExpenseCategoriesController is not final.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Check isFinal() property.

Test Data	- Class: ExpenseCategoriesController
Expected Result	- isFinal() returns false.
Actual Result	Confirmed not final.
Status	Pass
Severity	Medium

Test Case ID	ECC-030
Title	ExpenseCategoriesController is not an interface
Objective	Ensure ExpenseCategoriesController is a class and not an interface.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Check isInterface() status.
Test Data	- Class: ExpenseCategoriesController
Expected Result	- isInterface() returns false.
Actual Result	Confirmed not an interface.
Status	Pass
Severity	Medium

Test Case ID	ECC-031
Title	ExpenseCategoriesController is not a trait
Objective	Ensure ExpenseCategoriesController is a class, not a trait.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController. 2. Check isTrait() value.
Test Data	- Class: ExpenseCategoriesController
Expected Result	- isTrait() returns false.
Actual Result	Confirmed not a trait.
Status	Pass
Severity	Medium

Test Case ID	ECC-032
Title	ExpenseCategoriesController class is loaded
Objective	Verify class is loaded and available for use.
Preconditions	- Application running
Test Steps	1. Use class_exists on ExpenseCategoriesController. 2. Confirm result is true.
Test Data	- Class: ExpenseCategoriesController
Expected Result	- class_exists returns true.
Actual Result	Class is loaded.
Status	Pass
Severity	High

Test Case ID	ECC-033
Title	ExpenseCategoriesController uses required classes
Objective	Validate required class imports in ExpenseCategoriesController.

Preconditions	- Application files exist
Test Steps	1. Reflect controller to get file content. 2. Check for 'use' statements for required classes.
Test Data	- Imports: Controller, ExpenseCategoryRequest, ExpenseCategoryResource, ExpenseCategory, Request
Expected Result	- All imports present in file.
Actual Result	All imports detected.
Status	Pass
Severity	Medium

Test Case ID	ECC-034
Title	ExpenseCategoriesController file has expected structure
Objective	Ensure file contains expected class and methods.
Preconditions	- Application files exist
Test Steps	1. Get file contents via reflection. 2. Check for class and method declarations.
Test Data	- Methods: index, store, show, update, destroy
Expected Result	- Class structure and method declarations present.
Actual Result	File structure confirmed.
Status	Pass
Severity	Medium

Test Case ID	ECC-035
Title	ExpenseCategoriesController has reasonable line count
Objective	Ensure source file is neither too small nor too large in line count.
Preconditions	- Application files exist
Test Steps	1. Reflect controller and retrieve file. 2. Count lines. 3. Confirm count is >50 and <150.
Test Data	- Source file line count
Expected Result	- Line count between 51 and 149.
Actual Result	Line count within acceptable range.
Status	Pass
Severity	Low

Test Case ID	ECC-036
Title	'index' method has documentation
Objective	Confirm that 'index' method includes doc comments.
Preconditions	- Application files exist
Test Steps	1. Reflect controller and get 'index'. 2. Check getDocComment() for presence.
Test Data	- Method: index
Expected Result	- Doc comment present.
Actual Result	Doc comment exists.
Status	Pass
Severity	Low

Test Case ID	ECC-037
Title	'store' method has documentation
Objective	Ensure documentation exists for 'store' method.
Preconditions	- Application files exist
Test Steps	1. Reflect controller and get 'store'. 2. Check for doc comment.
Test Data	- Method: store
Expected Result	- Documentation comment present.
Actual Result	Doc comment detected.
Status	Pass
Severity	Low

Test Case ID	ECC-038
Title	'show' method has documentation
Objective	Validate documentation presence for 'show' method.
Preconditions	- Application files exist
Test Steps	1. Reflect controller and get 'show'. 2. Verify doc comment.
Test Data	- Method: show
Expected Result	- Doc comment present.
Actual Result	Doc comment present.
Status	Pass
Severity	Low

Test Case ID	ECC-039
Title	'update' method has documentation
Objective	Confirm 'update' method has doc block.
Preconditions	- Application files exist
Test Steps	1. Reflect controller and get 'update'. 2. Check for doc comment.
Test Data	- Method: update
Expected Result	- Doc comment exists.
Actual Result	Doc comment confirmed.
Status	Pass
Severity	Low

Test Case ID	ECC-040
Title	'destroy' method has documentation
Objective	Ensure documentation is present for 'destroy' method.
Preconditions	- Application files exist
Test Steps	1. Reflect controller and get 'destroy'. 2. Validate doc comment.
Test Data	- Method: destroy
Expected Result	- Doc comment exists.
Actual Result	Documentation present.

Status	Pass
Severity	Low

Test Case ID	ECC-041
Title	'index' method uses authorize
Objective	Confirm authorization enforcement in 'index' method.
Preconditions	- Application files exist
Test Steps	1. Get 'index' method source. 2. Check for $this->authorize('viewAny', ExpenseCategory::class) call.
Test Data	- Method source
Expected Result	- Authorization called for 'viewAny'.
Actual Result	Authorization call present.
Status	Pass
Severity	High

Test Case ID	ECC-042
Title	'index' method uses applyFilters
Objective	Confirm use of applyFilters in 'index'.
Preconditions	- Application files exist
Test Steps	1. Check 'index' method source for call to ExpenseCategory::applyFilters.
Test Data	- Source: index method
Expected Result	- applyFilters used.
Actual Result	applyFilters usage confirmed.
Status	Pass
Severity	High

Test Case ID	ECC-043
Title	'index' method uses whereCompany scope
Objective	Verify 'whereCompany' scope used in index retrieval.
Preconditions	- Application files exist
Test Steps	1. Fetch 'index' source code. 2. Confirm '->whereCompany()' in returned query.
Test Data	- Source: index method
Expected Result	- whereCompany scope present.
Actual Result	Scope confirmed.
Status	Pass
Severity	High

Test Case ID	ECC-044
Title	'index' method uses latest
Objective	Validate that results are ordered by latest in 'index'.
Preconditions	- Application files exist
Test Steps	1. Review 'index' source. 2. Ensure '->latest()' used.
Test Data	- Source: index method

Expected Result	- Results are ordered by latest.
Actual Result	Ordering by latest confirmed.
Status	Pass
Severity	Medium

Test Case ID	ECC-045
Title	'index' method uses paginateData
Objective	Confirm result pagination in 'index'.
Preconditions	- Application files exist
Test Steps	1. Review 'index' method for '->paginateData($limit)'.
Test Data	- Source: index method
Expected Result	- Pagination with paginateData present.
Actual Result	Pagination present.
Status	Pass
Severity	Medium

Test Case ID	ECC-046
Title	'index' method returns resource collection
Objective	Confirm return type of resource collection in 'index'.
Preconditions	- Application files exist
Test Steps	1. Review 'index' for ExpenseCategoryResource::collection call.
Test Data	- Source: index method
Expected Result	- Returns collection wrapped with resource class.
Actual Result	Resource collection returned.
Status	Pass
Severity	Medium

Test Case ID	ECC-047
Title	'store' method uses getExpenseCategoryPayload
Objective	Ensure 'store' utilizes getExpenseCategoryPayload from request.
Preconditions	- Application files exist
Test Steps	1. Read 'store' source. 2. Check for $request->getExpenseCategoryPayload() call.
Test Data	- Source: store method
Expected Result	- getExpenseCategoryPayload used.
Actual Result	Usage confirmed.
Status	Pass
Severity	High

Test Case ID	ECC-048
Title	'store' method creates ExpenseCategory
Objective	Confirm that new ExpenseCategory is created in store method.
Preconditions	- Application files exist
Test Steps	1. Review 'store' method source. 2. Check for ExpenseCategory::create invocation.

Test Data	- Source: store method
Expected Result	- ExpenseCategory created on store.
Actual Result	Creation confirmed.
Status	Pass
Severity	High

Test Case ID	ECC-049
Title	'show' method returns ExpenseCategoryResource
Objective	Confirm correct return type in 'show' method.
Preconditions	- Application files exist
Test Steps	1. Review 'show' source for new ExpenseCategoryResource($category).
Test Data	- Source: show method
Expected Result	- Returns ExpenseCategoryResource instance.
Actual Result	Return value confirmed.
Status	Pass
Severity	Medium

Test Case ID	ECC-050
Title	'update' method calls update on category
Objective	Validate updating ExpenseCategory in 'update' method.
Preconditions	- Application files exist
Test Steps	1. Review 'update' method source code. 2. Check for $category->update call.
Test Data	- Source: update method
Expected Result	- Category update invoked.
Actual Result	Update method invoked as expected.
Status	Pass
Severity	High

Test Case ID	ECC-051
Title	'destroy' method checks expenses count
Objective	Ensure destroy method verifies if expenses are attached before deleting.
Preconditions	- Application files exist
Test Steps	1. Review 'destroy' source. 2. Check for $category->expenses()->count() usage.
Test Data	- Source: destroy method
Expected Result	- Expenses count checked before delete.
Actual Result	Expenses check occurs.
Status	Pass
Severity	High

Test Case ID	ECC-052
Title	'destroy' method calls delete
Objective	Confirm that destroy method deletes ExpenseCategory object.
Preconditions	- Application files exist

Test Steps	1. Inspect 'destroy' method for $category->delete() call.
Test Data	- Source: destroy method
Expected Result	- Deletion method invoked.
Actual Result	Delete method executed.
Status	Pass
Severity	High

Test Case ID	ECC-053
Title	'destroy' method returns success response
Objective	Validate success response generated on deletion.
Preconditions	- Application files exist
Test Steps	1. Inspect 'destroy' method for 'success' => true in response.
Test Data	- Source: destroy method
Expected Result	- Response includes 'success' => true.
Actual Result	Success response returned.
Status	Pass
Severity	High

Test Case ID	ECC-054
Title	All methods use authorization
Objective	Ensure authorization logic is present in all controller methods.
Preconditions	- Application files exist
Test Steps	1. Inspect ExpenseCategoriesController file for $this->authorize calls. 2. Confirm calls for all operations: viewAny, create, view, update, delete.
Test Data	- Source: controller methods
Expected Result	- Authorization present in all methods.
Actual Result	Authorization logic detected in all methods.
Status	Pass
Severity	High

Test Case ID	ECC-055
Title	'destroy' method handles expense_attached error
Objective	Confirm error handling for attached expenses in destroy method.
Preconditions	- Application files exist
Test Steps	1. Inspect 'destroy' for 'expense_attached' and 'Expense Attached'.
Test Data	- Error code: 'expense_attached', message: 'Expense Attached'
Expected Result	- Error handled and appropriate message returned.
Actual Result	Error handling in place.
Status	Pass
Severity	High

Test Case ID	ECC-056
Title	'destroy' method uses respondJson helper
Objective	Validate JSON response helper usage in destroy method.
Preconditions	- Application files exist

Test Steps	1. Inspect 'destroy' for 'respondJson' function call.
Test Data	- Function: respondJson
Expected Result	- JSON response helper used.
Actual Result	respondJson used as expected.
Status	Pass
Severity	Medium

Test Case ID	ECC-057
Title	ExpenseCategoriesController parent is Controller
Objective	Confirm controller class inheritance structure.
Preconditions	- Application running
Test Steps	1. Reflect ExpenseCategoriesController for parent. 2. Check parent class name.
Test Data	- Parent class: Crater\Http\Controllers\Controller
Expected Result	- Parent class is Controller.
Actual Result	Parent class confirmed.
Status	Pass
Severity	High

Test Case ID	ECC-058
Title	ExpenseCategoriesController has exactly 5 public methods
Objective	Ensure correct count of public methods on controller.
Preconditions	- Application files exist
Test Steps	1. Reflect ExpenseCategoriesController for public methods. 2. Count excluding inherited. 3. Compare to expected value (5).
Test Data	- Public methods: 5
Expected Result	- Exactly 5 public methods.
Actual Result	Public method count matches.
Status	Pass
Severity	Medium

Test Case ID	ECC-059
Title	'index' method handles limit parameter
Objective	Confirm limit parameter is handled in index queries.
Preconditions	- Application files exist
Test Steps	1. Inspect 'index' for $request->has('limit') and $request->limit usage.
Test Data	- limit parameter
Expected Result	- Index method checks and applies limit.
Actual Result	Limit parameter handled in code.
Status	Pass
Severity	Medium

Test Case ID	ECC-060
Title	'index' method has default limit of 5

Objective	Validate default limit enforcement in index method.
Preconditions	- Application files exist
Test Steps	1. Review 'index' for default limit initialization as 5.
Test Data	- Default limit: 5
Expected Result	- Default limit is set to 5.
Actual Result	Default limit detected.
Status	Pass
Severity	Medium

File: ExpenseCategory-Test.txt

Test Case ID	EC-001
Title	ExpenseCategory can be instantiated
Objective	Verify that an instance of ExpenseCategory can be successfully created.
Preconditions	- Application running - Database seeded (if instantiation depends on DB connection) - ExpenseCategory class is autoloaded
Test Steps	1. Import the ExpenseCategory class. 2. Create a new instance using `new ExpenseCategory()`. 3. Check the type of the created instance.
Test Data	- Class: ExpenseCategory - Input: None
Expected Result	- The created object is an instance of ExpenseCategory.
Actual Result	Object is successfully instantiated as ExpenseCategory.
Status	Pass
Severity	Medium

Test Case ID	EC-002
Title	ExpenseCategory extends Model
Objective	Verify that ExpenseCategory inherits from Eloquent Model.
Preconditions	- Application running - Database seeded (if necessary) - ExpenseCategory and Model classes are autoloaded
Test Steps	1. Import ExpenseCategory class. 2. Create a new ExpenseCategory instance. 3. Check that the instance is also an instance of Model.
Test Data	- Class: ExpenseCategory - Input: None
Expected Result	- ExpenseCategory instance is an instance of Illuminate\Database\Eloquent\Model.
Actual Result	Instance verified as subclass of Model.
Status	Pass
Severity	Medium

Test Case ID	EC-003
Title	ExpenseCategory is in correct namespace
Objective	Verify that the ExpenseCategory class is declared in the Crater\Models namespace.
Preconditions	- Application running - ExpenseCategory class available
Test Steps	1. Use ReflectionClass on ExpenseCategory. 2. Retrieve namespace name. 3. Verify that namespace is 'Crater\Models'.
Test Data	- Class: ExpenseCategory - Expected namespace: Crater\Models
Expected Result	- Namespace is correctly set to 'Crater\Models'.
Actual Result	Namespace is 'Crater\Models'.
Status	Pass
Severity	Medium

Test Case ID	EC-004
Title	ExpenseCategory is not abstract
Objective	Confirm that ExpenseCategory is a concrete class and not declared abstract.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Check isAbstract property.
Test Data	- Class: ExpenseCategory
Expected Result	- isAbstract returns false.
Actual Result	isAbstract returns false.
Status	Pass
Severity	Medium

Test Case ID	EC-005
Title	ExpenseCategory is instantiable
Objective	Ensure ExpenseCategory can be instantiated (not abstract, interface, or trait).
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Check isInstantiable property.
Test Data	- Class: ExpenseCategory
Expected Result	- isInstantiable returns true.
Actual Result	isInstantiable returns true.
Status	Pass
Severity	Medium

Test Case ID	EC-006
Title	ExpenseCategory has fillable properties
Objective	Verify that model fillable properties include name, company_id, and description.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Retrieve fillable properties via getFillable(). 3. Check for 'name', 'company_id', and 'description' in the array.
Test Data	- Class: ExpenseCategory
Expected Result	- Fillable properties contain 'name', 'company_id', 'description'.
Actual Result	All three properties found in fillable list.
Status	Pass
Severity	Medium

Test Case ID	EC-007
Title	ExpenseCategory allows mass assignment of name
Objective	Verify mass assignment for name property via fill().
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Call fill(['name' => 'Test Category']). 3. Check that name property is correctly set.
Test Data	- Input: ['name' => 'Test Category']

Expected Result	- category->name is 'Test Category'.
Actual Result	category->name is 'Test Category'.
Status	Pass
Severity	Medium

Test Case ID	EC-008
Title	ExpenseCategory allows mass assignment of company_id
Objective	Verify mass assignment for company_id property via fill().
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Call fill(['company_id' => 1]). 3. Check that company_id property is correctly set.
Test Data	- Input: ['company_id' => 1]
Expected Result	- category->company_id is 1.
Actual Result	category->company_id is 1.
Status	Pass
Severity	Medium

Test Case ID	EC-009
Title	ExpenseCategory allows mass assignment of description
Objective	Verify mass assignment for description property via fill().
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Call fill(['description' => 'Test Description']). 3. Check that description property is correctly set.
Test Data	- Input: ['description' => 'Test Description']
Expected Result	- category->description is 'Test Description'.
Actual Result	category->description is 'Test Description'.
Status	Pass
Severity	Medium

Test Case ID	EC-010
Title	ExpenseCategory has appends property
Objective	Confirm that ExpenseCategory contains appends property with expected values.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Use ReflectionClass to access protected $appends property. 3. Verify contents include 'amount' and 'formattedCreatedAt'.
Test Data	- Expected appends: ['amount', 'formattedCreatedAt']
Expected Result	- 'amount' and 'formattedCreatedAt' found in $appends property.
Actual Result	Both values are present in appends.
Status	Pass
Severity	Medium

Test Case ID	EC-011
Title	expenses relationship exists

Objective	Verify that ExpenseCategory has an expenses() method defined.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for existence of expenses method.
Test Data	- class: ExpenseCategory
Expected Result	- expenses method exists on ExpenseCategory.
Actual Result	expenses method exists.
Status	Pass
Severity	Medium

Test Case ID	EC-012
Title	expenses relationship returns HasMany
Objective	Verify that the expenses relationship returns a HasMany relation.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Call expenses(). 3. Confirm return type is HasMany.
Test Data	- class: ExpenseCategory
Expected Result	- expenses() returns a HasMany instance.
Actual Result	Return type is HasMany.
Status	Pass
Severity	Medium

Test Case ID	EC-013
Title	expenses relationship is to Expense model
Objective	Ensure that the expenses relationship links to Expense model.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Call expenses(). 3. Retrieve related model from relationship. 4. Check that related model is Expense.
Test Data	- class: ExpenseCategory
Expected Result	- Related model is Expense.
Actual Result	Related model is Expense.
Status	Pass
Severity	Medium

Test Case ID	EC-014
Title	expenses relationship uses expense_category_id foreign key
Objective	Verify foreign key used in expenses relationship is expense_category_id.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Get foreign key name from expenses relationship. 3. Confirm it is 'expense_category_id'.
Test Data	- class: ExpenseCategory
Expected Result	- Foreign key name is 'expense_category_id'.
Actual Result	Foreign key is 'expense_category_id'.

Status	Pass
Severity	Medium

Test Case ID	EC-015
Title	expenses relationship uses id local key
Objective	Ensure the local key for expenses relationship is 'id'.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Get local key name from expenses relationship. 3. Confirm it is 'id'.
Test Data	- class: ExpenseCategory
Expected Result	- Local key name is 'id'.
Actual Result	Local key name is 'id'.
Status	Pass
Severity	Medium

Test Case ID	EC-016
Title	company relationship exists
Objective	Verify that ExpenseCategory has a company() method defined.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for existence of company method.
Test Data	- class: ExpenseCategory
Expected Result	- company method exists on ExpenseCategory.
Actual Result	company method exists.
Status	Pass
Severity	Medium

Test Case ID	EC-017
Title	company relationship returns BelongsTo
Objective	Ensure company relationship returns a BelongsTo relation.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Call company(). 3. Confirm return type is BelongsTo.
Test Data	- class: ExpenseCategory
Expected Result	- company() returns a BelongsTo instance.
Actual Result	Return type is BelongsTo.
Status	Pass
Severity	Medium

Test Case ID	EC-018
Title	company relationship is to Company model
Objective	Verify that company relationship links to Company model.
Preconditions	- Application running

Test Steps	1. Create ExpenseCategory instance. 2. Call company(). 3. Get related model. 4. Verify related model is instance of Company.
Test Data	- class: ExpenseCategory
Expected Result	- Related model is Company.
Actual Result	Related model is Company.
Status	Pass
Severity	Medium

Test Case ID	EC-019
Title	company relationship uses company_id foreign key
Objective	Confirm that foreign key used by company relationship is company_id.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Get foreign key name from company relationship. 3. Verify it's 'company_id'.
Test Data	- class: ExpenseCategory
Expected Result	- Foreign key name is 'company_id'.
Actual Result	Foreign key is 'company_id'.
Status	Pass
Severity	Medium

Test Case ID	EC-020
Title	company relationship uses id owner key
Objective	Verify the owner key for company relationship is 'id'.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Get owner key name from company relationship. 3. Confirm it's 'id'.
Test Data	- class: ExpenseCategory
Expected Result	- Owner key name is 'id'.
Actual Result	Owner key is 'id'.
Status	Pass
Severity	Medium

Test Case ID	EC-021
Title	ExpenseCategory has getFormattedCreatedAtAttribute method
Objective	Verify existence of getFormattedCreatedAtAttribute accessor.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for getFormattedCreatedAtAttribute method.
Test Data	- class: ExpenseCategory
Expected Result	- Method exists on ExpenseCategory.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	EC-022
Title	ExpenseCategory has getAmountAttribute method
Objective	Verify existence of getAmountAttribute accessor.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for getAmountAttribute method.
Test Data	- class: ExpenseCategory
Expected Result	- Method exists on ExpenseCategory.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	EC-023
Title	ExpenseCategory has scopeWhereCompany method
Objective	Verify existence of scopeWhereCompany query scope.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for scopeWhereCompany method.
Test Data	- class: ExpenseCategory
Expected Result	- Method exists on ExpenseCategory.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	EC-024
Title	ExpenseCategory has scopeWhereCategory method
Objective	Verify existence of scopeWhereCategory query scope.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for scopeWhereCategory method.
Test Data	- class: ExpenseCategory
Expected Result	- Method exists on ExpenseCategory.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	EC-025
Title	ExpenseCategory has scopeWhereSearch method
Objective	Verify existence of scopeWhereSearch query scope.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for scopeWhereSearch method.
Test Data	- class: ExpenseCategory
Expected Result	- Method exists on ExpenseCategory.
Actual Result	Method exists.

Status	Pass
Severity	Medium

Test Case ID	EC-026
Title	ExpenseCategory has scopeApplyFilters method
Objective	Verify existence of scopeApplyFilters query scope.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for scopeApplyFilters method.
Test Data	- class: ExpenseCategory
Expected Result	- Method exists on ExpenseCategory.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	EC-027
Title	ExpenseCategory has scopePaginateData method
Objective	Verify existence of scopePaginateData query scope.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for scopePaginateData method.
Test Data	- class: ExpenseCategory
Expected Result	- Method exists on ExpenseCategory.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	EC-028
Title	All scope methods are public
Objective	Confirm that all query scope methods are declared public.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Check isPublic for each scope method.
Test Data	- Methods: scopeWhereCompany, scopeWhereCategory, scopeWhereSearch, scopeApplyFilters, scopePaginateData
Expected Result	- All are public methods.
Actual Result	All methods confirmed as public.
Status	Pass
Severity	Medium

Test Case ID	EC-029
Title	All accessor methods are public
Objective	Confirm that all accessor methods are public.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Check isPublic for getFormattedCreatedAtAttribute and getAmountAttribute.

Test Data	- Methods: getFormattedCreatedAtAttribute, getAmountAttribute
Expected Result	- Both methods are public.
Actual Result	Both methods are public.
Status	Pass
Severity	Medium

Test Case ID	EC-030
Title	scopeWhereCategory accepts category_id parameter
Objective	Verify that scopeWhereCategory has correct parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Get scopeWhereCategory method parameters. 3. Ensure two parameters exist: query and category_id.
Test Data	- Method: scopeWhereCategory
Expected Result	- Parameters: query, category_id
Actual Result	Parameters are query and category_id.
Status	Pass
Severity	Medium

Test Case ID	EC-031
Title	scopeWhereSearch accepts search parameter
Objective	Verify that scopeWhereSearch has correct parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Get scopeWhereSearch method parameters. 3. Ensure two parameters exist: query and search.
Test Data	- Method: scopeWhereSearch
Expected Result	- Parameters: query, search
Actual Result	Parameters are query and search.
Status	Pass
Severity	Medium

Test Case ID	EC-032
Title	scopeApplyFilters accepts filters array parameter
Objective	Verify that scopeApplyFilters method has expected parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Get scopeApplyFilters method parameters. 3. Ensure two parameters exist: query and filters.
Test Data	- Method: scopeApplyFilters
Expected Result	- Parameters: query, filters
Actual Result	Parameters are query and filters.
Status	Pass
Severity	Medium

Test Case ID	EC-033

Title	scopePaginateData accepts limit parameter
Objective	Verify that scopePaginateData method has correct parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Get scopePaginateData method parameters. 3. Ensure two parameters exist: query and limit.
Test Data	- Method: scopePaginateData
Expected Result	- Parameters: query, limit
Actual Result	Parameters are query and limit.
Status	Pass
Severity	Medium

Test Case ID	EC-034
Title	ExpenseCategory uses HasFactory trait
Objective	Confirm that the HasFactory trait is used in ExpenseCategory.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Get trait names. 3. Check for 'Illuminate\Database\Eloquent\Factories\HasFactory'.
Test Data	- class: ExpenseCategory
Expected Result	- Trait 'HasFactory' is used.
Actual Result	Trait is present.
Status	Pass
Severity	Medium

Test Case ID	EC-035
Title	multiple ExpenseCategory instances can be created
Objective	Verify creation and uniqueness of multiple instances.
Preconditions	- Application running
Test Steps	1. Create two ExpenseCategory instances. 2. Check both are ExpenseCategory. 3. Check they are not the same object.
Test Data	- No unique input
Expected Result	- Two instances are created and not identical.
Actual Result	Instances created and not identical.
Status	Pass
Severity	Medium

Test Case ID	EC-036
Title	ExpenseCategory can be cloned
Objective	Verify that cloning an ExpenseCategory creates a distinct object.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance with 'name' set to 'Test'. 2. Clone the instance. 3. Check clone is also ExpenseCategory. 4. Ensure clone is not same as original.
Test Data	- Input: ['name' => 'Test']

Expected Result	- Clone is instance of ExpenseCategory and not same object.
Actual Result	Clone is distinct and type correct.
Status	Pass
Severity	Medium

Test Case ID	EC-037
Title	ExpenseCategory can be used in type hints
Objective	Confirm compatibility of ExpenseCategory with type hints.
Preconditions	- Application running
Test Steps	1. Define function with ExpenseCategory parameter. 2. Pass ExpenseCategory instance to function. 3. Check function returns correct instance.
Test Data	- Instance created, passed via function
Expected Result	- Returned value matches input instance.
Actual Result	Returned instance matches.
Status	Pass
Severity	Medium

Test Case ID	EC-038
Title	ExpenseCategory is not final
Objective	Verify that ExpenseCategory class is not marked as final.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Check isFinal property.
Test Data	- class: ExpenseCategory
Expected Result	- isFinal returns false.
Actual Result	isFinal returns false.
Status	Pass
Severity	Medium

Test Case ID	EC-039
Title	ExpenseCategory is not an interface
Objective	Verify ExpenseCategory is a concrete class, not interface.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Check isInterface property.
Test Data	- class: ExpenseCategory
Expected Result	- isInterface returns false.
Actual Result	isInterface returns false.
Status	Pass
Severity	Medium

Test Case ID	EC-040
Title	ExpenseCategory is not a trait
Objective	Confirm ExpenseCategory is not a trait.
Preconditions	- Application running

Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Check isTrait property.
Test Data	- class: ExpenseCategory
Expected Result	- isTrait returns false.
Actual Result	isTrait returns false.
Status	Pass
Severity	Medium

Test Case ID	EC-041
Title	ExpenseCategory class is loaded
Objective	Verify ExpenseCategory class exists in memory.
Preconditions	- Application running
Test Steps	1. Check with class_exists if ExpenseCategory is loaded.
Test Data	- class: ExpenseCategory
Expected Result	- class_exists returns true.
Actual Result	Class exists in memory.
Status	Pass
Severity	Medium

Test Case ID	EC-042
Title	ExpenseCategory uses required classes
Objective	Confirm use/imports of required classes in ExpenseCategory file.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get file contents. 2. Check for use statements for Carbon, HasFactory, and Model.
Test Data	- File contents
Expected Result	- All required classes are imported.
Actual Result	Required classes found.
Status	Pass
Severity	Medium

Test Case ID	EC-043
Title	ExpenseCategory file has expected structure
Objective	Verify file has correct class structure and expected properties.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Check presence of class extends statement and protected properties.
Test Data	- File contents
Expected Result	- Class extends Model, has $fillable and $appends
Actual Result	Structure matches expectations.
Status	Pass
Severity	Medium

Test Case ID	EC-044
Title	ExpenseCategory has reasonable line count

Objective	Confirm model file size is within expected bounds.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Count lines. 3. Ensure count is >50 and <150.
Test Data	- File contents
Expected Result	- Line count >50, <150.
Actual Result	Line count is within expected range.
Status	Pass
Severity	Low

Test Case ID	EC-045
Title	scopeWhereCompany uses request header
Objective	Verify implementation that uses HTTP request header in scopeWhereCompany.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Search for usage of 'request()->header('company')' in scopeWhereCompany.
Test Data	- File contents
Expected Result	- 'request()->header('company')' is present.
Actual Result	Header usage found.
Status	Pass
Severity	Medium

Test Case ID	EC-046
Title	scopeWhereCategory uses orWhere
Objective	Confirm that scopeWhereCategory uses orWhere on query.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Verify use of '$query->orWhere('id', $category_id)' in method.
Test Data	- File contents
Expected Result	- '$query->orWhere' usage found.
Actual Result	orWhere usage present.
Status	Pass
Severity	Medium

Test Case ID	EC-047
Title	scopeWhereSearch uses LIKE operator
Objective	Confirm LIKE operator is used for searching in scopeWhereSearch.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Check usage of 'LIKE' and '%'.$search.'%'.
Test Data	- File contents
Expected Result	- LIKE and '%'.$search.'%' found.
Actual Result	LIKE operator found.
Status	Pass

Severity	Medium

Test Case ID	EC-048
Title	scopeApplyFilters uses collect helper
Objective	Verify that filter array is wrapped using collect().
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Search for 'collect($filters)' in scopeApplyFilters.
Test Data	- File contents
Expected Result	- 'collect($filters)' is used.
Actual Result	Collect usage present.
Status	Pass
Severity	Medium

Test Case ID	EC-049
Title	scopeApplyFilters checks category_id filter
Objective	Confirm category_id filter is checked in scopeApplyFilters.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Check for '$filters->get('category_id')' and related logic.
Test Data	- File contents
Expected Result	- category_id filter check is present.
Actual Result	category_id filter logic found.
Status	Pass
Severity	Medium

Test Case ID	EC-050
Title	scopeApplyFilters checks company_id filter
Objective	Confirm company_id filter is checked in scopeApplyFilters.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Search for '$filters->get('company_id')' and related logic.
Test Data	- File contents
Expected Result	- company_id filter logic present.
Actual Result	company_id filter check found.
Status	Pass
Severity	Medium

Test Case ID	EC-051
Title	scopeApplyFilters checks search filter
Objective	Confirm search filter is checked in scopeApplyFilters.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Search for '$filters->get('search')' and related logic.
Test Data	- File contents
Expected Result	- search filter logic present.

Actual Result	search filter logic found.
Status	Pass
Severity	Medium

Test Case ID	EC-052
Title	scopePaginateData handles 'all' limit
Objective	Verify that scopePaginateData handles 'all' limit correctly.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Confirm if ($limit == 'all') returns $query->get().
Test Data	- File contents
Expected Result	- Logic for 'all' limit found.
Actual Result	'all' case is handled as expected.
Status	Pass
Severity	Medium

Test Case ID	EC-053
Title	scopePaginateData uses paginate for numeric limit
Objective	Verify that scopePaginateData paginates for numeric limits.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Confirm return of $query->paginate($limit).
Test Data	- File contents
Expected Result	- Numeric limit is paginated.
Actual Result	Pagination logic present.
Status	Pass
Severity	Medium

Test Case ID	EC-054
Title	getAmountAttribute uses expenses sum
Objective	Confirm getAmountAttribute sums expenses amounts.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Check for '$this->expenses()->sum('amount')' in getAmountAttribute.
Test Data	- File contents
Expected Result	- Summing logic is present.
Actual Result	Sum logic found.
Status	Pass
Severity	Medium

Test Case ID	EC-055
Title	getFormattedCreatedAtAttribute uses CompanySetting
Objective	Confirm CompanySetting usage in getFormattedCreatedAtAttribute.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Check for 'CompanySetting::getSetting' in getFormattedCreatedAtAttribute.

Test Data	- File contents
Expected Result	- CompanySetting is used.
Actual Result	CompanySetting usage found.
Status	Pass
Severity	Medium

Test Case ID	EC-056
Title	getFormattedCreatedAtAttribute uses Carbon parse
Objective	Verify that Carbon::parse is used to format creation date.
Preconditions	- Application running
Test Steps	1. Read ExpenseCategory file. 2. Check for 'Carbon::parse($this->created_at)' in getFormattedCreatedAtAttribute.
Test Data	- File contents
Expected Result	- Carbon parser is used.
Actual Result	Carbon parser found.
Status	Pass
Severity	Medium

Test Case ID	EC-057
Title	Can set and get name attribute
Objective	Verify attribute assignment and retrieval for name property.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Set name to 'Test Category'. 3. Retrieve name and verify.
Test Data	- Input: 'Test Category' - Field: name
Expected Result	- name is set and retrieved as 'Test Category'.
Actual Result	name correctly set and retrieved.
Status	Pass
Severity	Medium

Test Case ID	EC-058
Title	Can set and get company_id attribute
Objective	Verify attribute assignment and retrieval for company_id property.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Set company_id to 5. 3. Retrieve company_id and verify.
Test Data	- Input: 5 - Field: company_id
Expected Result	- company_id is set and retrieved as 5.
Actual Result	company_id correctly set and retrieved.
Status	Pass
Severity	Medium

Test Case ID	EC-059
Title	Can set and get description attribute
Objective	Verify attribute assignment and retrieval for description property.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Set description to 'Test Description'. 3. Retrieve description and verify.
Test Data	- Input: 'Test Description' - Field: description
Expected Result	- description is set and retrieved as 'Test Description'.
Actual Result	description correctly set and retrieved.
Status	Pass
Severity	Medium

Test Case ID	EC-060
Title	Different instances have independent data
Objective	Ensure that multiple ExpenseCategory instances hold separate data.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory with name 'Category 1'. 2. Create ExpenseCategory with name 'Category 2'. 3. Verify that names are exclusive to their instances.
Test Data	- Input: 'Category 1', 'Category 2'
Expected Result	- category1->name != category2->name - category1->name == 'Category 1' - category2->name == 'Category 2'
Actual Result	Instance data is independent.
Status	Pass
Severity	Medium

Test Case ID	EC-061
Title	ExpenseCategory parent is Model
Objective	Verify the parent class of ExpenseCategory is Model.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass for ExpenseCategory. 2. Get parent class. 3. Confirm name is 'Illuminate\Database\Eloquent\Model'.
Test Data	- class: ExpenseCategory
Expected Result	- Parent class is Model.
Actual Result	Parent is Model.
Status	Pass
Severity	Medium

Test Case ID	EC-062
Title	ExpenseCategory inherits Model methods
Objective	Ensure that ExpenseCategory inherits core Model functionality.
Preconditions	- Application running
Test Steps	1. Create ExpenseCategory instance. 2. Check for existence of methods: save, fill, toArray.

Test Data	- class: ExpenseCategory
Expected Result	- Methods save, fill, toArray exist.
Actual Result	All methods present.
Status	Pass
Severity	Medium

File: ExpenseCategoryCollection-Test.txt

Test Case ID	ECC-001
Title	Instantiation of ExpenseCategoryCollection
Objective	Verify that an ExpenseCategoryCollection object can be instantiated correctly.
Preconditions	- Application running - Database seeded - "ExpenseCategoryCollection" class available
Test Steps	1. Create a new ExpenseCategoryCollection with an empty Collection. 2. Assert that the created object is an instance of ExpenseCategoryCollection.
Test Data	- Input: new Collection([]) - Expected Class: ExpenseCategoryCollection
Expected Result	- The created object is an instance of ExpenseCategoryCollection.
Actual Result	The created object is an instance of ExpenseCategoryCollection.
Status	Pass
Severity	Medium

Test Case ID	ECC-002
Title	ExpenseCategoryCollection extends ResourceCollection
Objective	Ensure ExpenseCategoryCollection inherits from ResourceCollection.
Preconditions	- Application running - Database seeded - "ResourceCollection" and "ExpenseCategoryCollection" classes available
Test Steps	1. Create a new ExpenseCategoryCollection. 2. Assert that the object is an instance of ResourceCollection.
Test Data	- Input: new Collection([]) - Expected Base Class: Illuminate\Http\Resources\Json\ResourceCollection
Expected Result	- The created object is an instance of ResourceCollection.
Actual Result	The created object is an instance of ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	ECC-003
Title	Check ExpenseCategoryCollection Namespace
Objective	Verify that ExpenseCategoryCollection is under the correct namespace.
Preconditions	- Application running - Database seeded - "ExpenseCategoryCollection" class available
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Read the namespace value. 3. Assert that the namespace is "Crater\Http\Resources".
Test Data	- Input: ExpenseCategoryCollection class reference
Expected Result	- The class resides in the "Crater\Http\Resources" namespace.
Actual Result	The class resides in the "Crater\Http\Resources" namespace.
Status	Pass
Severity	Low

Test Case ID	ECC-004

Title	Presence of toArray Method in ExpenseCategoryCollection
Objective	Confirm ExpenseCategoryCollection has a toArray method.
Preconditions	- Application running - Database seeded - "ExpenseCategoryCollection" class available
Test Steps	1. Create an ExpenseCategoryCollection instance. 2. Check for the method "toArray" using method_exists.
Test Data	- Input: ExpenseCategoryCollection instance
Expected Result	- The method "toArray" exists in the ExpenseCategoryCollection class.
Actual Result	The method "toArray" exists in the ExpenseCategoryCollection class.
Status	Pass
Severity	Medium

Test Case ID	ECC-005
Title	toArray Method Accessibility
Objective	Ensure that the toArray method is declared public.
Preconditions	- Application running - Database seeded - "ExpenseCategoryCollection" class available
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Inspect the visibility of "toArray" method. 3. Assert method is public.
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- The "toArray" method is public.
Actual Result	The "toArray" method is public.
Status	Pass
Severity	Medium

Test Case ID	ECC-006
Title	toArray Method Accepts Request Parameter
Objective	Confirm that toArray method receives the "request" parameter correctly.
Preconditions	- Application running - Database seeded - "ExpenseCategoryCollection" class available
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Retrieve parameters for "toArray" method. 3. Assert the method has one parameter named "request".
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- "toArray" method accepts a parameter named "request".
Actual Result	"toArray" method accepts a parameter named "request".
Status	Pass
Severity	Medium

Test Case ID	ECC-007
Title	Handling of Empty Collection
Objective	Verify ExpenseCategoryCollection correctly transforms an empty collection.
Preconditions	- Application running - Database seeded

Test Steps	1. Create an empty ExpenseCategoryCollection. 2. Call toArray with a new Request. 3. Assert the result is an empty array.
Test Data	- Input: Collection([]) - Expected Output: []
Expected Result	- Returned array is empty.
Actual Result	Returned array is empty.
Status	Pass
Severity	High

Test Case ID	ECC-008
Title	Transformation of Single Expense Category Resource
Objective	Ensure a single ExpenseCategoryResource is transformed correctly in collection.
Preconditions	- Application running - Database seeded
Test Steps	1. Create a dummy category: id = 1, name = "Travel". 2. Wrap it in ExpenseCategoryResource. 3. Create ExpenseCategoryCollection with one resource. 4. Call toArray with a new Request. 5. Validate returned array has one item with id = 1 and required keys.
Test Data	- Input: id = 1, name = "Travel" - Expected Output: array with item having 'id', 'name', 'description', id = 1
Expected Result	- Returned array has one element. - The single element has key 'id'. - The single element's 'id' is 1.
Actual Result	Returned array has one element: ['id' => 1, ...], with correctly structured keys.
Status	Pass
Severity	High

Test Case ID	ECC-009
Title	Transformation of Multiple Expense Category Resources
Objective	Verify multiple ExpenseCategoryResources are transformed correctly.
Preconditions	- Application running - Database seeded
Test Steps	1. Create two dummy categories: (1, "Travel") and (2, "Supplies"). 2. Wrap each in ExpenseCategoryResource. 3. Create ExpenseCategoryCollection with both resources. 4. Call toArray with a new Request. 5. Check result contains two items with ids 1 and 2.
Test Data	- Input: id = 1, name = "Travel"; id = 2, name = "Supplies" - Expected Output: array with two items [{id: 1}, {id: 2}]
Expected Result	- Returned array has two elements. - Element 1: 'id' = 1; Element 2: 'id' = 2.
Actual Result	Returned array has two elements: [{'id' => 1}, {'id' => 2}].
Status	Pass
Severity	High

Test Case ID	ECC-010
Title	Transformed Item Contains Required Fields
Objective	Check each transformed resource contains all required fields.

Preconditions	- Application running - Database seeded
Test Steps	1. Create a dummy category: id = 1, name = "Travel". 2. Wrap as ExpenseCategoryResource. 3. Create ExpenseCategoryCollection. 4. Call toArray with a new Request. 5. Confirm returned resource has keys: 'id', 'name', 'description'.
Test Data	- Input: id = 1, name = "Travel" - Expected Keys: 'id', 'name', 'description'
Expected Result	- Returned resource array contains all required keys.
Actual Result	Returned resource array contains 'id', 'name', 'description'.
Status	Pass
Severity	High

Test Case ID	ECC-011
Title	Handling Large Collections
Objective	Validate transformation of a large collection of expense categories.
Preconditions	- Application running - Database seeded
Test Steps	1. Create 50 dummy categories (id 1 to 50). 2. Wrap each as ExpenseCategoryResource. 3. Create ExpenseCategoryCollection of 50 resources. 4. Call toArray with a new Request. 5. Verify array contains 50 items. 6. Confirm first item id = 1; last item id = 50.
Test Data	- Input: id = 1...50, name = "Category N" - Expected Output: array of length 50, {id: 1} ... {id: 50}
Expected Result	- Returned array has 50 elements. - First element 'id' is 1. - Last element 'id' is 50.
Actual Result	Returned array includes 50 items with ids from 1 to 50.
Status	Pass
Severity	High

Test Case ID	ECC-012
Title	toArray Delegates to Parent ResourceCollection
Objective	Ensure toArray method calls parent::toArray internally.
Preconditions	- Application running - Database seeded
Test Steps	1. Open ExpenseCategoryCollection source file. 2. Search for "parent::toArray". 3. Assert presence of this call within toArray method.
Test Data	- Input: Source file contents
Expected Result	- Source file contains "parent::toArray" in toArray method.
Actual Result	Source file contains "parent::toArray".
Status	Pass
Severity	Medium

Test Case ID	ECC-013
Title	ExpenseCategoryCollection Parent Class is ResourceCollection

Objective	Validate correct class inheritance.
Preconditions	- Application running - Database seeded
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Get parent class name. 3. Assert parent is ResourceCollection.
Test Data	- Input: ExpenseCategoryCollection class reference
Expected Result	- Parent class is Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	Parent class is Illuminate\Http\Resources\Json\ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	ECC-014
Title	Creating Multiple Collection Instances
Objective	Ensure multiple ExpenseCategoryCollection instances can be created and are unique.
Preconditions	- Application running - Database seeded
Test Steps	1. Create two ExpenseCategoryCollection objects separately. 2. Assert both are valid instances. 3. Assert both instances are unique (not equal).
Test Data	- Input: Two instances, each with Collection([])
Expected Result	- Both instances are objects of ExpenseCategoryCollection. - Objects are not equal.
Actual Result	Both created instances are valid and unique.
Status	Pass
Severity	Medium

Test Case ID	ECC-015
Title	ExpenseCategoryCollection Can be Cloned
Objective	Confirm that an ExpenseCategoryCollection object can be cloned.
Preconditions	- Application running - Database seeded
Test Steps	1. Create an ExpenseCategoryCollection instance. 2. Clone the object. 3. Assert clone is a valid instance. 4. Assert original and clone are different objects.
Test Data	- Input: Instance of ExpenseCategoryCollection
Expected Result	- Clone is an ExpenseCategoryCollection instance. - Clone and original are non-identical objects.
Actual Result	Cloned object valid and different from original.
Status	Pass
Severity	Medium

Test Case ID	ECC-016
Title	Type Hint Usability of ExpenseCategoryCollection
Objective	Validate ExpenseCategoryCollection can be used in type hints.
Preconditions	- Application running - Database seeded

Test Steps	1. Define a test function with ExpenseCategoryCollection type-hint. 2. Pass an ExpenseCategoryCollection instance to the function. 3. Assert returned value matches the input object.
Test Data	- Input: Function param: ExpenseCategoryCollection instance
Expected Result	- Passed object is returned and matches input.
Actual Result	Passed object returned correctly.
Status	Pass
Severity	Low

Test Case ID	ECC-017
Title	ExpenseCategoryCollection is Not Abstract
Objective	Check that ExpenseCategoryCollection class is not abstract.
Preconditions	- Application running - Database seeded
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Assert isAbstract() returns false.
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- isAbstract() returns false.
Actual Result	Class is not abstract.
Status	Pass
Severity	Low

Test Case ID	ECC-018
Title	ExpenseCategoryCollection is Not Final
Objective	Verify the class can be extended.
Preconditions	- Application running - Database seeded
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Assert isFinal() returns false.
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- isFinal() returns false.
Actual Result	Class is not final.
Status	Pass
Severity	Low

Test Case ID	ECC-019
Title	ExpenseCategoryCollection is Not an Interface
Objective	Confirm ExpenseCategoryCollection is a concrete class, not an interface.
Preconditions	- Application running - Database seeded
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Assert isInterface() returns false.
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- isInterface() returns false.
Actual Result	Class is not an interface.
Status	Pass
Severity	Low

Test Case ID	ECC-020
Title	ExpenseCategoryCollection is Not a Trait
Objective	Ensure ExpenseCategoryCollection is not a trait.
Preconditions	- Application running - Database seeded
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Assert isTrait() returns false.
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- isTrait() returns false.
Actual Result	Class is not a trait.
Status	Pass
Severity	Low

Test Case ID	ECC-021
Title	ExpenseCategoryCollection Class is Loaded
Objective	Confirm the ExpenseCategoryCollection class is loaded at runtime.
Preconditions	- Application running - Database seeded
Test Steps	1. Assert that class_exists(ExpenseCategoryCollection::class) returns true.
Test Data	- Input: ExpenseCategoryCollection class name
Expected Result	- class_exists returns true.
Actual Result	class_exists returns true.
Status	Pass
Severity	Medium

Test Case ID	ECC-022
Title	ExpenseCategoryCollection Uses ResourceCollection Import
Objective	Ensure correct importing of ResourceCollection class in ExpenseCategoryCollection file.
Preconditions	- Application running - Database seeded
Test Steps	1. Open ExpenseCategoryCollection source file. 2. Search for "use Illuminate\Http\Resources\Json\ResourceCollection". 3. Assert import statement is present.
Test Data	- Input: Source file contents
Expected Result	- Import statement is included.
Actual Result	Import statement found in source file.
Status	Pass
Severity	Low

Test Case ID	ECC-023
Title	toArray Method is Not Static
Objective	Confirm toArray is an instance method, not static.
Preconditions	- Application running - Database seeded

Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Get "toArray" method's static status. 3. Assert method is not static.
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- isStatic() returns false.
Actual Result	Method is not static.
Status	Pass
Severity	Low

Test Case ID	ECC-024
Title	toArray Method is Not Abstract
Objective	Check that toArray method is implemented and not abstract.
Preconditions	- Application running - Database seeded
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Get "toArray" method's abstract status. 3. Assert method is not abstract.
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- isAbstract() returns false.
Actual Result	Method is not abstract.
Status	Pass
Severity	Low

Test Case ID	ECC-025
Title	Expense Category Data Integrity
Objective	Ensure that ExpenseCategoryCollection preserves the details of categories.
Preconditions	- Application running - Database seeded
Test Steps	1. Create dummy category: id = 42, name = "Marketing". 2. Wrap in ExpenseCategoryResource. 3. Transform collection via toArray. 4. Validate output preserves 'id' and 'name'.
Test Data	- Input: id = 42, name = "Marketing" - Expected Output: array item with 'id' = 42, 'name' = "Marketing"
Expected Result	- Output maintains correct 'id' and 'name' for category.
Actual Result	Output preserves 'id' = 42 and 'name' = "Marketing".
Status	Pass
Severity	High

Test Case ID	ECC-026
Title	Handling of Different Category Names
Objective	Verify toArray handles categories with different names and preserves them.
Preconditions	- Application running - Database seeded
Test Steps	1. Create categories with names: "Travel", "Supplies", "Utilities". 2. Wrap each in ExpenseCategoryResource. 3. Create collection and call toArray. 4. Confirm output names match input.

Test Data	- Input: id = 1, name = "Travel"; id = 2, name = "Supplies"; id = 3, name = "Utilities" - Expected Output: names preserved
Expected Result	- Output[0]['name'] = "Travel" - Output[1]['name'] = "Supplies" - Output[2]['name'] = "Utilities"
Actual Result	Names in output match expected.
Status	Pass
Severity	Medium

Test Case ID	ECC-027
Title	ExpenseCategoryCollection File is Concise
Objective	Check that the source file is simple and concise (less than 1000 bytes).
Preconditions	- Application running
Test Steps	1. Open ExpenseCategoryCollection source file. 2. Measure file size in bytes. 3. Assert size is less than 1000 bytes.
Test Data	- Source file
Expected Result	- File size is < 1000 bytes.
Actual Result	File is concise and under 1000 bytes.
Status	Pass
Severity	Low

Test Case ID	ECC-028
Title	ExpenseCategoryCollection Has Minimal Line Count
Objective	Ensure the class source file contains less than 30 lines for simplicity.
Preconditions	- Application running
Test Steps	1. Open ExpenseCategoryCollection source file. 2. Count total lines. 3. Assert the line count is < 30.
Test Data	- Source file
Expected Result	- Line count is less than 30.
Actual Result	File has 24 lines.
Status	Pass
Severity	Low

Test Case ID	ECC-029
Title	Transformed Items Include All Category Fields
Objective	Confirm each item in the transformed collection includes all relevant fields.
Preconditions	- Application running - Database seeded
Test Steps	1. Create dummy category: id = 1, name = "Travel". 2. Wrap in ExpenseCategoryResource. 3. Create ExpenseCategoryCollection and transform via toArray. 4. Assert output contains 'id', 'name', 'description' keys.
Test Data	- Input: id = 1, name = "Travel" - Expected Output Fields: 'id', 'name', 'description'
Expected Result	- Output contains all specified keys.
Actual Result	All keys present in output.

Status	Pass
Severity	High

Test Case ID	ECC-030
Title	Collection Result is a Valid Array Structure
Objective	Ensure ExpenseCategoryCollection's toArray produces a valid array.
Preconditions	- Application running - Database seeded
Test Steps	1. Create dummy category: id = 1, name = "Travel". 2. Wrap in ExpenseCategoryResource. 3. Transform using toArray. 4. Assert output is array and is_array returns true.
Test Data	- Input: id = 1, name = "Travel"
Expected Result	- Output is an array. - is_array($result) returns true.
Actual Result	Output is a valid array.
Status	Pass
Severity	High

Test Case ID	ECC-031
Title	toArray Method Has Documentation
Objective	Verify that toArray method has a docblock comment.
Preconditions	- Application running
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Retrieve doc comment of toArray method. 3. Assert method has documentation.
Test Data	- Input: ExpenseCategoryCollection class reflection
Expected Result	- Doc comment exists for the toArray method.
Actual Result	Documentation present in toArray method.
Status	Pass
Severity	Low

Test Case ID	ECC-032
Title	toArray Method Has Return Type Documentation
Objective	Ensure toArray's PHPDoc includes @return annotation.
Preconditions	- Application running
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Get doc comment for toArray method. 3. Search for @return keyword.
Test Data	- Input: Method documentation
Expected Result	- Doc comment contains "@return".
Actual Result	Docblock contains "@return".
Status	Pass
Severity	Low

Test Case ID	ECC-033
Title	toArray Method Has Param Documentation

Objective	Ensure toArray's docblock includes @param annotation.
Preconditions	- Application running
Test Steps	1. Reflect on ExpenseCategoryCollection::class. 2. Get doc comment for toArray method. 3. Search for @param keyword.
Test Data	- Input: Method documentation
Expected Result	- Doc comment contains "@param".
Actual Result	Docblock contains "@param".
Status	Pass
Severity	Low

Test Case ID	ECC-034
Title	Handles Categories with Different Descriptions
Objective	Ensure category descriptions are preserved and handled correctly.
Preconditions	- Application running - Database seeded
Test Steps	1. Create two categories: "Travel" desc "Business travel expenses", "Supplies" desc "Office supplies and materials". 2. Wrap in ExpenseCategoryResource and collection. 3. Transform collection with toArray. 4. Assert descriptions match expected values.
Test Data	- category1: id = 1, name = "Travel", description = "Business travel expenses" - category2: id = 2, name = "Supplies", description = "Office supplies and materials"
Expected Result	- output[0]['description'] = "Business travel expenses" - output[1]['description'] = "Office supplies and materials"
Actual Result	Output matches expected descriptions.
Status	Pass
Severity	Medium

File: ExpenseCategoryPolicy-Test.txt

Test Case ID	ECP-001
Title	Instantiation of ExpenseCategoryPolicy
Objective	Verify that ExpenseCategoryPolicy can be instantiated correctly.
Preconditions	- Application running - Classes autoloaded - Database seeded (optional; not required for class instantiation)
Test Steps	1. Attempt to create a new instance of ExpenseCategoryPolicy. 2. Verify the created object is an instance of ExpenseCategoryPolicy.
Test Data	- Instantiation input: none
Expected Result	- A new object of type ExpenseCategoryPolicy is created successfully.
Actual Result	A new object of type ExpenseCategoryPolicy was created successfully.
Status	Pass
Severity	Medium

Test Case ID	ECP-002
Title	Namespace Validation of ExpenseCategoryPolicy
Objective	Ensure ExpenseCategoryPolicy is defined in the Crater\Policies namespace.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Use ReflectionClass to inspect ExpenseCategoryPolicy. 2. Retrieve the class namespace using getNamespaceName(). 3. Compare the retrieved namespace with 'Crater\Policies'.
Test Data	- Class to inspect: ExpenseCategoryPolicy
Expected Result	- The namespace returned is 'Crater\Policies'.
Actual Result	The namespace returned was 'Crater\Policies'.
Status	Pass
Severity	Medium

Test Case ID	ECP-003
Title	Abstract Check for ExpenseCategoryPolicy
Objective	Confirm that ExpenseCategoryPolicy is not an abstract class.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Use ReflectionClass to inspect ExpenseCategoryPolicy. 2. Invoke isAbstract() on the class reflection. 3. Verify the result is false.
Test Data	- Class to inspect: ExpenseCategoryPolicy
Expected Result	- ExpenseCategoryPolicy is not abstract.
Actual Result	ExpenseCategoryPolicy was confirmed to be non-abstract.
Status	Pass
Severity	Medium

Test Case ID	ECP-004
Title	Instantiability of ExpenseCategoryPolicy
Objective	Confirm that ExpenseCategoryPolicy can be instantiated (isInstantiable returns true).

Preconditions	- Application running - Classes autoloaded
Test Steps	1. Use ReflectionClass to inspect ExpenseCategoryPolicy. 2. Invoke isInstantiable() and verify the result is true.
Test Data	- Class to inspect: ExpenseCategoryPolicy
Expected Result	- ExpenseCategoryPolicy is instantiable.
Actual Result	ExpenseCategoryPolicy was confirmed as instantiable.
Status	Pass
Severity	Medium

Test Case ID	ECP-005
Title	HandlesAuthorization Trait Usage
Objective	Verify ExpenseCategoryPolicy uses the HandlesAuthorization trait.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Use ReflectionClass on ExpenseCategoryPolicy. 2. Retrieve trait names via getTraitNames(). 3. Check if 'Illuminate\Auth\Access\HandlesAuthorization' is present.
Test Data	- Trait expected: Illuminate\Auth\Access\HandlesAuthorization
Expected Result	- The trait list contains 'Illuminate\Auth\Access\HandlesAuthorization'.
Actual Result	The trait list included 'Illuminate\Auth\Access\HandlesAuthorization'.
Status	Pass
Severity	Medium

Test Case ID	ECP-006
Title	Existence of viewAny Method
Objective	Verify that ExpenseCategoryPolicy defines a viewAny method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Create ExpenseCategoryPolicy instance. 2. Check for the existence of the 'viewAny' method using method_exists().
Test Data	- Method name: 'viewAny'
Expected Result	- Method 'viewAny' exists on ExpenseCategoryPolicy.
Actual Result	Method 'viewAny' exists on ExpenseCategoryPolicy.
Status	Pass
Severity	High

Test Case ID	ECP-007
Title	Existence of view Method
Objective	Ensure ExpenseCategoryPolicy defines a view method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Create ExpenseCategoryPolicy instance. 2. Check for the existence of the 'view' method using method_exists().
Test Data	- Method name: 'view'
Expected Result	- Method 'view' exists on ExpenseCategoryPolicy.
Actual Result	Method 'view' exists on ExpenseCategoryPolicy.

Status	Pass
Severity	High

Test Case ID	ECP-008
Title	Existence of create Method
Objective	Ensure ExpenseCategoryPolicy defines a create method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Create ExpenseCategoryPolicy instance. 2. Check for the existence of the 'create' method using method_exists().
Test Data	- Method name: 'create'
Expected Result	- Method 'create' exists on ExpenseCategoryPolicy.
Actual Result	Method 'create' exists on ExpenseCategoryPolicy.
Status	Pass
Severity	High

Test Case ID	ECP-009
Title	Existence of update Method
Objective	Ensure ExpenseCategoryPolicy defines an update method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Create ExpenseCategoryPolicy instance. 2. Check for the existence of the 'update' method using method_exists().
Test Data	- Method name: 'update'
Expected Result	- Method 'update' exists on ExpenseCategoryPolicy.
Actual Result	Method 'update' exists on ExpenseCategoryPolicy.
Status	Pass
Severity	High

Test Case ID	ECP-010
Title	Existence of delete Method
Objective	Ensure ExpenseCategoryPolicy defines a delete method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Create ExpenseCategoryPolicy instance. 2. Check for the existence of the 'delete' method using method_exists().
Test Data	- Method name: 'delete'
Expected Result	- Method 'delete' exists on ExpenseCategoryPolicy.
Actual Result	Method 'delete' exists on ExpenseCategoryPolicy.
Status	Pass
Severity	High

Test Case ID	ECP-011
Title	Existence of restore Method
Objective	Ensure ExpenseCategoryPolicy defines a restore method.
Preconditions	- Application running - Classes autoloaded

Test Steps	1. Create ExpenseCategoryPolicy instance. 2. Check for the existence of the 'restore' method using method_exists().
Test Data	- Method name: 'restore'
Expected Result	- Method 'restore' exists on ExpenseCategoryPolicy.
Actual Result	Method 'restore' exists on ExpenseCategoryPolicy.
Status	Pass
Severity	High

Test Case ID	ECP-012
Title	Existence of forceDelete Method
Objective	Ensure ExpenseCategoryPolicy defines a forceDelete method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Create ExpenseCategoryPolicy instance. 2. Check for the existence of the 'forceDelete' method using method_exists().
Test Data	- Method name: 'forceDelete'
Expected Result	- Method 'forceDelete' exists on ExpenseCategoryPolicy.
Actual Result	Method 'forceDelete' exists on ExpenseCategoryPolicy.
Status	Pass
Severity	High

Test Case ID	ECP-013
Title	All Policy Methods Public
Objective	Ensure all relevant ExpenseCategoryPolicy methods are public.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect ExpenseCategoryPolicy using ReflectionClass. 2. For each of the 7 methods, verify isPublic() is true.
Test Data	- Method names: viewAny, view, create, update, delete, restore, forceDelete
Expected Result	- All listed methods are public.
Actual Result	All listed methods were confirmed as public.
Status	Pass
Severity	High

Test Case ID	ECP-014
Title	All Policy Methods Non-Static
Objective	Ensure all relevant ExpenseCategoryPolicy methods are non-static.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect ExpenseCategoryPolicy using ReflectionClass. 2. For each of the 7 methods, verify isStatic() is false.
Test Data	- Method names: viewAny, view, create, update, delete, restore, forceDelete
Expected Result	- All listed methods are non-static.
Actual Result	All listed methods were confirmed as non-static.
Status	Pass
Severity	High

Test Case ID	ECP-015
Title	viewAny Method Parameter Validation
Objective	Ensure viewAny accepts only a User parameter named 'user'.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect viewAny via ReflectionClass. 2. Check parameter count is 1. 3. Verify parameter name is 'user'.
Test Data	- Method inspected: 'viewAny' - Expected parameter: 'user'
Expected Result	- viewAny has one parameter named 'user'.
Actual Result	viewAny was confirmed to have one parameter named 'user'.
Status	Pass
Severity	High

Test Case ID	ECP-016
Title	view Method Parameter Validation
Objective	Ensure view method accepts User and ExpenseCategory parameters.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect view via ReflectionClass. 2. Check parameter count is 2. 3. Verify first parameter is 'user', second is 'expenseCategory'.
Test Data	- Method inspected: 'view' - Expected parameters: 'user', 'expenseCategory'
Expected Result	- view has two parameters named 'user' and 'expenseCategory'.
Actual Result	view confirmed to have two parameters: 'user', 'expenseCategory'.
Status	Pass
Severity	High

Test Case ID	ECP-017
Title	create Method Parameter Validation
Objective	Ensure create accepts only a User parameter named 'user'.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect create via ReflectionClass. 2. Check parameter count is 1. 3. Verify parameter name is 'user'.
Test Data	- Method inspected: 'create' - Expected parameter: 'user'
Expected Result	- create has one parameter named 'user'.
Actual Result	create confirmed to have one parameter named 'user'.
Status	Pass
Severity	High

Test Case ID	ECP-018
Title	update Method Parameter Validation
Objective	Ensure update method accepts User and ExpenseCategory parameters.

Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect update via ReflectionClass. 2. Check parameter count is 2. 3. Verify parameters are 'user' and 'expenseCategory'.
Test Data	- Method inspected: 'update' - Expected parameters: 'user', 'expenseCategory'
Expected Result	- update has two parameters named 'user' and 'expenseCategory'.
Actual Result	update confirmed to have two parameters: 'user', 'expenseCategory'.
Status	Pass
Severity	High

Test Case ID	ECP-019
Title	delete Method Parameter Validation
Objective	Ensure delete method accepts User and ExpenseCategory parameters.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect delete via ReflectionClass. 2. Check parameter count is 2. 3. Verify parameters are 'user' and 'expenseCategory'.
Test Data	- Method inspected: 'delete' - Expected parameters: 'user', 'expenseCategory'
Expected Result	- delete has two parameters named 'user' and 'expenseCategory'.
Actual Result	delete confirmed to have two parameters: 'user', 'expenseCategory'.
Status	Pass
Severity	High

Test Case ID	ECP-020
Title	restore Method Parameter Validation
Objective	Ensure restore method accepts User and ExpenseCategory parameters.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect restore via ReflectionClass. 2. Check parameter count is 2. 3. Verify parameters are 'user' and 'expenseCategory'.
Test Data	- Method inspected: 'restore' - Expected parameters: 'user', 'expenseCategory'
Expected Result	- restore has two parameters named 'user' and 'expenseCategory'.
Actual Result	restore confirmed to have two parameters: 'user', 'expenseCategory'.
Status	Pass
Severity	High

Test Case ID	ECP-021
Title	forceDelete Method Parameter Validation
Objective	Ensure forceDelete method accepts User and ExpenseCategory parameters.
Preconditions	- Application running - Classes autoloaded

Test Steps	1. Inspect forceDelete via ReflectionClass. 2. Check parameter count is 2. 3. Verify parameters are 'user' and 'expenseCategory'.
Test Data	- Method inspected: 'forceDelete' - Expected parameters: 'user', 'expenseCategory'
Expected Result	- forceDelete has two parameters named 'user' and 'expenseCategory'.
Actual Result	forceDelete confirmed to have two parameters: 'user', 'expenseCategory'.
Status	Pass
Severity	High

Test Case ID	ECP-022
Title	Multiple Instance Creation of ExpenseCategoryPolicy
Objective	Confirm multiple instances of ExpenseCategoryPolicy can be created without conflict.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Instantiate ExpenseCategoryPolicy twice. 2. Verify both instances are of correct class and are not the same object.
Test Data	- $policy1 = new ExpenseCategoryPolicy() - $policy2 = new ExpenseCategoryPolicy()
Expected Result	- Both instances are ExpenseCategoryPolicy and distinct from each other.
Actual Result	Both instances were distinct and of type ExpenseCategoryPolicy.
Status	Pass
Severity	Medium

Test Case ID	ECP-023
Title	Cloning ExpenseCategoryPolicy
Objective	Ensure ExpenseCategoryPolicy can be cloned and results in a distinct object.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Instantiate ExpenseCategoryPolicy. 2. Clone the instance. 3. Verify clone is a separate instance of ExpenseCategoryPolicy.
Test Data	- $policy = new ExpenseCategoryPolicy() - $clone = clone $policy
Expected Result	- Cloned instance is a new ExpenseCategoryPolicy object, not identical to the original.
Actual Result	Clone produced a distinct ExpenseCategoryPolicy instance.
Status	Pass
Severity	Medium

Test Case ID	ECP-024
Title	Type Hinting with ExpenseCategoryPolicy
Objective	Confirm ExpenseCategoryPolicy can be used as a type-hinted argument.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Define a function with type-hinted ExpenseCategoryPolicy argument. 2. Pass ExpenseCategoryPolicy instance and verify the function returns it.

Test Data	- Function parameter: ExpenseCategoryPolicy - Instance passed
Expected Result	- Function accepts and returns the ExpenseCategoryPolicy instance.
Actual Result	Function accepted and returned ExpenseCategoryPolicy instance correctly.
Status	Pass
Severity	Medium

Test Case ID	ECP-025
Title	ExpenseCategoryPolicy Class is Not Final
Objective	Ensure ExpenseCategoryPolicy is not declared as final.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect ExpenseCategoryPolicy using ReflectionClass. 2. Verify isFinal() returns false.
Test Data	- Class: ExpenseCategoryPolicy
Expected Result	- ExpenseCategoryPolicy is not final.
Actual Result	ExpenseCategoryPolicy confirmed as not final.
Status	Pass
Severity	Medium

Test Case ID	ECP-026
Title	ExpenseCategoryPolicy is Not an Interface
Objective	Ensure ExpenseCategoryPolicy is a class, not an interface.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect ExpenseCategoryPolicy using ReflectionClass. 2. Verify isInterface() returns false.
Test Data	- Class: ExpenseCategoryPolicy
Expected Result	- ExpenseCategoryPolicy is not an interface.
Actual Result	ExpenseCategoryPolicy confirmed as not an interface.
Status	Pass
Severity	Medium

Test Case ID	ECP-027
Title	ExpenseCategoryPolicy is Not a Trait
Objective	Ensure ExpenseCategoryPolicy is a class, not a trait.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect ExpenseCategoryPolicy using ReflectionClass. 2. Verify isTrait() returns false.
Test Data	- Class: ExpenseCategoryPolicy
Expected Result	- ExpenseCategoryPolicy is not a trait.
Actual Result	ExpenseCategoryPolicy confirmed as not a trait.
Status	Pass
Severity	Medium

Test Case ID	ECP-028

Title	ExpenseCategoryPolicy Class is Loaded
Objective	Confirm ExpenseCategoryPolicy is successfully loaded by the application.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Check existence of ExpenseCategoryPolicy using class_exists().
Test Data	- Class: ExpenseCategoryPolicy
Expected Result	- class_exists returns true.
Actual Result	class_exists returned true.
Status	Pass
Severity	Medium

Test Case ID	ECP-029
Title	Usage of Required Classes in ExpenseCategoryPolicy
Objective	Verify ExpenseCategoryPolicy uses all required classes in its file.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Get file contents of ExpenseCategoryPolicy. 2. Check for 'use' statements for required classes.
Test Data	- Required classes: - Crater\Models\Expense - Crater\Models\ExpenseCategory - Crater\Models\User - Illuminate\Auth\Access\HandlesAuthorization - Silber\Bouncer\BouncerFacade
Expected Result	- All required 'use' statements are present in the file.
Actual Result	All required 'use' statements found in the file.
Status	Pass
Severity	High

Test Case ID	ECP-030
Title	ExpenseCategoryPolicy File Structure Verification
Objective	Ensure ExpenseCategoryPolicy file contains expected structure and method signatures.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Retrieve ExpenseCategoryPolicy file contents. 2. Check for presence of 'class ExpenseCategoryPolicy'. 3. Verify usage of HandlesAuthorization and presence of all 7 method signatures.
Test Data	- Expected file lines: - 'class ExpenseCategoryPolicy' - 'use HandlesAuthorization' - 'public function viewAny' - 'public function view' - 'public function create' - 'public function update' - 'public function delete' - 'public function restore' - 'public function forceDelete'
Expected Result	- File contains all expected class and method/syntax lines.
Actual Result	File contains all required structure and method signatures.
Status	Pass

Severity	High

Test Case ID	ECP-031
Title	Reasonable Line Count for ExpenseCategoryPolicy File
Objective	Ensure the ExpenseCategoryPolicy file is of reasonable size.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Retrieve ExpenseCategoryPolicy file contents. 2. Count lines in the file. 3. Confirm line count is greater than 50 and less than 200.
Test Data	- Expected line count: 51-199
Expected Result	- File has >50 and <200 lines.
Actual Result	File contained a line count within expected limits.
Status	Pass
Severity	Low

Test Case ID	ECP-032
Title	Documentation on viewAny Method
Objective	Verify the presence of docblock documentation on the viewAny method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect 'viewAny' method using ReflectionClass. 2. Verify getDocComment() does not return false.
Test Data	- Method: viewAny
Expected Result	- Documentation exists for viewAny.
Actual Result	Documentation found for viewAny.
Status	Pass
Severity	Medium

Test Case ID	ECP-033
Title	Documentation on view Method
Objective	Verify the presence of docblock documentation on the view method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect 'view' method using ReflectionClass. 2. Verify getDocComment() does not return false.
Test Data	- Method: view
Expected Result	- Documentation exists for view.
Actual Result	Documentation found for view.
Status	Pass
Severity	Medium

Test Case ID	ECP-034
Title	Documentation on create Method
Objective	Verify the presence of docblock documentation on the create method.
Preconditions	- Application running - Classes autoloaded

Test Steps	1. Inspect 'create' method using ReflectionClass. 2. Verify getDocComment() does not return false.
Test Data	- Method: create
Expected Result	- Documentation exists for create.
Actual Result	Documentation found for create.
Status	Pass
Severity	Medium

Test Case ID	ECP-035
Title	Documentation on update Method
Objective	Verify the presence of docblock documentation on the update method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect 'update' method using ReflectionClass. 2. Verify getDocComment() does not return false.
Test Data	- Method: update
Expected Result	- Documentation exists for update.
Actual Result	Documentation found for update.
Status	Pass
Severity	Medium

Test Case ID	ECP-036
Title	Documentation on delete Method
Objective	Verify the presence of docblock documentation on the delete method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect 'delete' method using ReflectionClass. 2. Verify getDocComment() does not return false.
Test Data	- Method: delete
Expected Result	- Documentation exists for delete.
Actual Result	Documentation found for delete.
Status	Pass
Severity	Medium

Test Case ID	ECP-037
Title	Documentation on restore Method
Objective	Verify the presence of docblock documentation on the restore method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect 'restore' method using ReflectionClass. 2. Verify getDocComment() does not return false.
Test Data	- Method: restore
Expected Result	- Documentation exists for restore.
Actual Result	Documentation found for restore.
Status	Pass
Severity	Medium

Test Case ID	ECP-038
Title	Documentation on forceDelete Method
Objective	Verify the presence of docblock documentation on the forceDelete method.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect 'forceDelete' method using ReflectionClass. 2. Verify getDocComment() does not return false.
Test Data	- Method: forceDelete
Expected Result	- Documentation exists for forceDelete.
Actual Result	Documentation found for forceDelete.
Status	Pass
Severity	Medium

Test Case ID	ECP-039
Title	All Policy Methods Use BouncerFacade
Objective	Ensure all ExpenseCategoryPolicy methods make use of BouncerFacade for permissions.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Retrieve file contents for ExpenseCategoryPolicy. 2. Count number of 'BouncerFacade::can' calls. 3. Verify count is at least 7 (one per method).
Test Data	- Permission method: 'BouncerFacade::can'
Expected Result	- At least 7 occurrences of 'BouncerFacade::can' found.
Actual Result	At least 7 occurrences confirmed in ExpenseCategoryPolicy.
Status	Pass
Severity	High

Test Case ID	ECP-040
Title	view-expense Permission Checked in All Methods
Objective	Ensure all methods in ExpenseCategoryPolicy check for 'view-expense' permission.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Retrieve file contents for ExpenseCategoryPolicy. 2. Confirm 'view-expense' appears in permission checks.
Test Data	- Permission checked: 'view-expense'
Expected Result	- Each method verifies 'view-expense' permission.
Actual Result	'view-expense' permission confirmed as being checked in all methods.
Status	Pass
Severity	High

Test Case ID	ECP-041
Title	Expense Class Used in All Methods
Objective	Ensure all ExpenseCategoryPolicy methods reference the Expense class in permissions.
Preconditions	- Application running - Classes autoloaded

Test Steps	1. Retrieve file contents of ExpenseCategoryPolicy. 2. Count usage of 'Expense::class' in permission checks. 3. Confirm at least 7 occurrences.
Test Data	- Class referenced: 'Expense::class'
Expected Result	- Expense::class is referenced in all policy methods.
Actual Result	Expense::class found referenced in all methods as expected.
Status	Pass
Severity	High

Test Case ID	ECP-042
Title	Methods with Category Parameter Check hasCompany
Objective	Ensure all relevant methods check $user->hasCompany attribute.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Retrieve ExpenseCategoryPolicy file contents. 2. Count occurrences of '$user->hasCompany' in methods with an expenseCategory parameter. 3. Confirm count is exactly 5.
Test Data	- Attribute checked: '$user->hasCompany'
Expected Result	- $user->hasCompany checked in 5 methods with category parameter.
Actual Result	$user->hasCompany checked in correct methods.
Status	Pass
Severity	High

Test Case ID	ECP-043
Title	Methods with Category Parameter Check company_id
Objective	Ensure relevant methods check $expenseCategory->company_id.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Retrieve ExpenseCategoryPolicy file contents. 2. Check for usage of '$expenseCategory->company_id' in methods with an expenseCategory parameter.
Test Data	- Attribute checked: '$expenseCategory->company_id'
Expected Result	- Usage of $expenseCategory->company_id found in all relevant methods.
Actual Result	$expenseCategory->company_id found checked in methods as required.
Status	Pass
Severity	High

Test Case ID	ECP-044
Title	All Methods Return Boolean
Objective	Confirm all ExpenseCategoryPolicy methods return boolean values.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Retrieve ExpenseCategoryPolicy file contents. 2. Count number of 'return true' and 'return false' statements. 3. Verify counts match number of methods (7 each).
Test Data	- Expected return statements: 'return true', 'return false'
Expected Result	- Each method contains both return true and return false; total 7 of each.

Actual Result	All methods return proper boolean values.
Status	Pass
Severity	High

Test Case ID	ECP-045
Title	Exactly Seven Public Methods in ExpenseCategoryPolicy
Objective	Confirm ExpenseCategoryPolicy defines exactly 7 public methods.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Use ReflectionClass to retrieve all public methods. 2. Filter out inherited methods. 3. Count own public methods. 4. Verify count is 7.
Test Data	- Expected method count: 7
Expected Result	- ExpenseCategoryPolicy has exactly 7 public methods.
Actual Result	ExpenseCategoryPolicy confirmed to have 7 public methods.
Status	Pass
Severity	High

Test Case ID	ECP-046
Title	viewAny Method Structure Validation
Objective	Ensure viewAny method has a simple if-return structure and one parameter.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect viewAny using ReflectionClass. 2. Verify number of parameters is 1.
Test Data	- Method: viewAny - Expected parameter count: 1
Expected Result	- viewAny has one parameter, indicating simple logic structure.
Actual Result	viewAny confirmed with one parameter and simple structure.
Status	Pass
Severity	Medium

Test Case ID	ECP-047
Title	view Method Compound Condition Structure
Objective	Ensure view method uses compound if condition with logical '&&'.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Retrieve file content for ExpenseCategoryPolicy. 2. Check for compound condition 'BouncerFacade::can('view-expense', Expense::class) && $user->hasCompany'.
Test Data	- Expected code snippet: compound '&&' condition in view method
Expected Result	- Compound condition with '&&' is present in view method.
Actual Result	Compound condition found in view method as expected.
Status	Pass
Severity	High

Test Case ID	ECP-048

Title	create Method Structure Validation
Objective	Ensure create method has same parameter structure as viewAny.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect create using ReflectionClass. 2. Verify method has one parameter.
Test Data	- Method: create - Expected parameter count: 1
Expected Result	- create method has one parameter and simple structure.
Actual Result	create method confirmed with one parameter, matching structure of viewAny.
Status	Pass
Severity	Medium

Test Case ID	ECP-049
Title	update Method Structure Validation
Objective	Ensure update method has same parameter structure as view (compound condition, two parameters).
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect update via ReflectionClass. 2. Confirm method has two parameters.
Test Data	- Method: update - Expected parameter count: 2
Expected Result	- update method has two parameters, matching view method’s structure.
Actual Result	update method confirmed with two parameters and matching structure.
Status	Pass
Severity	High

Test Case ID	ECP-050
Title	delete Method Structure Validation
Objective	Ensure delete method has same parameter structure as view (compound condition, two parameters).
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect delete via ReflectionClass. 2. Confirm method has two parameters.
Test Data	- Method: delete - Expected parameter count: 2
Expected Result	- delete method has two parameters, matching view method’s structure.
Actual Result	delete method confirmed with two parameters and matching structure.
Status	Pass
Severity	High

Test Case ID	ECP-051
Title	restore Method Structure Validation
Objective	Ensure restore method has same structure and parameter count as view.
Preconditions	- Application running - Classes autoloaded

Test Steps	1. Inspect restore via ReflectionClass. 2. Verify method has two parameters.
Test Data	- Method: restore - Expected parameter count: 2
Expected Result	- restore method has two parameters, matching view method’s structure.
Actual Result	restore method confirmed with two parameters and matching structure.
Status	Pass
Severity	High

Test Case ID	ECP-052
Title	forceDelete Method Structure Validation
Objective	Ensure forceDelete method has same structure and parameter count as view.
Preconditions	- Application running - Classes autoloaded
Test Steps	1. Inspect forceDelete via ReflectionClass. 2. Verify method has two parameters.
Test Data	- Method: forceDelete - Expected parameter count: 2
Expected Result	- forceDelete method has two parameters, matching view method’s structure.
Actual Result	forceDelete method confirmed with two parameters and matching structure.
Status	Pass
Severity	High

File: ExpenseCategoryRequest-Test.txt

Test Case ID	ECR-001
Title	Authorization Always Returns True
Objective	Verify that the `authorize` method in ExpenseCategoryRequest always returns true.
Preconditions	- Application running - Database seeded - PHP Pest testing environment set up - ExpenseCategoryRequest class available
Test Steps	1. Instantiate an ExpenseCategoryRequest object. 2. Call the `authorize` method. 3. Assert that the result is true.
Test Data	- Request object: ExpenseCategoryRequest instance - No additional input
Expected Result	- The `authorize` method returns true.
Actual Result	The `authorize` method returned true as expected.
Status	Pass
Severity	Medium

Test Case ID	ECR-002
Title	Validation Rules Returned Correctly
Objective	Verify that the `rules` method returns correct validation rules for 'name' and 'description'.
Preconditions	- Application running - Database seeded - ExpenseCategoryRequest class available
Test Steps	1. Instantiate an ExpenseCategoryRequest object. 2. Call the `rules` method. 3. Assert the returned value is an array. 4. Assert that the keys 'name' and 'description' exist in the array. 5. Assert that 'name' rule is ['required']. 6. Assert that 'description' rule is ['nullable'].
Test Data	- Request object: ExpenseCategoryRequest instance
Expected Result	- The returned rules array contains 'name' => ['required'] and 'description' => ['nullable'].
Actual Result	Validation rules array was returned as expected with correct values.
Status	Pass
Severity	Medium

Test Case ID	ECR-003
Title	Expense Category Payload with All Fields
Objective	Verify that `getExpenseCategoryPayload()` returns correct payload with all expected fields when provided.
Preconditions	- Application running - Database seeded - ExpenseCategoryRequest class available - Mockery available for mocking methods

Test Steps	1. Mock an ExpenseCategoryRequest object. 2. Set up `validated` method to return: - name: "Test Category" - description: "A test description" 3. Set up `header('company')` to return 123. 4. Call `getExpenseCategoryPayload()`. 5. Assert returned payload is an array. 6. Assert payload contains 'name', 'description', and 'company_id' keys. 7. Assert payload values match the mocked data.
Test Data	- validated: ['name' => 'Test Category', 'description' => 'A test description'] - header('company'): 123
Expected Result	- Payload array with keys 'name', 'description', 'company_id' and corresponding expected values.
Actual Result	Payload returned with all fields and correct values as expected.
Status	Pass
Severity	High

Test Case ID	ECR-004
Title	Expense Category Payload with Null Description
Objective	Verify that `getExpenseCategoryPayload()` correctly sets description as null if validated description is null.
Preconditions	- Application running - Database seeded - ExpenseCategoryRequest class available - Mockery set up
Test Steps	1. Mock an ExpenseCategoryRequest object. 2. Set up `validated` to return: - name: "Another Category" - description: null 3. Set up `header('company')` to return 456. 4. Call `getExpenseCategoryPayload()`. 5. Assert returned payload is an array. 6. Assert payload contains 'name', 'description', and 'company_id'. 7. Assert description value is null.
Test Data	- validated: ['name' => 'Another Category', 'description' => null] - header('company'): 456
Expected Result	- Payload array with 'description' key having value null.
Actual Result	Payload contained 'description' key with value null as expected.
Status	Pass
Severity	High

Test Case ID	ECR-005
Title	Expense Category Payload without Description Key
Objective	Verify that `getExpenseCategoryPayload()` excludes 'description' when not present in validated data.
Preconditions	- Application running - Database seeded - ExpenseCategoryRequest class available - Mockery set up

Test Steps	1. Mock an ExpenseCategoryRequest object. 2. Set up `validated` to return: - name: "Category Without Desc" 3. Set up `header('company')` to return 789. 4. Call `getExpenseCategoryPayload()`. 5. Assert returned payload is an array. 6. Assert payload contains 'name' and 'company_id' keys only. 7. Assert 'description' key is absent.
Test Data	- validated: ['name' => 'Category Without Desc'] - header('company'): 789
Expected Result	- Payload array excludes 'description' key and includes 'name' and 'company_id'.
Actual Result	Payload did not include 'description' and contained expected keys as asserted.
Status	Pass
Severity	High

Test Case ID	ECR-006
Title	Expense Category Payload with Empty Validated Data
Objective	Verify that `getExpenseCategoryPayload()` handles empty validated array gracefully and returns only 'company_id'.
Preconditions	- Application running - Database seeded - ExpenseCategoryRequest class available - Mockery set up
Test Steps	1. Mock an ExpenseCategoryRequest object. 2. Set up `validated` to return an empty array. 3. Set up `header('company')` to return 101. 4. Call `getExpenseCategoryPayload()`. 5. Assert returned payload is an array. 6. Assert payload contains only 'company_id'. 7. Assert 'name' and 'description' are not present.
Test Data	- validated: [] - header('company'): 101
Expected Result	- Payload only contains 'company_id' key with value 101.
Actual Result	Payload contained only 'company_id' and excluded 'name' and 'description' as expected.
Status	Pass
Severity	High

Test Case ID	ECR-007
Title	Expense Category Payload with Null Company Header
Objective	Verify that `getExpenseCategoryPayload()` sets 'company_id' to null when header('company') is null.
Preconditions	- Application running - Database seeded - ExpenseCategoryRequest class available - Mockery set up
Test Steps	1. Mock an ExpenseCategoryRequest object. 2. Set up `validated` to return: - name: "Null Company Category" 3. Set up `header('company')` to return null. 4. Call `getExpenseCategoryPayload()`. 5. Assert returned payload is an array. 6. Assert 'company_id' key exists and is null.

Test Data	- validated: ['name' => 'Null Company Category'] - header('company'): null
Expected Result	- Payload has 'company_id' key with value null.
Actual Result	Payload returned with 'company_id' key set to null as expected.
Status	Pass
Severity	High

Test Case ID	ECR-008
Title	Expense Category Payload with Zero Company Header
Objective	Verify that `getExpenseCategoryPayload()` correctly handles 'company_id' being zero if header('company') returns 0.
Preconditions	- Application running - Database seeded - ExpenseCategoryRequest class available - Mockery set up
Test Steps	1. Mock an ExpenseCategoryRequest object. 2. Set up `validated` to return: - name: "Zero Company Category" 3. Set up `header('company')` to return 0. 4. Call `getExpenseCategoryPayload()`. 5. Assert returned payload is an array. 6. Assert 'company_id' key exists and is set to 0.
Test Data	- validated: ['name' => 'Zero Company Category'] - header('company'): 0
Expected Result	- Payload includes 'company_id' key set to 0.
Actual Result	Payload returned with 'company_id' key set to 0 as required.
Status	Pass
Severity	High

Test Case ID	ECR-009
Title	Expense Category Payload with Empty String Company Header
Objective	Verify that `getExpenseCategoryPayload()` correctly handles 'company_id' being an empty string if header('company') returns ''.
Preconditions	- Application running - Database seeded - ExpenseCategoryRequest class available - Mockery set up
Test Steps	1. Mock an ExpenseCategoryRequest object. 2. Set up `validated` to return: - name: "Empty String Company Category" 3. Set up `header('company')` to return '' (empty string). 4. Call `getExpenseCategoryPayload()`. 5. Assert returned payload is an array. 6. Assert 'company_id' key exists and is set to empty string.
Test Data	- validated: ['name' => 'Empty String Company Category'] - header('company'): ''
Expected Result	- Payload includes 'company_id' key set to empty string ('').
Actual Result	Payload returned with 'company_id' key as empty string as required.
Status	Pass
Severity	High

File: ExpenseCategoryResource-Test.txt

Test Case ID	ECR-001
Title	ExpenseCategoryResource object instantiation
Objective	Verify that ExpenseCategoryResource can be instantiated with a valid expense category entity.
Preconditions	- Application running - Expense category object available - Required classes autoloaded
Test Steps	1. Create a dummy expense category with id=1, name='Travel', amount=100. 2. Instantiate ExpenseCategoryResource with the dummy category. 3. Validate the instance belongs to ExpenseCategoryResource class.
Test Data	- id: 1 - name: 'Travel' - amount: 100
Expected Result	- ExpenseCategoryResource instance is created and is of correct class.
Actual Result	ExpenseCategoryResource instance was created as expected.
Status	Pass
Severity	Medium

Test Case ID	ECR-002
Title	ExpenseCategoryResource extends JsonResource
Objective	Verify that ExpenseCategoryResource inherits from JsonResource.
Preconditions	- Application running - ExpenseCategoryResource and JsonResource classes autoloaded
Test Steps	1. Create dummy expense category with id=1, name='Travel', amount=100. 2. Instantiate ExpenseCategoryResource. 3. Check if the instance is also of type JsonResource.
Test Data	- id: 1 - name: 'Travel' - amount: 100
Expected Result	- ExpenseCategoryResource instance is an instance of JsonResource.
Actual Result	Instance is correctly of type JsonResource.
Status	Pass
Severity	Medium

Test Case ID	ECR-003
Title	ExpenseCategoryResource namespace validation
Objective	Confirm that ExpenseCategoryResource resides in Crater\Http\Resources namespace.
Preconditions	- Application running - ExpenseCategoryResource class autoloaded
Test Steps	1. Use reflection on ExpenseCategoryResource class. 2. Retrieve the namespace of the class. 3. Validate namespace equals 'Crater\Http\Resources'.
Test Data	- N/A
Expected Result	- Namespace is 'Crater\Http\Resources'.
Actual Result	Namespace is correctly reported as 'Crater\Http\Resources'.
Status	Pass
Severity	Low

Test Case ID	ECR-004
Title	ExpenseCategoryResource has toArray method
Objective	Ensure that ExpenseCategoryResource has a 'toArray' method.
Preconditions	- Application running - ExpenseCategoryResource class autoloaded
Test Steps	1. Create dummy expense category. 2. Instantiate ExpenseCategoryResource. 3. Check for the existence of method 'toArray'.
Test Data	- id: 1 - name: 'Travel' - amount: 100
Expected Result	- 'toArray' method exists on ExpenseCategoryResource.
Actual Result	'toArray' method exists.
Status	Pass
Severity	Medium

Test Case ID	ECR-005
Title	toArray returns array with id
Objective	Confirm 'id' is included and correct in toArray output.
Preconditions	- Application running - ExpenseCategoryResource and Request classes available
Test Steps	1. Create dummy expense category with id=42. 2. Instantiate ExpenseCategoryResource. 3. Call toArray() with a request. 4. Assert result contains key 'id' and its value is 42.
Test Data	- id: 42
Expected Result	- Array output contains key 'id' with value 42.
Actual Result	'toArray' output has 'id' key set to 42.
Status	Pass
Severity	High

Test Case ID	ECR-006
Title	toArray returns array with name
Objective	Verify 'name' field is present and correct in toArray output.
Preconditions	- Application running
Test Steps	1. Create dummy expense category with name='Office Supplies'. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert result contains key 'name' with 'Office Supplies'.
Test Data	- name: 'Office Supplies'
Expected Result	- Output array contains 'name' => 'Office Supplies'
Actual Result	'toArray' output has 'name' key with correct value.
Status	Pass
Severity	High

Test Case ID	ECR-007
Title	toArray returns array with description

Objective	Ensure the 'description' field is correctly mapped in toArray output.
Preconditions	- Application running
Test Steps	1. Create dummy expense category with name='Travel'. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert result contains key 'description' as 'Description for Travel'.
Test Data	- name: 'Travel' - description: 'Description for Travel'
Expected Result	- Output array includes 'description' as 'Description for Travel'.
Actual Result	Description returned as expected.
Status	Pass
Severity	High

Test Case ID	ECR-008
Title	toArray returns array with company_id
Objective	Validate that company_id is present and correct in toArray output.
Preconditions	- Application running
Test Steps	1. Create dummy expense category with company_id=1. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert company_id equals 1.
Test Data	- company_id: 1
Expected Result	- Output array includes 'company_id' set to 1.
Actual Result	company_id set to 1 in output.
Status	Pass
Severity	High

Test Case ID	ECR-009
Title	toArray returns array with amount
Objective	Confirm 'amount' field inclusion and accuracy in toArray output.
Preconditions	- Application running
Test Steps	1. Create dummy expense category with amount=250.75. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert amount equals 250.75.
Test Data	- amount: 250.75
Expected Result	- 'amount' key is present and equals 250.75 in output.
Actual Result	Amount is 250.75 as expected.
Status	Pass
Severity	High

Test Case ID	ECR-010
Title	toArray returns array with formatted_created_at
Objective	Validate presence and correctness of formatted_created_at field.
Preconditions	- Application running

Test Steps	1. Create dummy expense category with formattedCreatedAt='2024-01-01 00:00:00'. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert formatted_created_at equals '2024-01-01 00:00:00'.
Test Data	- formattedCreatedAt: '2024-01-01 00:00:00'
Expected Result	- 'formatted_created_at' key appears with expected value.
Actual Result	Field present and correct.
Status	Pass
Severity	High

Test Case ID	ECR-011
Title	toArray includes all required fields
Objective	Ensure all required keys appear in toArray output.
Preconditions	- Application running
Test Steps	1. Create dummy expense category. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert output array has keys: 'id', 'name', 'description', 'company_id', 'amount', 'formatted_created_at', 'company'.
Test Data	- id, name, description, company_id, amount, formatted_created_at, company
Expected Result	- Output array includes all specified keys.
Actual Result	All required keys were present in output.
Status	Pass
Severity	High

Test Case ID	ECR-012
Title	toArray handles null id
Objective	Confirm that toArray correctly processes a null id value.
Preconditions	- Application running
Test Steps	1. Create dummy expense category with id=null. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Verify output 'id' is null.
Test Data	- id: null
Expected Result	- Output 'id' key has value null.
Actual Result	'id' returned as null.
Status	Pass
Severity	High

Test Case ID	ECR-013
Title	toArray handles null name
Objective	Ensure that a null name value is handled and returned by toArray.
Preconditions	- Application running
Test Steps	1. Create dummy expense category with name=null. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert 'name' key is null in output.
Test Data	- name: null

Expected Result	- 'name' appears with null value.
Actual Result	'name' key is null.
Status	Pass
Severity	High

Test Case ID	ECR-014
Title	toArray handles null description
Objective	Validate toArray can process a category with null description.
Preconditions	- Application running
Test Steps	1. Create expense category with description=null. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Check output 'description' is null.
Test Data	- description: null
Expected Result	- 'description' is null in output.
Actual Result	'description' field is null.
Status	Pass
Severity	High

Test Case ID	ECR-015
Title	toArray handles zero amount
Objective	Ensure toArray returns amount as zero when provided.
Preconditions	- Application running
Test Steps	1. Create expense category with amount=0. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert 'amount' equals 0 in output.
Test Data	- amount: 0
Expected Result	- 'amount' is zero as expected.
Actual Result	Zero amount returned.
Status	Pass
Severity	High

Test Case ID	ECR-016
Title	toArray handles negative amount
Objective	Confirm toArray processes negative 'amount' values correctly.
Preconditions	- Application running
Test Steps	1. Create expense category with amount=-50.25. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert 'amount' equals -50.25 in output.
Test Data	- amount: -50.25
Expected Result	- 'amount' is -50.25 in output.
Actual Result	Negative amount present.
Status	Pass
Severity	High

Test Case ID	ECR-017
Title	toArray handles different category names
Objective	Validate that multiple different names are handled and mapped correctly.
Preconditions	- Application running
Test Steps	1. For each name in ['Travel', 'Office Supplies', 'Utilities', 'Marketing']: a. Create dummy expense category with name. b. Instantiate ExpenseCategoryResource. c. Call toArray(). d. Assert 'name' equals expected value.
Test Data	- name: varies as above
Expected Result	- 'name' field matches expected across all inputs.
Actual Result	All names returned correctly.
Status	Pass
Severity	High

Test Case ID	ECR-018
Title	toArray handles different amounts
Objective	Ensure all provided amounts are accurately reflected in the output.
Preconditions	- Application running
Test Steps	1. For each amount in [0, 10.50, 100, 999.99, 1000.00, 5000.75]: a. Create dummy expense category with given amount. b. Instantiate ExpenseCategoryResource. c. Call toArray(). d. Assert amount matches input.
Test Data	- amount: see above
Expected Result	- Output 'amount' matches each test value.
Actual Result	All amounts mapped correctly.
Status	Pass
Severity	High

Test Case ID	ECR-019
Title	toArray includes company field
Objective	Confirm that the 'company' field is present in toArray output.
Preconditions	- Application running
Test Steps	1. Create dummy expense category. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Check for key 'company'.
Test Data	- category with company
Expected Result	- 'company' appears as a key in output.
Actual Result	'company' key present.
Status	Pass
Severity	High

Test Case ID	ECR-020
Title	toArray company field null scenario
Objective	Ensure 'company' is present and handled properly even when relationship does not exist.

Preconditions	- Application running
Test Steps	1. Create dummy expense category (with company()->exists() returns false). 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert array_key_exists('company', result) is true.
Test Data	- category with no company relation
Expected Result	- 'company' key exists, may be null or MissingValue.
Actual Result	'company' field present as expected.
Status	Pass
Severity	High

Test Case ID	ECR-021
Title	Expense category data integrity
Objective	Validate that all field values are preserved accurately in serialization.
Preconditions	- Application running
Test Steps	1. Create expense category with id=123, name='Marketing', amount=456.78. 2. Instantiate ExpenseCategoryResource. 3. Call toArray(). 4. Assert output matches original data.
Test Data	- id: 123 - name: 'Marketing' - amount: 456.78
Expected Result	- Output contains these values precisely.
Actual Result	All values matched exactly.
Status	Pass
Severity	High

Test Case ID	ECR-022
Title	Different instances have independent data
Objective	Confirm that separate instances do not share or overwrite data.
Preconditions	- Application running
Test Steps	1. Create category1: id=1, name='Travel', amount=100. 2. Create category2: id=2, name='Supplies', amount=200. 3. Create resource1 and resource2 for each. 4. Call toArray() and compare their values for id, name, amount.
Test Data	- category1: id=1, name='Travel', amount=100 - category2: id=2, name='Supplies', amount=200
Expected Result	- Values for id, name, amount are independent.
Actual Result	Data stayed independent for each instance.
Status	Pass
Severity	High

Test Case ID	ECR-023
Title	ExpenseCategoryResource class is not abstract
Objective	Ensure the class is not declared as abstract.
Preconditions	- Application running
Test Steps	1. Use reflection on ExpenseCategoryResource class. 2. Determine isAbstract property.
Test Data	- N/A

Expected Result	- isAbstract == false
Actual Result	Class is not abstract.
Status	Pass
Severity	Low

Test Case ID	ECR-024
Title	ExpenseCategoryResource class is not final
Objective	Confirm the class may be extended and is not final.
Preconditions	- Application running
Test Steps	1. Use reflection on ExpenseCategoryResource class. 2. Evaluate isFinal property.
Test Data	- N/A
Expected Result	- isFinal == false
Actual Result	Class not final.
Status	Pass
Severity	Low

Test Case ID	ECR-025
Title	ExpenseCategoryResource is not interface
Objective	Validate the class is not an interface.
Preconditions	- Application running
Test Steps	1. Use reflection on ExpenseCategoryResource. 2. Check isInterface property.
Test Data	- N/A
Expected Result	- isInterface == false
Actual Result	Class is not an interface.
Status	Pass
Severity	Low

Test Case ID	ECR-026
Title	ExpenseCategoryResource is not trait
Objective	Confirm the class is a regular class, not a trait.
Preconditions	- Application running
Test Steps	1. Use reflection on ExpenseCategoryResource. 2. Check isTrait property.
Test Data	- N/A
Expected Result	- isTrait == false
Actual Result	Not a trait.
Status	Pass
Severity	Low

Test Case ID	ECR-027
Title	ExpenseCategoryResource class is loaded
Objective	Ensure class definition is loaded in runtime.
Preconditions	- Application running

Test Steps	1. Check class_exists for ExpenseCategoryResource.
Test Data	- N/A
Expected Result	- class_exists returns true
Actual Result	Class is loaded.
Status	Pass
Severity	Low

Test Case ID	ECR-028
Title	toArray method is public
Objective	Validate that the toArray method is publicly accessible.
Preconditions	- Application running
Test Steps	1. Use reflection on ExpenseCategoryResource. 2. Retrieve 'toArray' method. 3. Confirm method is public.
Test Data	- N/A
Expected Result	- isPublic == true
Actual Result	Method is public.
Status	Pass
Severity	Medium

Test Case ID	ECR-029
Title	toArray method is not static
Objective	Ensure toArray is an instance method.
Preconditions	- Application running
Test Steps	1. Use reflection to get 'toArray' method. 2. Check isStatic property.
Test Data	- N/A
Expected Result	- isStatic == false
Actual Result	Method is not static.
Status	Pass
Severity	Medium

Test Case ID	ECR-030
Title	toArray method request parameter
Objective	Confirm toArray takes a single 'request' parameter.
Preconditions	- Application running
Test Steps	1. Use reflection on 'toArray' method. 2. Retrieve parameter list. 3. Confirm parameter count is one and name is 'request'.
Test Data	- N/A
Expected Result	- Parameter named 'request' is present.
Actual Result	Request parameter found.
Status	Pass
Severity	Medium

Test Case ID	ECR-031

Title	Multiple ExpenseCategoryResource instances creation
Objective	Validate that multiple independent instances can exist.
Preconditions	- Application running
Test Steps	1. Create two expense category objects. 2. Instantiate two ExpenseCategoryResource objects. 3. Assert both instances are correct and distinct.
Test Data	- Two categories
Expected Result	- Each instance is valid and unique.
Actual Result	Instances were unique and of correct class.
Status	Pass
Severity	Medium

Test Case ID	ECR-032
Title	ExpenseCategoryResource can be cloned
Objective	Test that an ExpenseCategoryResource instance can be cloned.
Preconditions	- Application running
Test Steps	1. Create an instance of ExpenseCategoryResource. 2. Clone the instance. 3. Assert cloned object is distinct and correct.
Test Data	- category: id=1, name='Travel', amount=100
Expected Result	- Clone is valid ExpenseCategoryResource and not the same reference.
Actual Result	Clone was distinct and correct type.
Status	Pass
Severity	Medium

Test Case ID	ECR-033
Title	ExpenseCategoryResource type hint compliance
Objective	Ensure the class works with type hinting in function signatures.
Preconditions	- Application running
Test Steps	1. Create a function with ExpenseCategoryResource type hint. 2. Invoke function with a valid instance. 3. Assert return value matches input.
Test Data	- instance of ExpenseCategoryResource
Expected Result	- Type hint passes and function returns instance.
Actual Result	Type hint accepted and instance returned.
Status	Pass
Severity	Low

Test Case ID	ECR-034
Title	ExpenseCategoryResource uses JsonResource
Objective	Validate that the file imports JsonResource.
Preconditions	- Application running
Test Steps	1. Use reflection to get file contents. 2. Check for import statement: use Illuminate\Http\Resources\Json\JsonResource.
Test Data	- N/A
Expected Result	- Import statement for JsonResource exists.

Actual Result	JsonResource was imported.
Status	Pass
Severity	Low

Test Case ID	ECR-035
Title	ExpenseCategoryResource uses CompanyResource
Objective	Confirm that the file references CompanyResource import.
Preconditions	- Application running
Test Steps	1. Use reflection to get file contents. 2. Check for 'CompanyResource' reference.
Test Data	- N/A
Expected Result	- 'CompanyResource' usage found.
Actual Result	CompanyResource imported.
Status	Pass
Severity	Low

Test Case ID	ECR-036
Title	toArray uses when helper for company data
Objective	Ensure toArray uses Laravel's 'when' helper for conditional company inclusion.
Preconditions	- Application running
Test Steps	1. Get file contents for ExpenseCategoryResource. 2. Search for '$this->when'.
Test Data	- N/A
Expected Result	- '$this->when' usage found.
Actual Result	'when' helper observed in code.
Status	Pass
Severity	Low

Test Case ID	ECR-037
Title	toArray checks company exists
Objective	Ensure toArray conditionally includes company based on existence.
Preconditions	- Application running
Test Steps	1. Get file contents. 2. Check for '$this->company()->exists()'.
Test Data	- N/A
Expected Result	- Presence of company existence check.
Actual Result	Company existence check found.
Status	Pass
Severity	Low

Test Case ID	ECR-038
Title	toArray creates CompanyResource when company exists
Objective	Validate that toArray constructs CompanyResource for company data.
Preconditions	- Application running
Test Steps	1. Get ExpenseCategoryResource file contents. 2. Search for 'new CompanyResource($this->company)'.

Test Data	- N/A
Expected Result	- Construction of CompanyResource found.
Actual Result	CompanyResource correctly instantiated.
Status	Pass
Severity	Low

Test Case ID	ECR-039
Title	toArray method has documentation
Objective	Ensure toArray is documented with a docblock.
Preconditions	- Application running
Test Steps	1. Use reflection to get 'toArray' method's doc comment. 2. Confirm doc comment exists.
Test Data	- N/A
Expected Result	- Doc comment is present.
Actual Result	Docblock present.
Status	Pass
Severity	Low

Test Case ID	ECR-040
Title	toArray has return type documentation
Objective	Check that toArray's docblock documents the return type.
Preconditions	- Application running
Test Steps	1. Get doc comment for 'toArray'. 2. Search for '@return'.
Test Data	- N/A
Expected Result	- '@return' type tag is present.
Actual Result	Return type documented.
Status	Pass
Severity	Low

Test Case ID	ECR-041
Title	toArray has param documentation
Objective	Verify parameter documentation in the docblock of toArray.
Preconditions	- Application running
Test Steps	1. Get doc comment of 'toArray'. 2. Check for '@param'.
Test Data	- N/A
Expected Result	- '@param' tag found in docblock.
Actual Result	Param tag present.
Status	Pass
Severity	Low

Test Case ID	ECR-042
Title	ExpenseCategoryResource file is concise
Objective	Validate file length does not exceed 2000 characters.

Preconditions	- Application running
Test Steps	1. Get ExpenseCategoryResource file size. 2. Ensure file size is < 2000 characters.
Test Data	- N/A
Expected Result	- File size less than 2000 chars.
Actual Result	File size is concise.
Status	Pass
Severity	Low

Test Case ID	ECR-043
Title	ExpenseCategoryResource has minimal line count
Objective	Ensure file lines for ExpenseCategoryResource class are less than 50.
Preconditions	- Application running
Test Steps	1. Count number of lines in ExpenseCategoryResource file. 2. Assert line count < 50.
Test Data	- N/A
Expected Result	- Line count less than 50
Actual Result	Line count is below threshold.
Status	Pass
Severity	Low

Test Case ID	ECR-044
Title	ExpenseCategoryResource parent is JsonResource
Objective	Confirm parent class of ExpenseCategoryResource is JsonResource.
Preconditions	- Application running
Test Steps	1. Use reflection to get parent class. 2. Assert parent class exists and is 'Illuminate\Http\Resources\Json\JsonResource'.
Test Data	- N/A
Expected Result	- Parent class is JsonResource.
Actual Result	Parent class confirmed.
Status	Pass
Severity	Low

File: FileDisk-Test.txt

Test Case ID	FD-001
Title	Instantiation of FileDisk
Objective	Verify that FileDisk object can be successfully instantiated.
Preconditions	- Application running - Database seeded - Crater\Models\FileDisk is autoloaded
Test Steps	1. Instantiate a new FileDisk object. 2. Check if the object is of type FileDisk.
Test Data	- Instantiation with default constructor
Expected Result	- The object is an instance of FileDisk::class.
Actual Result	Object is correctly instantiated as FileDisk.
Status	Pass
Severity	Medium

Test Case ID	FD-002
Title	FileDisk Extends Model
Objective	Verify that FileDisk class extends Illuminate\Database\Eloquent\Model.
Preconditions	- Application running - Database seeded - Crater\Models\FileDisk is autoloaded
Test Steps	1. Instantiate a FileDisk object. 2. Validate the object type against Illuminate\Database\Eloquent\Model.
Test Data	- FileDisk instance
Expected Result	- Object is also an instance of Model::class.
Actual Result	FileDisk object is an instance of Model.
Status	Pass
Severity	Medium

Test Case ID	FD-003
Title	FileDisk Namespace Verification
Objective	Ensure FileDisk is defined under Crater\Models namespace.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to inspect the namespace of FileDisk. 2. Retrieve namespace name.
Test Data	- Class reference: FileDisk::class
Expected Result	- Namespace name is 'Crater\Models'.
Actual Result	Namespace is 'Crater\Models'.
Status	Pass
Severity	Low

Test Case ID	FD-004
Title	FileDisk Not Abstract
Objective	Verify FileDisk is not defined as an abstract class.
Preconditions	- Application running

Test Steps	1. Use ReflectionClass on FileDisk. 2. Check isAbstract property.
Test Data	- Class reference: FileDisk::class
Expected Result	- isAbstract returns false.
Actual Result	FileDisk is not abstract.
Status	Pass
Severity	Medium

Test Case ID	FD-005
Title	FileDisk Is Instantiable
Objective	Check that FileDisk class can be instantiated.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Check isInstantiable property.
Test Data	- Class reference: FileDisk::class
Expected Result	- isInstantiable returns true.
Actual Result	FileDisk is instantiable.
Status	Pass
Severity	Medium

Test Case ID	FD-006
Title	DISK_TYPE_SYSTEM Constant Exists
Objective	Ensure FileDisk defines DISK_TYPE_SYSTEM constant with value 'SYSTEM'.
Preconditions	- Application running
Test Steps	1. Access FileDisk::DISK_TYPE_SYSTEM constant. 2. Verify its value.
Test Data	- None (code constant)
Expected Result	- DISK_TYPE_SYSTEM is set to 'SYSTEM'.
Actual Result	Constant value is 'SYSTEM'.
Status	Pass
Severity	Medium

Test Case ID	FD-007
Title	DISK_TYPE_REMOTE Constant Exists
Objective	Ensure FileDisk defines DISK_TYPE_REMOTE constant with value 'REMOTE'.
Preconditions	- Application running
Test Steps	1. Access FileDisk::DISK_TYPE_REMOTE constant. 2. Validate value.
Test Data	- None (code constant)
Expected Result	- DISK_TYPE_REMOTE is set to 'REMOTE'.
Actual Result	Constant value is 'REMOTE'.
Status	Pass
Severity	Medium

Test Case ID	FD-008
Title	Guarded Properties Contain 'id'
Objective	Ensure FileDisk has guarded properties including 'id'.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk object. 2. Retrieve guarded properties via getGuarded(). 3. Check for 'id' in array.
Test Data	- None
Expected Result	- 'id' is found in the guarded properties.
Actual Result	'id' is guarded.
Status	Pass
Severity	Medium

Test Case ID	FD-009
Title	Casts for set_as_default
Objective	Verify the set_as_default property is cast to boolean.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Call getCasts(). 3. Check that 'set_as_default' is present and value is 'boolean'.
Test Data	- None
Expected Result	- 'set_as_default' exists in casts. - Its value is 'boolean'.
Actual Result	Casts include 'set_as_default' as boolean.
Status	Pass
Severity	Medium

Test Case ID	FD-010
Title	FileDisk Uses HasFactory Trait
Objective	Confirm FileDisk class uses the HasFactory trait.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Retrieve trait names. 3. Check for 'Illuminate\Database\Eloquent\Factories\HasFactory'.
Test Data	- None
Expected Result	- 'HasFactory' trait is present.
Actual Result	Trait found as expected.
Status	Pass
Severity	Medium

Test Case ID	FD-011
Title	setCredentialsAttribute Method Exists
Objective	Verify presence of setCredentialsAttribute method in FileDisk.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check method existence using method_exists.
Test Data	- Method name: setCredentialsAttribute

Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-012
Title	scopeWhereOrder Method Exists
Objective	Verify presence of scopeWhereOrder method in FileDisk.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check method existence using method_exists.
Test Data	- Method name: scopeWhereOrder
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-013
Title	scopeFileDisksBetween Method Exists
Objective	Verify presence of scopeFileDisksBetween method in FileDisk.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check method existence.
Test Data	- Method name: scopeFileDisksBetween
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-014
Title	scopeWhereSearch Method Exists
Objective	Verify presence of scopeWhereSearch method.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check for 'scopeWhereSearch' method.
Test Data	- Method name: scopeWhereSearch
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-015
Title	scopePaginateData Method Exists
Objective	Verify presence of scopePaginateData method in FileDisk.
Preconditions	- Application running

Test Steps	1. Instantiate FileDisk. 2. Check method existence.
Test Data	- Method name: scopePaginateData
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-016
Title	scopeApplyFilters Method Exists
Objective	Verify presence of scopeApplyFilters method.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check method existence.
Test Data	- Method name: scopeApplyFilters
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-017
Title	setConfig Method Exists
Objective	Verify setConfig method is defined in FileDisk.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check method existence.
Test Data	- Method name: setConfig
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-018
Title	setAsDefault Method Exists
Objective	Verify setAsDefault method is defined.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check method existence.
Test Data	- Method name: setAsDefault
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-019
Title	setFilesystem Static Method Exists

Objective	Verify setFilesystem static method exists.
Preconditions	- Application running
Test Steps	1. Check method_exists on FileDisk::class for setFilesystem.
Test Data	- Method name: setFilesystem
Expected Result	- Static method exists.
Actual Result	Static method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-020
Title	validateCredentials Static Method Exists
Objective	Verify validateCredentials static method exists.
Preconditions	- Application running
Test Steps	1. Check method_exists on FileDisk::class for validateCredentials.
Test Data	- Method name: validateCredentials
Expected Result	- Static method exists.
Actual Result	Static method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-021
Title	createDisk Static Method Exists
Objective	Verify createDisk static method exists.
Preconditions	- Application running
Test Steps	1. Check method_exists on FileDisk::class for createDisk.
Test Data	- Method name: createDisk
Expected Result	- Static method exists.
Actual Result	Static method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-022
Title	updateDefaultDisks Static Method Exists
Objective	Verify updateDefaultDisks static method exists.
Preconditions	- Application running
Test Steps	1. Check method_exists on FileDisk::class for updateDefaultDisks.
Test Data	- Method name: updateDefaultDisks
Expected Result	- Static method exists.
Actual Result	Static method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-023
Title	updateDisk Method Exists

Objective	Verify updateDisk method is present in FileDisk.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check method existence.
Test Data	- Method name: updateDisk
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-024
Title	setAsDefaultDisk Method Exists
Objective	Verify setAsDefaultDisk method is present.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check method existence.
Test Data	- Method name: setAsDefaultDisk
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-025
Title	isSystem Method Exists
Objective	Confirm isSystem method is defined.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Verify method existence.
Test Data	- Method name: isSystem
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-026
Title	isRemote Method Exists
Objective	Confirm isRemote method is defined.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Verify method existence.
Test Data	- Method name: isRemote
Expected Result	- Method exists.
Actual Result	Method exists.
Status	Pass
Severity	Medium

Test Case ID	FD-027
Title	setCredentialsAttribute Encodes Array to JSON
Objective	Verify setCredentialsAttribute serializes array to JSON in attributes.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Prepare an array ['key' => 'value', 'secret' => '123']. 3. Pass array to setCredentialsAttribute. 4. Use reflection to access attributes. 5. Validate that credentials field is JSON-encoded.
Test Data	- Credentials: ['key' => 'value', 'secret' => '123']
Expected Result	- 'credentials' attribute is '{"key":"value","secret":"123"}'
Actual Result	Credentials attribute is JSON-encoded as expected.
Status	Pass
Severity	Medium

Test Case ID	FD-028
Title	setCredentialsAttribute Handles Empty Array
Objective	Ensure setCredentialsAttribute serializes empty array as '[]'.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Call setCredentialsAttribute([]). 3. Use reflection to access attributes. 4. Validate credentials field is set to '[]'.
Test Data	- Credentials: []
Expected Result	- 'credentials' attribute is '[]'
Actual Result	Credentials attribute correctly set to '[]'.
Status	Pass
Severity	Medium

Test Case ID	FD-029
Title	setAsDefault Returns set_as_default Attribute True
Objective	Confirm setAsDefault returns true when set_as_default is true.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Set set_as_default to true. 3. Call setAsDefault. 4. Assert return value is true.
Test Data	- set_as_default = true
Expected Result	- setAsDefault() returns true.
Actual Result	Method returned true.
Status	Pass
Severity	Medium

Test Case ID	FD-030
Title	setAsDefault Returns False When Not Set
Objective	Confirm setAsDefault returns false when set_as_default is false.
Preconditions	- Application running

Test Steps	1. Instantiate FileDisk. 2. Set set_as_default to false. 3. Call setAsDefault. 4. Assert return value is false.
Test Data	- set_as_default = false
Expected Result	- setAsDefault() returns false.
Actual Result	Method returned false.
Status	Pass
Severity	Medium

Test Case ID	FD-031
Title	isSystem Returns True for SYSTEM Type
Objective	Confirm isSystem returns true when type is SYSTEM.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Set type to FileDisk::DISK_TYPE_SYSTEM. 3. Call isSystem. 4. Assert return value is true.
Test Data	- type = 'SYSTEM'
Expected Result	- isSystem() returns true.
Actual Result	Returned true for SYSTEM type.
Status	Pass
Severity	Medium

Test Case ID	FD-032
Title	isSystem Returns False for Non-SYSTEM Type
Objective	Confirm isSystem returns false when type is REMOTE.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Set type to FileDisk::DISK_TYPE_REMOTE. 3. Call isSystem. 4. Assert return value is false.
Test Data	- type = 'REMOTE'
Expected Result	- isSystem() returns false.
Actual Result	Returned false for REMOTE type.
Status	Pass
Severity	Medium

Test Case ID	FD-033
Title	isRemote Returns True for REMOTE Type
Objective	Confirm isRemote returns true when type is REMOTE.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Set type to FileDisk::DISK_TYPE_REMOTE. 3. Call isRemote. 4. Assert return value is true.
Test Data	- type = 'REMOTE'
Expected Result	- isRemote() returns true.

Actual Result	Returned true for REMOTE type.
Status	Pass
Severity	Medium

Test Case ID	FD-034
Title	isRemote Returns False for Non-REMOTE Type
Objective	Confirm isRemote returns false when type is SYSTEM.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Set type to FileDisk::DISK_TYPE_SYSTEM. 3. Call isRemote. 4. Assert return value is false.
Test Data	- type = 'SYSTEM'
Expected Result	- isRemote() returns false.
Actual Result	Returned false for SYSTEM type.
Status	Pass
Severity	Medium

Test Case ID	FD-035
Title	All Scope Methods Are Public
Objective	Verify all scope methods are declared public.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Check access level of scopeWhereOrder, scopeFileDisksBetween, scopeWhereSearch, scopePaginateData, scopeApplyFilters.
Test Data	- Method list
Expected Result	- All methods are public.
Actual Result	All scope methods confirmed as public.
Status	Pass
Severity	Medium

Test Case ID	FD-036
Title	Static Methods Are Static
Objective	Verify setFilesystem, validateCredentials, createDisk, updateDefaultDisks are static methods.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Check isStatic for each target method.
Test Data	- Method list
Expected Result	- All specified methods are static.
Actual Result	Methods are static.
Status	Pass
Severity	Medium

Test Case ID	FD-037
Title	Instance Methods Are Not Static
Objective	Verify certain instance methods are not static.

Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Check isStatic property for setAsDefault, updateDisk, setAsDefaultDisk, isSystem, isRemote.
Test Data	- Method list
Expected Result	- All specified methods are not static.
Actual Result	Methods confirmed as non-static.
Status	Pass
Severity	Medium

Test Case ID	FD-038
Title	scopeWhereOrder Accepts Three Parameters
Objective	Confirm scopeWhereOrder accepts three correct parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Retrieve scopeWhereOrder method parameters. 3. Check count and names.
Test Data	- N/A
Expected Result	- Parameter count is 3: query, orderByField, orderBy.
Actual Result	Parameters verified.
Status	Pass
Severity	Medium

Test Case ID	FD-039
Title	scopeFileDisksBetween Accepts Three Parameters
Objective	Confirm scopeFileDisksBetween accepts start, end, query.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Retrieve method parameters. 3. Check count and names.
Test Data	- N/A
Expected Result	- Parameter count is 3: query, start, end.
Actual Result	Parameters verified.
Status	Pass
Severity	Medium

Test Case ID	FD-040
Title	scopeWhereSearch Accepts Two Parameters
Objective	Confirm scopeWhereSearch accepts two parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Retrieve method parameters. 3. Check count and names.
Test Data	- N/A
Expected Result	- Parameter count is 2: query, search.
Actual Result	Parameters verified.
Status	Pass

Severity	Medium

Test Case ID	FD-041
Title	scopeApplyFilters Accepts Two Parameters
Objective	Confirm scopeApplyFilters has correct parameters.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass. 2. Retrieve method parameters. 3. Assert there are two: query, filters.
Test Data	- N/A
Expected Result	- Parameter count is 2: query, filters.
Actual Result	Parameters verified.
Status	Pass
Severity	Medium

Test Case ID	FD-042
Title	Multiple FileDisk Instances
Objective	Verify multiple FileDisk instances are independent.
Preconditions	- Application running
Test Steps	1. Instantiate two FileDisk objects. 2. Verify both are instances of FileDisk. 3. Ensure instances are not the same.
Test Data	- Two instances
Expected Result	- Both are FileDisk objects, not equal.
Actual Result	Instances are independent.
Status	Pass
Severity	Medium

Test Case ID	FD-043
Title	FileDisk Cloning
Objective	Confirm FileDisk object can be cloned.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Clone the instance. 3. Verify clone is FileDisk. 4. Ensure clone is not the same as original.
Test Data	- FileDisk clone
Expected Result	- Clone is FileDisk, not identical to original.
Actual Result	Clone successful.
Status	Pass
Severity	Medium

Test Case ID	FD-044
Title	FileDisk Type Hinting
Objective	Verify FileDisk can be used as type hint in function.
Preconditions	- Application running

Test Steps	1. Define a function accepting a FileDisk argument. 2. Pass a FileDisk instance. 3. Verify function returns the same instance.
Test Data	- Test function: function(FileDisk $disk)
Expected Result	- Instance is returned unchanged.
Actual Result	Type hint works correctly.
Status	Pass
Severity	Medium

Test Case ID	FD-045
Title	FileDisk is Not Final
Objective	Ensure FileDisk is not declared final.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Check isFinal property.
Test Data	- N/A
Expected Result	- isFinal is false.
Actual Result	FileDisk is not final.
Status	Pass
Severity	Medium

Test Case ID	FD-046
Title	FileDisk is Not Interface
Objective	Verify FileDisk is a class, not an interface.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Check isInterface property.
Test Data	- N/A
Expected Result	- isInterface is false.
Actual Result	Class is not an interface.
Status	Pass
Severity	Medium

Test Case ID	FD-047
Title	FileDisk is Not a Trait
Objective	Ensure FileDisk is a class and not a trait.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Check isTrait property.
Test Data	- N/A
Expected Result	- isTrait is false.
Actual Result	Class is not a trait.
Status	Pass
Severity	Medium

Test Case ID	FD-048

Title	FileDisk Class is Loaded
Objective	Ensure FileDisk class is loaded and accessible.
Preconditions	- Application running
Test Steps	1. Check class_exists(FileDisk::class).
Test Data	- N/A
Expected Result	- class_exists returns true.
Actual Result	Class is loaded.
Status	Pass
Severity	Medium

Test Case ID	FD-049
Title	Required Classes Imported in FileDisk
Objective	Ensure FileDisk imports all required classes.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Read file contents. 3. Check for use statements for Crater\Carbon, HasFactory, Model.
Test Data	- N/A
Expected Result	- Use statements for all required classes present.
Actual Result	Imports detected.
Status	Pass
Severity	Medium

Test Case ID	FD-050
Title	FileDisk File Structure Verification
Objective	Ensure FileDisk file contains class extension, guarded and casts properties.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass. 2. Read file contents. 3. Verify existence of 'class FileDisk extends Model', 'protected $guarded', 'protected $casts'.
Test Data	- N/A
Expected Result	- All structures are present.
Actual Result	FileDisk structured as expected.
Status	Pass
Severity	Medium

Test Case ID	FD-051
Title	FileDisk Reasonable Line Count
Objective	Ensure FileDisk file has expected size (100 < lines < 300).
Preconditions	- Application running
Test Steps	1. Use ReflectionClass. 2. Read file contents. 3. Count lines. 4. Check if line count is within acceptable bounds.
Test Data	- N/A
Expected Result	- Line count is >100 and <300.

Actual Result	Line count is within range.
Status	Pass
Severity	Low

Test Case ID	FD-052
Title	scopeWhereSearch Uses LIKE Operator
Objective	Ensure scopeWhereSearch implements search with LIKE SQL operator.
Preconditions	- Application running
Test Steps	1. Read FileDisk code. 2. Check usage of 'LIKE' and '%'.$term.'%'.
Test Data	- Search term
Expected Result	- 'LIKE' operator used in SQL.
Actual Result	LIKE operator found.
Status	Pass
Severity	Medium

Test Case ID	FD-053
Title	scopeWhereSearch Searches Name and Driver Fields
Objective	Ensure search functionality covers both name and driver columns.
Preconditions	- Application running
Test Steps	1. Inspect FileDisk source. 2. Check for where on 'name' and 'driver' fields.
Test Data	- N/A
Expected Result	- Both fields searched with 'LIKE'.
Actual Result	Both fields are used for searching.
Status	Pass
Severity	Medium

Test Case ID	FD-054
Title	scopePaginateData Handles 'all' Limit
Objective	Ensure scopePaginateData supports non-numeric 'all' limit case.
Preconditions	- Application running
Test Steps	1. Inspect FileDisk code. 2. Check for 'if ($limit == 'all')'. 3. Confirm query->get() used.
Test Data	- Limit = 'all'
Expected Result	- Returns complete query data set.
Actual Result	'All' limit handled as expected.
Status	Pass
Severity	Medium

Test Case ID	FD-055
Title	scopePaginateData Uses Paginate for Numeric Limit
Objective	Validate that numeric limits invoke paginate().
Preconditions	- Application running

Test Steps	1. Inspect FileDisk code. 2. Check for 'return $query->paginate($limit)'.
Test Data	- Limit = numeric value
Expected Result	- Calls paginate($limit) for numeric input.
Actual Result	Pagination with numeric limit performed.
Status	Pass
Severity	Medium

Test Case ID	FD-056
Title	scopeApplyFilters Uses Collect Helper
Objective	Ensure use of collect() helper for $filters variable.
Preconditions	- Application running
Test Steps	1. Inspect code for 'collect($filters)'.
Test Data	- Filters array
Expected Result	- Filters converted via collect().
Actual Result	Use of collect detected.
Status	Pass
Severity	Medium

Test Case ID	FD-057
Title	scopeApplyFilters Checks Search Filter
Objective	Confirm filter processing for the 'search' field.
Preconditions	- Application running
Test Steps	1. Inspect FileDisk code. 2. Check for '$filters->get('search')'. 3. Assert query->whereSearch is triggered.
Test Data	- filters['search']
Expected Result	- Search filter is checked and handled.
Actual Result	Check performed.
Status	Pass
Severity	Medium

Test Case ID	FD-058
Title	scopeApplyFilters Checks Date Range Filters
Objective	Ensure filters 'from_date' and 'to_date' are processed.
Preconditions	- Application running
Test Steps	1. Check FileDisk code for '$filters->get('from_date')' and '$filters->get('to_date')'. 2. Verify fileDisksBetween invoked.
Test Data	- filters['from_date'], filters['to_date']
Expected Result	- Date range filters invoke fileDisksBetween.
Actual Result	Date range filtering present.
Status	Pass
Severity	Medium

Test Case ID	FD-059

Title	scopeApplyFilters Checks Order Filters
Objective	Validate orderByField and orderBy filters in scopeApplyFilters.
Preconditions	- Application running
Test Steps	1. Inspect code for '$filters->get('orderByField')' and '$filters->get('orderBy')'. 2. Check invocation of whereOrder.
Test Data	- filters['orderByField'], filters['orderBy']
Expected Result	- Order filters are checked and handled by whereOrder.
Actual Result	Order filtering present.
Status	Pass
Severity	Medium

Test Case ID	FD-060
Title	setFilesystem Uses env Helper
Objective	Ensure env('DYNAMIC_DISK_PREFIX', 'temp_') is used in setFilesystem.
Preconditions	- Application running
Test Steps	1. Inspect FileDisk code. 2. Find 'env' usage for the disk prefix.
Test Data	- N/A
Expected Result	- env helper used for prefix retrieval.
Actual Result	env helper usage confirmed.
Status	Pass
Severity	Medium

Test Case ID	FD-061
Title	Set and Get Name Attribute
Objective	Validate the ability to set and retrieve the 'name' property.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Assign name to 'Test Disk'. 3. Retrieve and verify value.
Test Data	- name = 'Test Disk'
Expected Result	- Retrieving name returns 'Test Disk'.
Actual Result	Attribute set and retrieved correctly.
Status	Pass
Severity	Medium

Test Case ID	FD-062
Title	Set and Get Driver Attribute
Objective	Confirm setting and getting of 'driver' property.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Assign driver to 's3'. 3. Retrieve and verify value.
Test Data	- driver = 's3'
Expected Result	- Retrieving driver returns 's3'.
Actual Result	Attribute set and retrieved correctly.

Status	Pass
Severity	Medium

Test Case ID	FD-063
Title	Set and Get set_as_default Attribute
Objective	Confirm setting and getting of 'set_as_default' property.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Assign set_as_default to true. 3. Retrieve and verify value.
Test Data	- set_as_default = true
Expected Result	- Retrieving set_as_default returns true.
Actual Result	set_as_default set and retrieved as true.
Status	Pass
Severity	Medium

Test Case ID	FD-064
Title	Set and Get Type Attribute
Objective	Confirm type property can be set and retrieved.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Assign type to DISK_TYPE_SYSTEM. 3. Retrieve and verify.
Test Data	- type = 'SYSTEM'
Expected Result	- Retrieving type returns 'SYSTEM'.
Actual Result	Type set and retrieved as expected.
Status	Pass
Severity	Medium

Test Case ID	FD-065
Title	Independent Data Between FileDisk Instances
Objective	Verify each FileDisk instance maintains separate data.
Preconditions	- Application running
Test Steps	1. Instantiate disk1 and assign name 'Disk 1'. 2. Instantiate disk2 and assign name 'Disk 2'. 3. Assert names are not equal and correspond to set values.
Test Data	- disk1: name = 'Disk 1' - disk2: name = 'Disk 2'
Expected Result	- disk1 name is 'Disk 1' - disk2 name is 'Disk 2' - disk1 name != disk2 name
Actual Result	Data independence confirmed.
Status	Pass
Severity	Medium

Test Case ID	FD-066
Title	FileDisk Parent Is Model

Objective	Verify FileDisk parent class is Illuminate\Database\Eloquent\Model.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDisk. 2. Get parent class. 3. Validate name.
Test Data	- N/A
Expected Result	- Parent class exists and is 'Illuminate\Database\Eloquent\Model'.
Actual Result	Parent class verified.
Status	Pass
Severity	Medium

Test Case ID	FD-067
Title	FileDisk Inherits Model Methods
Objective	Confirm essential Model methods are inherited by FileDisk.
Preconditions	- Application running
Test Steps	1. Instantiate FileDisk. 2. Check existence of 'save', 'fill', 'toArray' via method_exists.
Test Data	- N/A
Expected Result	- All methods ('save', 'fill', 'toArray') exist.
Actual Result	Model methods inherited.
Status	Pass
Severity	Medium

File: FileDiskCollection-Test.txt

Test Case ID	FDC-001
Title	Instantiation of FileDiskCollection
Objective	Verify that the FileDiskCollection class can be instantiated.
Preconditions	- Application running - All dependencies installed - Autoload and class mapping available
Test Steps	1. Attempt to instantiate a new FileDiskCollection object with an empty Collection.
Test Data	- Input: new Collection([]) - Expected type: FileDiskCollection
Expected Result	- The object is successfully instantiated and is an instance of FileDiskCollection.
Actual Result	The object was created and confirmed to be a FileDiskCollection instance.
Status	Pass
Severity	Low

Test Case ID	FDC-002
Title	FileDiskCollection Extends ResourceCollection
Objective	Verify that FileDiskCollection is a subclass of ResourceCollection.
Preconditions	- Application running
Test Steps	1. Instantiate FileDiskCollection with an empty Collection. 2. Verify the object is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Test Data	- Input: new Collection([])
Expected Result	- FileDiskCollection object is also an instance of ResourceCollection.
Actual Result	The object is a valid instance of ResourceCollection.
Status	Pass
Severity	Low

Test Case ID	FDC-003
Title	FileDiskCollection Namespace Verification
Objective	Ensure FileDiskCollection class resides in the correct namespace.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on FileDiskCollection. 2. Check the namespace name returned.
Test Data	- Class: FileDiskCollection::class
Expected Result	- Namespace should be 'Crater\Http\Resources'.
Actual Result	Namespace identified as 'Crater\Http\Resources'.
Status	Pass
Severity	Low

Test Case ID	FDC-004
Title	FileDiskCollection has toArray Method
Objective	Confirm that FileDiskCollection defines a public toArray method.
Preconditions	- Application running

Test Steps	1. Instantiate FileDiskCollection. 2. Use method_exists to check for 'toArray'.
Test Data	- Method name: 'toArray'
Expected Result	- Method 'toArray' is present on the class.
Actual Result	Method 'toArray' exists as specified.
Status	Pass
Severity	Low

Test Case ID	FDC-005
Title	toArray Method is Public
Objective	Ensure that the toArray method in FileDiskCollection is public.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get method details. 2. Verify the method’s visibility via isPublic().
Test Data	- Method: 'toArray'
Expected Result	- Method is public.
Actual Result	Method confirmed to be public.
Status	Pass
Severity	Low

Test Case ID	FDC-006
Title	toArray Accepts Request Parameter
Objective	Verify that the toArray method takes a Request parameter.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to obtain toArray method. 2. Check the parameter count and the name of the first parameter.
Test Data	- Method: toArray
Expected Result	- Method has one parameter named 'request'.
Actual Result	Parameter count and name matches expected.
Status	Pass
Severity	Low

Test Case ID	FDC-007
Title	Handles Empty Collection Input
Objective	Ensure toArray returns an empty array when given an empty collection.
Preconditions	- Application running
Test Steps	1. Create a Request instance. 2. Instantiate FileDiskCollection with an empty Collection. 3. Invoke toArray with the Request.
Test Data	- Input: []
Expected Result	- Output is an empty array.
Actual Result	Returned array is empty as expected.
Status	Pass
Severity	Medium

Test Case ID	FDC-008

Title	Transforms Single FileDisk Resource
Objective	Verify that toArray correctly transforms a single FileDiskResource.
Preconditions	- Application running
Test Steps	1. Create a dummy file disk with id=1, name='S3 Disk'. 2. Wrap it with FileDiskResource. 3. Place resource in FileDiskCollection. 4. Call toArray with Request.
Test Data	- Disk: {id: 1, name: 'S3 Disk', ...}
Expected Result	- Output is an array with one element containing 'id' key with value 1.
Actual Result	Output contains expected id and structure.
Status	Pass
Severity	High

Test Case ID	FDC-009
Title	Transforms Multiple FileDisk Resources
Objective	Verify that toArray correctly transforms multiple FileDiskResource objects.
Preconditions	- Application running
Test Steps	1. Create two dummy file disks: (id=1, 'S3 Disk'), (id=2, 'Local Disk'). 2. Wrap each in FileDiskResource. 3. Place in FileDiskCollection. 4. Call toArray with Request.
Test Data	- Disks: [{id:1, name:'S3 Disk'}, {id:2, name:'Local Disk'}]
Expected Result	- Output array has 2 items: first with id 1, second with id 2.
Actual Result	Array returns two transformed items with correct IDs.
Status	Pass
Severity	High

Test Case ID	FDC-010
Title	All Required Fields Present in Transformed Item
Objective	Verify that each transformed disk has the required fields: id, name, driver.
Preconditions	- Application running
Test Steps	1. Create dummy disk with id=1, name='S3 Disk'. 2. Wrap in FileDiskResource and FileDiskCollection. 3. Call toArray and check keys.
Test Data	- Disk: {id:1, name:'S3 Disk'}
Expected Result	- Output includes keys: 'id', 'name', 'driver'.
Actual Result	All required fields are present in the result.
Status	Pass
Severity	High

Test Case ID	FDC-011
Title	Handles Large Collection of FileDisks
Objective	Confirm that large collections (e.g., 50 items) are processed correctly and efficiently.
Preconditions	- Application running

Test Steps	1. Create 50 dummy disks with ids 1-50. 2. Wrap each disk in FileDiskResource. 3. Collect into FileDiskCollection. 4. Invoke toArray with Request.
Test Data	- Disks: [{id:1,...}, ..., {id:50,...}]
Expected Result	- Output array has 50 elements, first with id=1, last with id=50.
Actual Result	All 50 disks present in output array with correct ids.
Status	Pass
Severity	High

Test Case ID	FDC-012
Title	toArray Delegates to Parent ResourceCollection
Objective	Ensure that FileDiskCollection's toArray method calls parent::toArray.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to read source code. 2. Check if 'parent::toArray' exists in FileDiskCollection file.
Test Data	- File content inspection
Expected Result	- Source contains 'parent::toArray'.
Actual Result	Code includes required delegation.
Status	Pass
Severity	Medium

Test Case ID	FDC-013
Title	Parent Class is ResourceCollection
Objective	Confirm that FileDiskCollection's parent class is ResourceCollection.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get parent class. 2. Verify parent class name.
Test Data	- Reflection data
Expected Result	- Parent class is Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	Parent class verified as ResourceCollection.
Status	Pass
Severity	Low

Test Case ID	FDC-014
Title	Multiple FileDiskCollection Instances
Objective	Verify that multiple instances of FileDiskCollection can be created and are distinct.
Preconditions	- Application running
Test Steps	1. Instantiate two separate FileDiskCollection objects. 2. Confirm both are instances and not equal.
Test Data	- Input: Two new Collection([])
Expected Result	- Both objects are FileDiskCollection, reference is not the same.
Actual Result	Instances created and mutually distinct.
Status	Pass
Severity	Low

Test Case ID	FDC-015
Title	Cloning FileDiskCollection Works Correctly
Objective	Ensure FileDiskCollection supports cloning and the clone is distinct from the original.
Preconditions	- Application running
Test Steps	1. Create FileDiskCollection instance. 2. Clone the instance. 3. Confirm both are instances and not the same object.
Test Data	- Input: FileDiskCollection object
Expected Result	- Cloned object is a FileDiskCollection and not the same reference as original.
Actual Result	Clone successful and distinct.
Status	Pass
Severity	Low

Test Case ID	FDC-016
Title	FileDiskCollection Usable in Type Hints
Objective	Confirm FileDiskCollection can be used as a parameter type.
Preconditions	- Application running
Test Steps	1. Create a function with FileDiskCollection as a parameter. 2. Pass a FileDiskCollection instance to the function. 3. Confirm it is received and equals input.
Test Data	- Input: FileDiskCollection instance
Expected Result	- Function accepts FileDiskCollection and returns it as is.
Actual Result	Type hints work and input is returned.
Status	Pass
Severity	Low

Test Case ID	FDC-017
Title	FileDiskCollection is Not Abstract
Objective	Ensure FileDiskCollection class is not abstract.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to check isAbstract().
Test Data	- Class: FileDiskCollection
Expected Result	- isAbstract returns false.
Actual Result	Class is not abstract.
Status	Pass
Severity	Low

Test Case ID	FDC-018
Title	FileDiskCollection is Not Final
Objective	Ensure FileDiskCollection is not marked as final.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass and check isFinal().
Test Data	- Class: FileDiskCollection
Expected Result	- isFinal returns false.
Actual Result	Class confirmed not final.

Status	Pass
Severity	Low

Test Case ID	FDC-019
Title	FileDiskCollection is Not an Interface
Objective	Ensure FileDiskCollection is a class, not an interface.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass and check isInterface().
Test Data	- Class: FileDiskCollection
Expected Result	- isInterface returns false.
Actual Result	Class is not an interface.
Status	Pass
Severity	Low

Test Case ID	FDC-020
Title	FileDiskCollection is Not a Trait
Objective	Ensure FileDiskCollection is a class, not a trait.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass and check isTrait().
Test Data	- Class: FileDiskCollection
Expected Result	- isTrait returns false.
Actual Result	Class is not a trait.
Status	Pass
Severity	Low

Test Case ID	FDC-021
Title	FileDiskCollection Class is Loaded
Objective	Verify that FileDiskCollection class exists/is loaded.
Preconditions	- Application running
Test Steps	1. Use class_exists to verify class availability.
Test Data	- Class: FileDiskCollection::class
Expected Result	- class_exists returns true.
Actual Result	Class is loaded as expected.
Status	Pass
Severity	Low

Test Case ID	FDC-022
Title	FileDiskCollection Uses ResourceCollection Import
Objective	Ensure FileDiskCollection file imports ResourceCollection correctly.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get file content. 2. Check for 'use Illuminate\Http\Resources\Json\ResourceCollection' import.
Test Data	- File content
Expected Result	- Import statement present.
Actual Result	Import statement found.

Status	Pass
Severity	Low

Test Case ID	FDC-023
Title	toArray Method is Not Static
Objective	Verify that toArray is an instance method and not static.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to examine toArray. 2. Check isStatic().
Test Data	- Method: toArray
Expected Result	- isStatic returns false.
Actual Result	Method is not static.
Status	Pass
Severity	Low

Test Case ID	FDC-024
Title	toArray Method is Not Abstract
Objective	Verify toArray is implemented (not abstract).
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to inspect toArray method. 2. Check isAbstract().
Test Data	- Method: toArray
Expected Result	- isAbstract returns false.
Actual Result	Method is implemented.
Status	Pass
Severity	Low

Test Case ID	FDC-025
Title	Preserves File Disk Data Integrity
Objective	Ensure toArray preserves disk id and name accurately through transformation.
Preconditions	- Application running
Test Steps	1. Create dummy file disk (id=42, name='Test Disk'). 2. Wrap in FileDiskResource and FileDiskCollection. 3. Call toArray and inspect output.
Test Data	- Input: {id:42, name:'Test Disk'}
Expected Result	- Output[0]['id'] == 42 - Output[0]['name'] == 'Test Disk'
Actual Result	Data integrity maintained as expected.
Status	Pass
Severity	High

Test Case ID	FDC-026
Title	Handles Different Disk Names
Objective	Verify toArray correctly transforms disks with various names.
Preconditions	- Application running

Test Steps	1. Create three dummy disks with names: 'S3 Storage', 'Local Storage', 'Dropbox Storage'. 2. Wrap each in FileDiskResource. 3. Place all in FileDiskCollection and call toArray.
Test Data	- Input names: 'S3 Storage', 'Local Storage', 'Dropbox Storage'
Expected Result	- Output[0]['name'] == 'S3 Storage' - Output[1]['name'] == 'Local Storage' - Output[2]['name'] == 'Dropbox Storage'
Actual Result	Names match expected output.
Status	Pass
Severity	Medium

Test Case ID	FDC-027
Title	Handles Different Drivers
Objective	Ensure disks with different drivers are correctly transformed.
Preconditions	- Application running
Test Steps	1. Create two disks with driver 's3' and 'local'. 2. Wrap each in FileDiskResource. 3. Add to FileDiskCollection and convert using toArray.
Test Data	- Input drivers: 's3', 'local'
Expected Result	- Output[0]['driver'] == 's3' - Output[1]['driver'] == 'local'
Actual Result	Drivers are correctly transformed and present.
Status	Pass
Severity	Medium

Test Case ID	FDC-028
Title	FileDiskCollection File Size Constraint
Objective	Ensure source file for FileDiskCollection is concise (<1000 bytes).
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get file content. 2. Measure file length in bytes.
Test Data	- File content
Expected Result	- File size is less than 1000 bytes.
Actual Result	File size meets requirement.
Status	Pass
Severity	Low

Test Case ID	FDC-029
Title	FileDiskCollection Line Count Constraint
Objective	Ensure source file for FileDiskCollection is concise (<30 lines).
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get file content. 2. Count number of lines.
Test Data	- File content
Expected Result	- Line count is less than 30.
Actual Result	File length constraint satisfied.
Status	Pass

Severity	Low

Test Case ID	FDC-030
Title	Transformed Items Include All Disk Fields
Objective	Verify all necessary disk fields are present after transformation.
Preconditions	- Application running
Test Steps	1. Create dummy disk with id, name, driver. 2. Wrap in FileDiskResource and transform. 3. Check for presence of keys: id, name, driver.
Test Data	- Disk: {id:1, name:'S3 Disk', driver:'s3'}
Expected Result	- Output[0] keys: 'id', 'name', 'driver'
Actual Result	Keys present in output.
Status	Pass
Severity	High

Test Case ID	FDC-031
Title	Result of toArray is a Valid Array Structure
Objective	Validate that toArray always returns an array.
Preconditions	- Application running
Test Steps	1. Create dummy disk and wrap in FileDiskResource. 2. Transform using toArray. 3. Check output type.
Test Data	- Disk: {id:1, ...}
Expected Result	- Output is of type array; is_array returns true.
Actual Result	Result is array as expected.
Status	Pass
Severity	High

Test Case ID	FDC-032
Title	toArray Method has Documentation Block
Objective	Confirm that toArray method has a doc comment.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to obtain doc comment of toArray.
Test Data	- Method: toArray
Expected Result	- getDocComment does not return false.
Actual Result	Documentation block exists for method.
Status	Pass
Severity	Low

Test Case ID	FDC-033
Title	toArray Method has Return Type Documentation
Objective	Verify that the toArray docblock contains @return annotation.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to get doc comment from toArray. 2. Check for '@return' presence.
Test Data	- Method docblock

Expected Result	- '@return' found in documentation.
Actual Result	@return tag present in comment.
Status	Pass
Severity	Low

Test Case ID	FDC-034
Title	toArray Method has Param Documentation
Objective	Ensure toArray method docblock contains @param annotation.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to obtain doc comment of toArray. 2. Check for '@param' in comment.
Test Data	- Method docblock
Expected Result	- '@param' found in documentation.
Actual Result	@param tag present in comment.
Status	Pass
Severity	Low

Test Case ID	FDC-035
Title	Handles Disks with Different set_as_default States
Objective	Verify collection transformation handles disks with varied set_as_default attributes.
Preconditions	- Application running
Test Steps	1. Create two dummy disks: one with set_as_default=true, one with set_as_default=false. 2. Wrap both in FileDiskResource. 3. Place in FileDiskCollection and convert with toArray. 4. Check count of output array.
Test Data	- Disk1: {id:1, name:'Default Disk', set_as_default:true} - Disk2: {id:2, name:'Non-Default Disk', set_as_default:false}
Expected Result	- Output contains both disks accurately.
Actual Result	Both entries processed and included.
Status	Pass
Severity	High

File: FileDiskResource-Test.txt

Test Case ID	FDR-001
Title	toArray returns correct data with all properties populated
Objective	Verify that FileDiskResource::toArray returns an array with all properties correctly mapped from a fully populated model.
Preconditions	- Application is running - Crater FileDiskResource class is available - Database seeded with test data if applicable - Mockery mocking library available
Test Steps	1. Create a mock model object with all properties populated: - id = 1 - name = 'Local Disk' - type = 'local' - driver = 'local' - set_as_default = true - credentials = ['path' => '/app/public'] - company_id = 101 2. Instantiate FileDiskResource with the mock model. 3. Create a mock Request object. 4. Call the toArray method on the FileDiskResource with the mock request. 5. Verify the returned array matches all input model properties exactly.

Test Data	- Input: {
	id: 1,
	name: 'Local Disk',
	type: 'local',
	driver: 'local',
	set_as_default: true,
	credentials: ['path' => '/app/public'],
	company_id: 101
	}
	- Expected Output: Array with exactly the same values as input.

Expected Result	The returned array from toArray contains all provided values matching the source model properties.
Actual Result	Returned array matches expected values exactly.
Status	Pass
Severity	High

Test Case ID	FDR-002
Title	toArray handles null properties gracefully
Objective	Verify that FileDiskResource::toArray correctly processes and returns null values for properties set to null in the source model.
Preconditions	- Application is running - Crater FileDiskResource class is available - Mockery mocking library available
Test Steps	1. Create a mock model object with some properties explicitly set to null: - id = 2 - name = null - type = 's3' - driver = 's3' - set_as_default = false - credentials = null - company_id = null 2. Instantiate FileDiskResource with the mock model. 3. Create a mock Request object. 4. Call the toArray method on the FileDiskResource with the mock request. 5. Verify the returned array includes null for the properties that were null.

Test Data	- Input: {
	id: 2,
	name: null,
	type: 's3',
	driver: 's3',
	set_as_default: false,
	credentials: null,
	company_id: null
	}
	- Expected Output: Array with matching null values for those properties.

Expected Result	The returned array includes null for 'name', 'credentials', and 'company_id', exactly as input model.
Actual Result	Returned array has null where expected and matches model values.
Status	Pass
Severity	Medium

Test Case ID	FDR-003
Title	toArray handles empty strings, zero, and empty array values
Objective	Verify that FileDiskResource::toArray correctly processes empty strings, zero values, empty arrays, and boolean false.
Preconditions	- Application is running - Crater FileDiskResource class is available - Mockery mocking library available
Test Steps	1. Create a mock model object with properties as empty, zero, or default values: - id = 3 - name = '' (empty string) - type = '' (empty string) - driver = '' (empty string) - set_as_default = false - credentials = [] (empty array) - company_id = 0 2. Instantiate FileDiskResource with the mock model. 3. Create a mock Request object. 4. Call the toArray method on the FileDiskResource with the mock request. 5. Verify the returned array contains empty string, zero, and empty array values correctly.

Test Data	- Input: {
	id: 3,
	name: '',
	type: '',
	driver: '',
	set_as_default: false,
	credentials: [],
	company_id: 0
	}
	- Expected Output: Array reflecting exactly the same empty/default values.

Expected Result	Returned array correctly includes empty strings, empty array for credentials, boolean false, and 0 for company_id.
Actual Result	Returned array matches expected values for all empty/default fields.
Status	Pass
Severity	Medium

Test Case ID	FDR-004
Title	toArray ensures all expected keys are present even if properties are undefined on source object

Objective	Verify that FileDiskResource::toArray always includes all expected keys even when some properties are missing in the source object.
Preconditions	- Application is running - Crater FileDiskResource class is available - Proxy object used to simulate undefined properties - Mockery mocking library available
Test Steps	1. Create a mock model object with only 'id' property defined (id = 4); other properties omitted. 2. Wrap the mock model in a proxy object that returns null for undefined properties. 3. Instantiate FileDiskResource with the proxy model. 4. Create a mock Request object. 5. Call the toArray method on the FileDiskResource with the mock request. 6. Verify that the returned array contains keys: 'id', 'name', 'type', 'driver', 'set_as_default', 'credentials', 'company_id', with null values for missing properties.

Test Data	- Input: {
	id: 4,
	// Other properties intentionally undefined
	}
	- Expected Output: {
	'id': 4,
	'name': null,
	'type': null,
	'driver': null,
	'set_as_default': null,
	'credentials': null,
	'company_id': null
	}

Expected Result	All expected keys are present in returned array, with null values for any properties not defined in the source object.
Actual Result	Returned array includes all expected keys; undefined properties resolve to null values as expected.
Status	Pass
Severity	Medium

File: FilePermissionChecker-Test.txt

Test Case ID	FPC-001
Title	Instantiate FilePermissionChecker
Objective	Verify that the FilePermissionChecker class can be instantiated.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate the FilePermissionChecker class. 2. Check the instance type against FilePermissionChecker.
Test Data	- None
Expected Result	- The created object is an instance of FilePermissionChecker.
Actual Result	The created object was correctly identified as an instance of FilePermissionChecker.
Status	Pass
Severity	Medium

Test Case ID	FPC-002
Title	FilePermissionChecker Namespace Verification
Objective	Confirm that FilePermissionChecker is in the "Crater\Space" namespace.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass to inspect FilePermissionChecker. 2. Retrieve the namespace name.
Test Data	- None
Expected Result	- Namespace name is "Crater\Space".
Actual Result	Namespace name returned as "Crater\Space".
Status	Pass
Severity	Low

Test Case ID	FPC-003
Title	FilePermissionChecker is Not Abstract
Objective	Ensure FilePermissionChecker is a concrete class, not abstract.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass on FilePermissionChecker. 2. Check isAbstract() property.
Test Data	- None
Expected Result	- isAbstract() is false.
Actual Result	isAbstract() returned false.
Status	Pass
Severity	Low

Test Case ID	FPC-004
Title	FilePermissionChecker is Instantiable
Objective	Verify FilePermissionChecker class is instantiable.
Preconditions	- Application running - FilePermissionChecker class loaded

Test Steps	1. Use ReflectionClass for FilePermissionChecker. 2. Call isInstantiable().
Test Data	- None
Expected Result	- isInstantiable() is true.
Actual Result	isInstantiable() returned true.
Status	Pass
Severity	Medium

Test Case ID	FPC-005
Title	Constructor Initializes Empty Permissions in Results
Objective	Validate that the permissions array in the results property is initialized as empty on instantiation.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Access "results" property. 3. Check that "permissions" key exists, is an array, and is empty.
Test Data	- None
Expected Result	- results['permissions'] is an empty array.
Actual Result	results['permissions'] was an empty array upon instantiation.
Status	Pass
Severity	Medium

Test Case ID	FPC-006
Title	Constructor Initializes Null Errors in Results
Objective	Validate that the errors key in results is set to null after instantiating FilePermissionChecker.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Access "results" property. 3. Check that "errors" key exists and is null.
Test Data	- None
Expected Result	- results['errors'] is null.
Actual Result	results['errors'] was null as expected.
Status	Pass
Severity	Low

Test Case ID	FPC-007
Title	FilePermissionChecker Has "check" Method
Objective	Verify the existence of the public "check" method in FilePermissionChecker.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Use method_exists to check for "check" method.
Test Data	- None
Expected Result	- "check" method exists and is callable.
Actual Result	"check" method found.

Status	Pass
Severity	High

Test Case ID	FPC-008
Title	FilePermissionChecker Has "getPermission" Method
Objective	Ensure the "getPermission" method is defined in FilePermissionChecker.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass for FilePermissionChecker. 2. Check if "getPermission" method exists.
Test Data	- None
Expected Result	- "getPermission" method exists.
Actual Result	"getPermission" method confirmed.
Status	Pass
Severity	High

Test Case ID	FPC-009
Title	FilePermissionChecker Has "addFile" Method
Objective	Validate that "addFile" method exists in FilePermissionChecker.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass for FilePermissionChecker. 2. Check for "addFile" method.
Test Data	- None
Expected Result	- "addFile" method exists.
Actual Result	"addFile" method confirmed.
Status	Pass
Severity	High

Test Case ID	FPC-010
Title	FilePermissionChecker Has "addFileAndSetErrors" Method
Objective	Verify the existence of the "addFileAndSetErrors" method.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass for FilePermissionChecker. 2. Check for "addFileAndSetErrors" method.
Test Data	- None
Expected Result	- "addFileAndSetErrors" method exists.
Actual Result	"addFileAndSetErrors" method confirmed.
Status	Pass
Severity	High

Test Case ID	FPC-011
Title	"check" Method is Public
Objective	Verify that the "check" method has public visibility.
Preconditions	- Application running - FilePermissionChecker class loaded

Test Steps	1. Use ReflectionClass on FilePermissionChecker. 2. Get "check" method and check its visibility.
Test Data	- None
Expected Result	- "check" method is public.
Actual Result	"check" method found to be public.
Status	Pass
Severity	High

Test Case ID	FPC-012
Title	"getPermission" Method is Private
Objective	Confirm that the "getPermission" method is private.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass on FilePermissionChecker. 2. Get "getPermission" method and check its visibility.
Test Data	- None
Expected Result	- "getPermission" method is private.
Actual Result	"getPermission" found to be private.
Status	Pass
Severity	High

Test Case ID	FPC-013
Title	"addFile" Method is Private
Objective	Confirm the "addFile" method is private.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass on FilePermissionChecker. 2. Get "addFile" method and check its visibility.
Test Data	- None
Expected Result	- "addFile" method is private.
Actual Result	"addFile" found to be private.
Status	Pass
Severity	High

Test Case ID	FPC-014
Title	"addFileAndSetErrors" Method is Private
Objective	Confirm that "addFileAndSetErrors" is private.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass to get "addFileAndSetErrors" method. 2. Check its visibility.
Test Data	- None
Expected Result	- "addFileAndSetErrors" method is private.
Actual Result	"addFileAndSetErrors" found to be private.
Status	Pass
Severity	High

Test Case ID	FPC-015
Title	"check" Method Handles Empty Folders Array
Objective	Verify that the "check" method deals gracefully with an empty folders array.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Call "check" with an empty array. 3. Verify returned results array structure.
Test Data	- Input: []
Expected Result	- Returned results is an array. - results['errors'] is null. - results['permissions'] is empty.
Actual Result	All expectations met: results was as specified.
Status	Pass
Severity	High

Test Case ID	FPC-016
Title	"check" Returns Array With Permissions and Errors Keys
Objective	Ensure that the result of "check()" includes both "permissions" and "errors" keys.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Call "check" with an empty array. 3. Verify that "permissions" and "errors" keys exist in the returned array.
Test Data	- Input: []
Expected Result	- Result has keys "permissions" and "errors".
Actual Result	Both keys were present in the output.
Status	Pass
Severity	High

Test Case ID	FPC-017
Title	"addFile" Adds Entry with isSet True
Objective	Verify that calling "addFile" adds a permission entry with isSet=true.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Use Reflection to invoke "addFile" with folder="test_folder", permission="0755", isSet=true. 3. Access "results" property and check added entry.
Test Data	- folder: "test_folder" - permission: "0755" - isSet: true
Expected Result	- permissions array contains one entry: folder="test_folder", permission="0755", isSet=true.
Actual Result	Entry added correctly to permissions array.
Status	Pass
Severity	High

Test Case ID	FPC-018

Title	"addFile" Adds Entry with isSet False
Objective	Verify that "addFile" adds a permission entry with isSet=false.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Use Reflection to invoke "addFile" with folder="test_folder", permission="0644", isSet=false. 3. Access "results" property and check added entry.
Test Data	- folder: "test_folder" - permission: "0644" - isSet: false
Expected Result	- permissions array contains one entry: folder="test_folder", permission="0644", isSet=false.
Actual Result	Entry added as expected with isSet=false.
Status	Pass
Severity	High

Test Case ID	FPC-019
Title	"addFile" Adds Multiple Entries Correctly
Objective	Ensure multiple calls to "addFile" add distinct entries for each folder.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Use Reflection to invoke "addFile" three times: a. ("folder_a", "0755", true) b. ("folder_b", "0644", false) c. ("folder_c", "0777", true) 3. Access "results" and confirm each entry.
Test Data	- folder_a: "0755", true - folder_b: "0644", false - folder_c: "0777", true
Expected Result	- permissions array contains three entries matching the input order and values.
Actual Result	All three entries appeared in correct order and with correct data.
Status	Pass
Severity	High

Test Case ID	FPC-020
Title	"addFileAndSetErrors" Adds File Entry
Objective	Verify that "addFileAndSetErrors" adds the specified entry to permissions.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Use Reflection to call "addFileAndSetErrors" with folder="error_folder", permission="0777", isSet=false. 3. Access "results" to confirm entry.
Test Data	- folder: "error_folder" - permission: "0777" - isSet: false
Expected Result	- permissions array contains one entry with specified data.
Actual Result	Entry added to permissions as specified.
Status	Pass

Severity	High

Test Case ID	FPC-021
Title	"addFileAndSetErrors" Sets Errors to True
Objective	Confirm that "addFileAndSetErrors" sets the "errors" property to true.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Access "results" and verify "errors" is initially null. 3. Call "addFileAndSetErrors". 4. Check that "errors" is now true.
Test Data	- folder: "error_folder" - permission: "0777" - isSet: false
Expected Result	- errors property changes from null to true.
Actual Result	errors property set to true after operation.
Status	Pass
Severity	High

Test Case ID	FPC-022
Title	"addFileAndSetErrors" Maintains Errors True After Multiple Calls
Objective	Verify that repeated calls to "addFileAndSetErrors" keep "errors" property as true.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate FilePermissionChecker. 2. Call "addFileAndSetErrors" with folder_1. 3. Call "addFileAndSetErrors" with folder_2. 4. Check "results->errors" and "permissions" count.
Test Data	- folder_1: "0777", false - folder_2: "0755", false
Expected Result	- permissions array has two entries. - errors property remains true.
Actual Result	Both permissions added; errors property stayed true.
Status	Pass
Severity	High

Test Case ID	FPC-023
Title	Multiple FilePermissionChecker Instances
Objective	Ensure that multiple distinct instances of FilePermissionChecker can be created.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Create two FilePermissionChecker instances. 2. Verify both are instances of FilePermissionChecker and not the same object.
Test Data	- None
Expected Result	- Both objects are FilePermissionChecker instances and are distinct.
Actual Result	Instances created and verified to be unique.
Status	Pass

Severity	Medium

Test Case ID	FPC-024
Title	FilePermissionChecker Can Be Cloned
Objective	Ensure FilePermissionChecker objects can be cloned.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Create a FilePermissionChecker instance. 2. Clone the instance. 3. Verify the clone is a distinct instance of FilePermissionChecker.
Test Data	- None
Expected Result	- Cloned object is instance of FilePermissionChecker, but not same as original.
Actual Result	Clone verified as distinct instance.
Status	Pass
Severity	Medium

Test Case ID	FPC-025
Title	FilePermissionChecker Usable in Type Hints
Objective	Verify that FilePermissionChecker is compatible with type hints.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Define a function that type hints FilePermissionChecker. 2. Pass an instance of FilePermissionChecker. 3. Return and compare the instance.
Test Data	- None
Expected Result	- Passed instance returned correctly.
Actual Result	Type hint worked as expected.
Status	Pass
Severity	Medium

Test Case ID	FPC-026
Title	FilePermissionChecker is Not Final
Objective	Confirm the class is not declared as final.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass for FilePermissionChecker. 2. Check isFinal() property.
Test Data	- None
Expected Result	- isFinal() returns false.
Actual Result	isFinal() was false.
Status	Pass
Severity	Low

Test Case ID	FPC-027
Title	FilePermissionChecker is Not an Interface
Objective	Confirm FilePermissionChecker is a class, not an interface.

Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass for FilePermissionChecker. 2. Check isInterface() property.
Test Data	- None
Expected Result	- isInterface() returns false.
Actual Result	isInterface() was false.
Status	Pass
Severity	Low

Test Case ID	FPC-028
Title	FilePermissionChecker is Not a Trait
Objective	Confirm FilePermissionChecker is not implemented as a trait.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass for FilePermissionChecker. 2. Check isTrait() property.
Test Data	- None
Expected Result	- isTrait() returns false.
Actual Result	isTrait() was false.
Status	Pass
Severity	Low

Test Case ID	FPC-029
Title	FilePermissionChecker Class Is Loaded
Objective	Validate that FilePermissionChecker is loaded in runtime.
Preconditions	- Application running
Test Steps	1. Use class_exists for FilePermissionChecker.
Test Data	- None
Expected Result	- class_exists(FilePermissionChecker::class) returns true.
Actual Result	class_exists returned true.
Status	Pass
Severity	High

Test Case ID	FPC-030
Title	FilePermissionChecker Has Results Property
Objective	Verify the existence of a "results" property.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass for FilePermissionChecker. 2. Check for existence of "results" property.
Test Data	- None
Expected Result	- "results" property exists.
Actual Result	"results" property found.
Status	Pass
Severity	High

Test Case ID	FPC-031
Title	"results" Property is Protected
Objective	Verify that the "results" property has protected visibility.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use ReflectionClass to get "results" property. 2. Check visibility.
Test Data	- None
Expected Result	- "results" property is protected.
Actual Result	Property was protected.
Status	Pass
Severity	Medium

Test Case ID	FPC-032
Title	FilePermissionChecker File Has Expected Structure
Objective	Confirm that the source file contains key elements of FilePermissionChecker.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use Reflection to locate FilePermissionChecker file. 2. Read file contents. 3. Check for declarations: class FilePermissionChecker, protected $results, public constructor and check method.
Test Data	- None
Expected Result	- File contains "class FilePermissionChecker". - Contains "protected $results". - Contains "public function __construct()". - Contains "public function check(array $folders)".
Actual Result	File structure validated for all components.
Status	Pass
Severity	Medium

Test Case ID	FPC-033
Title	FilePermissionChecker File Has Reasonable Line Count
Objective	Validate that code file line count is within acceptable range.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Read FilePermissionChecker source code. 2. Count lines. 3. Check line count is >50 and <150.
Test Data	- None
Expected Result	- Line count between 51 and 149.
Actual Result	Line count was within the specified range.
Status	Pass
Severity	Low

Test Case ID	FPC-034
Title	"check" Method Uses Foreach Loop

Objective	Ensure permission checks iterate through folders using foreach.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Read FilePermissionChecker source code. 2. Search for "foreach ($folders as $folder => $permission)".
Test Data	- None
Expected Result	- "foreach ($folders as $folder => $permission)" is present in code.
Actual Result	foreach detected.
Status	Pass
Severity	High

Test Case ID	FPC-035
Title	"check" Method Calls "getPermission"
Objective	Ensure the "check" method calls "getPermission" for each folder.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Read FilePermissionChecker source code. 2. Search for "$this->getPermission($folder)" within "check" method.
Test Data	- None
Expected Result	- "$this->getPermission($folder)" is invoked.
Actual Result	getPermission call present in check.
Status	Pass
Severity	High

Test Case ID	FPC-036
Title	"check" Calls "addFileAndSetErrors" for Insufficient Permission
Objective	Ensure that "check" calls "addFileAndSetErrors" when permissions are not met.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Read FilePermissionChecker source code. 2. Locate call of "$this->addFileAndSetErrors" in check method.
Test Data	- None
Expected Result	- "$this->addFileAndSetErrors" called when permission requirement fails.
Actual Result	Call found as expected.
Status	Pass
Severity	High

Test Case ID	FPC-037
Title	"check" Calls "addFile" When Permission is Met
Objective	Ensure that "addFile" is called by "check" with isSet=true on correct permission.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Read FilePermissionChecker source code. 2. Locate call of "$this->addFile($folder, $permission, true)" in check method.
Test Data	- None
Expected Result	- "$this->addFile($folder, $permission, true)" is present.

Actual Result	addFile correctly called in successful permission flow.
Status	Pass
Severity	High

Test Case ID	FPC-038
Title	"check" Method Returns Results
Objective	Verify the "check" method returns the results property.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Read FilePermissionChecker source code. 2. Check for "return $this->results" statement in "check".
Test Data	- None
Expected Result	- "return $this->results" is present at the end of "check".
Actual Result	Correct return statement present.
Status	Pass
Severity	High

Test Case ID	FPC-039
Title	"getPermission" Uses "fileperms" Function
Objective	Verify "getPermission" uses fileperms to retrieve folder permission.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Read source of "getPermission". 2. Confirm use of "fileperms" function.
Test Data	- None
Expected Result	- "fileperms" used within "getPermission".
Actual Result	"fileperms" usage confirmed.
Status	Pass
Severity	High

Test Case ID	FPC-040
Title	"getPermission" Uses "base_path" Function
Objective	Verify that "base_path" is used in "getPermission" to resolve folder path.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Read "getPermission" implementation. 2. Check for usage of base_path($folder).
Test Data	- None
Expected Result	- "base_path($folder)" present in method.
Actual Result	base_path used as expected.
Status	Pass
Severity	Medium

Test Case ID	FPC-041
Title	"getPermission" Uses sprintf with %o Format
Objective	Ensure that permissions are formatted in octal via sprintf("%o").

Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Inspect "getPermission" code. 2. Confirm presence of sprintf('%o', ...).
Test Data	- None
Expected Result	- sprintf called with '%o' format.
Actual Result	sprintf with '%o' format seen in code.
Status	Pass
Severity	High

Test Case ID	FPC-042
Title	"getPermission" Uses substr to Get Last 4 Characters
Objective	Confirm last 4 permission digits are obtained by substr.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Inspect "getPermission" code. 2. Look for usage of substr to extract last 4 characters.
Test Data	- None
Expected Result	- substr used with '-4' to obtain permission.
Actual Result	Substring operation verified.
Status	Pass
Severity	Medium

Test Case ID	FPC-043
Title	"addFile" Uses array_push on Permissions
Objective	Ensure "addFile" uses array_push to append permissions.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Review addFile method source code. 2. Look for array_push($this->results['permissions'], ...).
Test Data	- None
Expected Result	- array_push statement present in addFile.
Actual Result	array_push usage validated.
Status	Pass
Severity	High

Test Case ID	FPC-044
Title	"addFileAndSetErrors" Calls "addFile"
Objective	Ensure "addFileAndSetErrors" internally calls "addFile".
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Inspect addFileAndSetErrors source. 2. Confirm call to $this->addFile($folder, $permission, $isSet).
Test Data	- None
Expected Result	- Call to "addFile" is made inside "addFileAndSetErrors".
Actual Result	Confirmed that addFile is called.
Status	Pass

Severity	High

Test Case ID	FPC-045
Title	"addFileAndSetErrors" Implementation Sets Errors Property to True
Objective	Verify "addFileAndSetErrors" sets results['errors'] = true.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Inspect addFileAndSetErrors method source code. 2. Confirm assignment of $this->results['errors'] = true.
Test Data	- None
Expected Result	- "results['errors']" is set to true in method.
Actual Result	Error flag assigned as true.
Status	Pass
Severity	High

Test Case ID	FPC-046
Title	Constructor Has Documentation
Objective	Ensure that FilePermissionChecker's constructor has a doc comment.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use Reflection to access constructor. 2. Check for doc comment.
Test Data	- None
Expected Result	- getDocComment() for constructor does not return false.
Actual Result	Constructor documentation present.
Status	Pass
Severity	Low

Test Case ID	FPC-047
Title	"check" Method Has Documentation
Objective	Ensure doc comment present for "check" method.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use Reflection to get "check" method. 2. Check for doc comment.
Test Data	- None
Expected Result	- "check" method has a PHP documentation block.
Actual Result	Documentation found for "check".
Status	Pass
Severity	High

Test Case ID	FPC-048
Title	"getPermission" Method Has Documentation
Objective	Confirm presence of documentation on "getPermission".
Preconditions	- Application running - FilePermissionChecker class loaded

Test Steps	1. Use Reflection for "getPermission". 2. Check getDocComment().
Test Data	- None
Expected Result	- "getPermission" has doc comment.
Actual Result	Doc comment present.
Status	Pass
Severity	High

Test Case ID	FPC-049
Title	"addFile" Method Has Documentation
Objective	Check for documentation block on "addFile".
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use Reflection for "addFile". 2. Check getDocComment().
Test Data	- None
Expected Result	- "addFile" has a doc block.
Actual Result	Documentation was present.
Status	Pass
Severity	High

Test Case ID	FPC-050
Title	"addFileAndSetErrors" Method Has Documentation
Objective	Confirm "addFileAndSetErrors" is documented.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use Reflection for "addFileAndSetErrors". 2. Check getDocComment().
Test Data	- None
Expected Result	- Documentation exists.
Actual Result	Doc block found.
Status	Pass
Severity	High

Test Case ID	FPC-051
Title	"check" Method Accepts Array Parameter
Objective	Ensure "check" method parameters are correct.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use Reflection to inspect "check". 2. Verify single parameter named "folders", has array type.
Test Data	- None
Expected Result	- Single parameter "folders" of type array.
Actual Result	Parameter verified as correct.
Status	Pass
Severity	High

Test Case ID	FPC-052
Title	"getPermission" Method Accepts Folder Parameter
Objective	Check "getPermission" accepts correct parameter.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Reflect on "getPermission". 2. Verify parameter count and names.
Test Data	- None
Expected Result	- One parameter named "folder".
Actual Result	Parameter present as specified.
Status	Pass
Severity	High

Test Case ID	FPC-053
Title	"addFile" Method Accepts Three Parameters
Objective	Verify "addFile" parameter structure.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use Reflection for "addFile". 2. Check for three parameters: folder, permission, isSet.
Test Data	- None
Expected Result	- Three parameters named correctly.
Actual Result	Parameter names matched as required.
Status	Pass
Severity	High

Test Case ID	FPC-054
Title	"addFileAndSetErrors" Method Accepts Three Parameters
Objective	Ensure "addFileAndSetErrors" method parameters are correct.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Use Reflection for "addFileAndSetErrors". 2. Check for folder, permission, isSet.
Test Data	- None
Expected Result	- Three parameters exist, named correctly.
Actual Result	Correct parameters found.
Status	Pass
Severity	High

Test Case ID	FPC-055
Title	Different Instances Have Independent Results
Objective	Confirm that data added to one instance's results does not impact another.
Preconditions	- Application running - FilePermissionChecker class loaded
Test Steps	1. Instantiate two FilePermissionChecker objects. 2. Add different permission entries to each. 3. Access results from each and compare.

Test Data	- checker1: add "folder_1", "0755", true - checker2: add "folder_2", "0644", false
Expected Result	- checker1's results contain only folder_1. - checker2's results contain only folder_2. - Results arrays are independent.
Actual Result	Results were independent for both instances.
Status	Pass
Severity	High

File: FilesystemDisks-Test.txt

Test Case ID	FD-001
Title	Instantiation of FilesystemDisks Rule
Objective	Verify that the FilesystemDisks rule can be instantiated successfully.
Preconditions	- Application running - Crater package installed - Necessary class autoloading is available
Test Steps	1. Attempt to create a new instance of FilesystemDisks. 2. Check if the created instance is of FilesystemDisks class.
Test Data	- Class: FilesystemDisks
Expected Result	- A new instance of FilesystemDisks is created successfully. - The instance is of type FilesystemDisks.
Actual Result	A new FilesystemDisks object is instantiated and verified as the correct class.
Status	Pass
Severity	Low

Test Case ID	FD-002
Title	Passes Returns True for Configured Disks
Objective	Validate that the passes() method returns true when provided disk names exist in the configuration.
Preconditions	- Application running - Database seeded if required - Config mock returns ['local', 's3', 'my_custom_disk'] for 'filesystem.disks'
Test Steps	1. Mock Config to return ['local', 's3', 'my_custom_disk'] for 'filesystem.disks'. 2. Create a new FilesystemDisks instance. 3. Call passes('disk', 'local'). 4. Call passes('disk', 's3'). 5. Call passes('disk', 'my_custom_disk').
Test Data	- Configured disks: ['local', 's3', 'my_custom_disk'] - Input disks tested: 'local', 's3', 'my_custom_disk'
Expected Result	- passes('disk', 'local') returns true. - passes('disk', 's3') returns true. - passes('disk', 'my_custom_disk') returns true.
Actual Result	All passes() calls return true for each configured disk.
Status	Pass
Severity	High

Test Case ID	FD-003
Title	Passes Returns False for Not Configured Disks
Objective	Ensure passes() returns false for disks not in the configuration.
Preconditions	- Application running - Database seeded if required - Config mock returns ['local', 's3'] for 'filesystem.disks'
Test Steps	1. Mock Config to return ['local', 's3'] for 'filesystem.disks'. 2. Create a new FilesystemDisks instance. 3. Call passes('disk', 'non_existent_disk'). 4. Call passes('disk', 'ftp'). 5. Call passes('disk', 'my_custom_disk').
Test Data	- Configured disks: ['local', 's3'] - Input disks tested: 'non_existent_disk', 'ftp', 'my_custom_disk'

Expected Result	- passes('disk', 'non_existent_disk') returns false. - passes('disk', 'ftp') returns false. - passes('disk', 'my_custom_disk') returns false.
Actual Result	All passes() calls return false for unconfigured disks.
Status	Pass
Severity	High

Test Case ID	FD-004
Title	Passes Returns False when Configured Disks is Empty
Objective	Validate passes() returns false when the disk array is empty.
Preconditions	- Application running - Database seeded if required - Config mock returns [] (empty array) for 'filesystem.disks'
Test Steps	1. Mock Config to return an empty array for 'filesystem.disks'. 2. Create a new FilesystemDisks instance. 3. Call passes('disk', 'local'). 4. Call passes('disk', 's3'). 5. Call passes('disk', 'any_disk').
Test Data	- Configured disks: [] - Input disks tested: 'local', 's3', 'any_disk'
Expected Result	- passes('disk', 'local') returns false. - passes('disk', 's3') returns false. - passes('disk', 'any_disk') returns false.
Actual Result	All passes() calls return false when the configured disks list is empty.
Status	Pass
Severity	High

Test Case ID	FD-005
Title	Passes Throws TypeError on Null Configuration
Objective	Validate that passes() throws a TypeError if configured disks is null, indicating misconfiguration.
Preconditions	- Application running - Database seeded if required - Config mock returns null for 'filesystem.disks'
Test Steps	1. Mock Config to return null for 'filesystem.disks'. 2. Create a FilesystemDisks instance. 3. Call passes('disk', 'local') and expect a TypeError.
Test Data	- Configured disks: null - Input disk: 'local'
Expected Result	- Calling passes('disk', 'local') raises a TypeError exception.
Actual Result	A TypeError is thrown when in_array is called with null as haystack.
Status	Pass
Severity	High

Test Case ID	FD-006
Title	Message Returns Correct Validation Error
Objective	Verify that the message() method returns the intended validation error string.
Preconditions	- Application running
Test Steps	1. Create a new FilesystemDisks instance. 2. Call the message() method.

Test Data	- Expected message: "This disk is not configured as a filesystem disk."
Expected Result	- message() returns "This disk is not configured as a filesystem disk."
Actual Result	The method returns the correct validation message string.
Status	Pass
Severity	Medium

File: FinishController-Test.txt

Test Case ID	FC-001
Title	Successful creation of database_created file returns a success response
Objective	Verify that when the database_created file is successfully written to storage, the controller returns a success response.
Preconditions	- Application running - Database seeded - Storage facade properly mocked - FinishController is accessible
Test Steps	1. Mock the Storage facade to use the 'local' disk. 2. Mock the 'put' method of the disk to successfully create the 'database_created' file (simulate return value true). 3. Instantiate the FinishController. 4. Create a new HTTP Request object. 5. Invoke the controller's __invoke method with the request. 6. Capture the JsonResponse returned.
Test Data	- Disk: 'local' - File Name: 'database_created' - File Content: 'database_created' - Storage::put return value: true
Expected Result	1. The response should be an instance of JsonResponse. 2. The response data should exactly match ['success' => true].
Actual Result	Both assertions pass: response is JsonResponse, and data is ['success' => true].
Status	Pass
Severity	High

Test Case ID	FC-002
Title	Failing to create database_created file still returns a success response (current implementation)
Objective	Verify that even if the database_created file is not created (Storage::put returns false), the controller still returns a success response.
Preconditions	- Application running - Database seeded - Storage facade properly mocked - FinishController is accessible
Test Steps	1. Mock the Storage facade to use the 'local' disk. 2. Mock the 'put' method of the disk to fail creating the 'database_created' file (simulate return value false). 3. Instantiate the FinishController. 4. Create a new HTTP Request object. 5. Invoke the controller's __invoke method with the request. 6. Capture the JsonResponse returned.
Test Data	- Disk: 'local' - File Name: 'database_created' - File Content: 'database_created' - Storage::put return value: false
Expected Result	1. The response should be an instance of JsonResponse. 2. The response data should exactly match ['success' => true].
Actual Result	Both assertions pass: response is JsonResponse, and data is ['success' => true].
Status	Pass
Severity	High

File: FinishUpdateController-Test.txt

Test Case ID	FUC-001
Title	Unauthorized response when no user is authenticated
Objective	Verify that the FinishUpdateController returns a 401 Unauthorized response when a request is made without an authenticated user.
Preconditions	- Application running - No user is authenticated - Update endpoint is accessible
Test Steps	1. Create a mock Request object without an authenticated user. 2. Invoke the FinishUpdateController with the mock Request. 3. Capture the response.
Test Data	- Mock Request object with user() method returning null.

Expected Result	- Response status code is 401.
	- Response data is:
	{
	'success': false,
	'message': 'You are not allowed to update this app.'
	}

Actual Result	- Response returned status code 401 and the expected data.
Status	Pass
Severity	High

Test Case ID	FUC-002
Title	Unauthorized response when authenticated user is not owner
Objective	Verify that the FinishUpdateController returns a 401 Unauthorized response when the authenticated user does not have owner privileges.
Preconditions	- Application running - An authenticated user exists - Authenticated user is not an owner - Update endpoint is accessible
Test Steps	1. Create a mock user object with isOwner() method returning false. 2. Create a mock Request object with user() returning the mock user. 3. Invoke the FinishUpdateController with the mock Request. 4. Capture the response.
Test Data	- Mock user object with isOwner() returning false. - Mock Request object with user() returning the mock user.

Expected Result	- Response status code is 401.
	- Response data is:
	{
	'success': false,
	'message': 'You are not allowed to update this app.'
	}

Actual Result	- Response returned status code 401 and the expected data.
Status	Pass
Severity	High

Test Case ID	FUC-003
Title	Request parameter validation for 'installed' and 'version'
Objective	Verify that the FinishUpdateController correctly validates the 'installed' and 'version' parameters as required.

Preconditions	- Application running - Authenticated user with owner privileges - Update endpoint is accessible
Test Steps	1. Create a mock user object with isOwner() returning true. 2. Create a mock Request object with user() returning the mock user. 3. Mock the validate() method on Request to expect rules: - 'installed' => 'required' - 'version' => 'required' 4. Set installed and version properties on the mock Request. 5. Mock Updater::finishUpdate to expect and return a response. 6. Invoke the FinishUpdateController with the mock Request.
Test Data	- User is owner - Request parameters: installed = '2.0.0', version = '2.0.1' - Validation rules: ['installed' => 'required', 'version' => 'required']
Expected Result	- Request.validate() is called with the specified rules. - Updater::finishUpdate is called with '2.0.0' and '2.0.1'.
Actual Result	- validate() was called as required; Updater::finishUpdate was invoked with correct arguments.
Status	Pass
Severity	High

Test Case ID	FUC-004
Title	Successful update request returns updater response
Objective	Verify that the FinishUpdateController successfully processes a valid update request and returns the expected updater response.
Preconditions	- Application running - Authenticated user with owner privileges - Valid 'installed' and 'version' parameters provided - Update endpoint is accessible
Test Steps	1. Create a mock user object with isOwner() returning true. 2. Create a mock Request object with user() returning the mock user. 3. Mock validate() on Request to pass. 4. Set installed = '2.0.0' and version = '2.0.1' on Request. 5. Mock Updater::finishUpdate to return the expected successful response. 6. Invoke the FinishUpdateController with the mock Request. 7. Capture the response.

Test Data	- User is owner
	- Request parameters: installed = '2.0.0', version = '2.0.1'
	- Updater::finishUpdate response:
	{
	'status': 'success',
	'message': 'Update completed successfully.',
	'new_version': '2.0.1'
	}

Expected Result	- Response status code is 200.
	- Response data matches:
	{
	'status': 'success',
	'message': 'Update completed successfully.',
	'new_version': '2.0.1'
	}

Actual Result	- Response returned status code 200 and the expected data.
Status	Pass
Severity	High

Test Case ID	FUC-005
Title	Error response from updater is correctly returned

Objective	Verify that the FinishUpdateController passes through error responses received from the updater service.
Preconditions	- Application running - Authenticated user with owner privileges - Valid 'installed' and 'version' parameters provided - Update endpoint is accessible
Test Steps	1. Create a mock user object with isOwner() returning true. 2. Create a mock Request object with user() returning the mock user. 3. Mock validate() on Request to pass. 4. Set installed = '2.0.0' and version = '2.0.1' on Request. 5. Mock Updater::finishUpdate to return an error response. 6. Invoke the FinishUpdateController with the mock Request. 7. Capture the response.

Test Data	- User is owner
	- Request parameters: installed = '2.0.0', version = '2.0.1'
	- Updater::finishUpdate response:
	{
	'status': 'error',
	'message': 'An error occurred during update processing.',
	'details': 'Could not write to disk.'
	}

Expected Result	- Response status code is 200.
	- Response data matches:
	{
	'status': 'error',
	'message': 'An error occurred during update processing.',
	'details': 'Could not write to disk.'
	}

Actual Result	- Response returned status code 200 and the expected error data.
Status	Pass
Severity	High

File: ForgotPasswordController-Test.txt

Test Case ID	FPC-001
Title	ForgotPasswordController broker method returns customer password broker
Objective	Verify that the broker() method returns the correct customer password broker instance.
Preconditions	- Application running - Mockery available for mocking - Password facade accessible - Database seeded with customer authentication broker - ForgotPasswordController class available
Test Steps	1. Mock the PasswordBroker class instance. 2. Set up the Password facade to return the mocked broker when broker('customers') is called. 3. Instantiate the ForgotPasswordController. 4. Call the broker() method on the controller. 5. Assert that the returned object is the mocked broker instance.
Test Data	- Password broker type: 'customers' - Expected password broker: mocked PasswordBroker instance
Expected Result	- broker() returns the mocked PasswordBroker instance.
Actual Result	- broker() successfully returns the mocked PasswordBroker instance.
Status	Pass
Severity	High

Test Case ID	FPC-002
Title	sendResetLinkResponse returns successful JSON response
Objective	Verify that sendResetLinkResponse returns a JSON response with correct message and data on password reset request success.
Preconditions	- Application running - ForgotPasswordController class available - Controller exposing sendResetLinkResponse for testing - HTTP request with Accept: application/json header - Database seeded for password reset functionality
Test Steps	1. Create an anonymous class extending ForgotPasswordController and expose sendResetLinkResponse as a public method. 2. Create a POST HTTP request to '/test-url' with Accept: application/json header. 3. Set response string to 'passwords.sent'. 4. Call publicSendResetLinkResponse on the controller with the prepared request and response string. 5. Assert that the response is a JsonResponse instance. 6. Assert that status code is 200. 7. Assert that response data includes 'message' key with value 'Password reset email sent.' 8. Assert that response data includes 'data' key with value 'passwords.sent'.
Test Data	- Request: POST '/test-url', headers: Accept: application/json - Response string: 'passwords.sent'
Expected Result	- Response is a JsonResponse instance. - Status code is 200. - Response data includes 'message': 'Password reset email sent.' - Response data includes 'data': 'passwords.sent'.
Actual Result	- Response is a JsonResponse instance with status code 200, 'message' as 'Password reset email sent.', and 'data' as 'passwords.sent'.
Status	Pass
Severity	High

Test Case ID	FPC-003
Title	sendResetLinkFailedResponse returns failed response with 403 status
Objective	Verify that sendResetLinkFailedResponse returns a Response with correct failure message and status code for failed password reset attempts.
Preconditions	- Application running - ForgotPasswordController class available - Controller exposing sendResetLinkFailedResponse for testing - HTTP request with Accept: application/json header - Database seeded for password reset functionality
Test Steps	1. Create an anonymous class extending ForgotPasswordController and expose sendResetLinkFailedResponse as a public method. 2. Create a POST HTTP request to '/test-url' with Accept: application/json header. 3. Set response string to 'passwords.user'. 4. Call publicSendResetLinkFailedResponse on the controller with the prepared request and response string. 5. Assert that the response is an instance of Illuminate\Http\Response. 6. Assert that status code is 403. 7. Assert that response content is 'Email could not be sent to this email address.'
Test Data	- Request: POST '/test-url', headers: Accept: application/json - Response string: 'passwords.user'
Expected Result	- Response is an instance of Illuminate\Http\Response. - Status code is 403. - Response content is 'Email could not be sent to this email address.'
Actual Result	- Response is an instance of Illuminate\Http\Response with status code 403, and content 'Email could not be sent to this email address.'
Status	Pass
Severity	High

File: GenerateEstimatePdfJob-Test.txt

Test Case ID	GEPJ-001
Title	Constructor sets estimate and deleteExistingFile properties with default value
Objective	Verify that the GenerateEstimatePdfJob constructor correctly sets the estimate property and assigns false to deleteExistingFile when no explicit value is provided.
Preconditions	- Application running - Estimate object exists - Required dependencies are loaded
Test Steps	1. Create a minimal mock estimate object with id = 1 and estimate_number = 'EST-001'. 2. Instantiate GenerateEstimatePdfJob with the mock estimate object as the only parameter. 3. Inspect the job instance for the value of its estimate and deleteExistingFile properties.
Test Data	- Input: mockEstimate = { id: 1, estimate_number: 'EST-001' } - Expected values: job->estimate === mockEstimate; job->deleteExistingFile === false
Expected Result	- The job's estimate property is set to the mockEstimate object. - The job's deleteExistingFile property is set to false.
Actual Result	- The job's estimate property was set to the mockEstimate object. - The job's deleteExistingFile property was set to false.
Status	Pass
Severity	Medium

Test Case ID	GEPJ-002
Title	Constructor sets estimate and deleteExistingFile properties when explicitly true
Objective	Verify that the GenerateEstimatePdfJob constructor correctly sets the estimate property and assigns true to deleteExistingFile when explicitly set.
Preconditions	- Application running - Estimate object exists - Required dependencies are loaded
Test Steps	1. Create a mock estimate object with id = 2 and estimate_number = 'EST-002'. 2. Instantiate GenerateEstimatePdfJob with the mock estimate object and deleteExistingFile set to true. 3. Inspect the job instance for the value of its estimate and deleteExistingFile properties.
Test Data	- Input: mockEstimate = { id: 2, estimate_number: 'EST-002' }, deleteExistingFile = true - Expected values: job->estimate === mockEstimate; job->deleteExistingFile === true
Expected Result	- The job's estimate property is set to the mockEstimate object. - The job's deleteExistingFile property is set to true.
Actual Result	- The job's estimate property was set to the mockEstimate object. - The job's deleteExistingFile property was set to true.
Status	Pass
Severity	Medium

Test Case ID	GEPJ-003
Title	Handle calls generatePDF with correct arguments when deleteExistingFile is false

Objective	Verify that the handle() method of GenerateEstimatePdfJob calls generatePDF on the estimate object with correct arguments when deleteExistingFile is false.
Preconditions	- Application running - Mockery installed and configured for mocking objects - Estimate object with estimate_number='EST-003' exists
Test Steps	1. Mock an estimate object and set its estimate_number property to 'EST-003'. 2. Set up an expectation for generatePDF to be called once with arguments: 'estimate', 'EST-003', false. 3. Instantiate GenerateEstimatePdfJob with the mock estimate and deleteExistingFile = false. 4. Invoke the handle() method on the job.
Test Data	- Input: mockEstimate (estimate_number: 'EST-003'), deleteExistingFile = false - Expected call: generatePDF('estimate', 'EST-003', false)
Expected Result	- generatePDF is called once on the estimate object with arguments: 'estimate', 'EST-003', false.
Actual Result	- generatePDF was called once on the estimate object with arguments: 'estimate', 'EST-003', false.
Status	Pass
Severity	High

Test Case ID	GEPJ-004
Title	Handle calls generatePDF with correct arguments when deleteExistingFile is true
Objective	Verify that the handle() method of GenerateEstimatePdfJob calls generatePDF on the estimate object with correct arguments when deleteExistingFile is true.
Preconditions	- Application running - Mockery installed and configured for mocking objects - Estimate object with estimate_number='EST-004' exists
Test Steps	1. Mock an estimate object and set its estimate_number property to 'EST-004'. 2. Set up an expectation for generatePDF to be called once with arguments: 'estimate', 'EST-004', true. 3. Instantiate GenerateEstimatePdfJob with the mock estimate and deleteExistingFile = true. 4. Invoke the handle() method on the job.
Test Data	- Input: mockEstimate (estimate_number: 'EST-004'), deleteExistingFile = true - Expected call: generatePDF('estimate', 'EST-004', true)
Expected Result	- generatePDF is called once on the estimate object with arguments: 'estimate', 'EST-004', true.
Actual Result	- generatePDF was called once on the estimate object with arguments: 'estimate', 'EST-004', true.
Status	Pass
Severity	High

Test Case ID	GEPJ-005
Title	Handle returns 0 after successful execution
Objective	Verify that the handle() method of GenerateEstimatePdfJob returns 0 on successful completion.
Preconditions	- Application running - Mockery installed and configured for mocking objects - Estimate object with estimate_number='EST-005' exists

Test Steps	1. Mock an estimate object and set its estimate_number property to 'EST-005'. 2. Set up an expectation for generatePDF to be called once (arguments not critical). 3. Instantiate GenerateEstimatePdfJob with the mock estimate. 4. Invoke the handle() method on the job and store the return value.
Test Data	- Input: mockEstimate (estimate_number: 'EST-005') - Expected return value: 0
Expected Result	- The handle() method returns 0 after successful execution.
Actual Result	- The handle() method returned 0 after successful execution.
Status	Pass
Severity	High

File: GenerateInvoicePdfJob-Test.txt

Test Case ID	GIPJ-001
Title	Constructor assigns invoice and deleteExistingFile as true
Objective	Verify that GenerateInvoicePdfJob constructs with the correct invoice object and sets deleteExistingFile property to true.
Preconditions	- Application running - Database seeded with invoices - GenerateInvoicePdfJob class available
Test Steps	1. Create a mock invoice object with id=1 and invoice_number='INV-001'. 2. Set deleteExistingFile to true. 3. Construct a GenerateInvoicePdfJob instance with the invoice and deleteExistingFile as true. 4. Assert that the job's invoice property matches the created invoice object. 5. Assert that the job's deleteExistingFile property is true.
Test Data	- Input invoice: { id: 1, invoice_number: 'INV-001' } - Input deleteExistingFile: true - Expected invoice property: matches input invoice - Expected deleteExistingFile property: true
Expected Result	- The job's invoice property is the input invoice object. - The job's deleteExistingFile property is true.
Actual Result	- The job's invoice property matched the input invoice. - The job's deleteExistingFile property was true.
Status	Pass
Severity	Medium

Test Case ID	GIPJ-002
Title	Constructor assigns invoice and deleteExistingFile as false
Objective	Verify that GenerateInvoicePdfJob constructs with the correct invoice object and sets deleteExistingFile property to false.
Preconditions	- Application running - Database seeded with invoices - GenerateInvoicePdfJob class available
Test Steps	1. Create a mock invoice object with id=1 and invoice_number='INV-002'. 2. Set deleteExistingFile to false. 3. Construct a GenerateInvoicePdfJob instance with the invoice and deleteExistingFile as false. 4. Assert that the job's invoice property matches the created invoice object. 5. Assert that the job's deleteExistingFile property is false.
Test Data	- Input invoice: { id: 1, invoice_number: 'INV-002' } - Input deleteExistingFile: false - Expected invoice property: matches input invoice - Expected deleteExistingFile property: false
Expected Result	- The job's invoice property is the input invoice object. - The job's deleteExistingFile property is false.
Actual Result	- The job's invoice property matched the input invoice. - The job's deleteExistingFile property was false.
Status	Pass
Severity	Medium

Test Case ID	GIPJ-003
Title	Constructor assigns invoice with default deleteExistingFile as false
Objective	Verify that GenerateInvoicePdfJob constructs with the correct invoice object and defaults deleteExistingFile property to false.

Preconditions	- Application running - Database seeded with invoices - GenerateInvoicePdfJob class available
Test Steps	1. Create a mock invoice object with id=1 and invoice_number='INV-003'. 2. Construct GenerateInvoicePdfJob with only the invoice, omitting deleteExistingFile. 3. Assert that the job's invoice property matches the created invoice object. 4. Assert that the job's deleteExistingFile property is false.
Test Data	- Input invoice: { id: 1, invoice_number: 'INV-003' } - Input deleteExistingFile: unspecified (should default to false) - Expected invoice property: matches input invoice - Expected deleteExistingFile property: false
Expected Result	- The job's invoice property is the input invoice object. - The job's deleteExistingFile property is false.
Actual Result	- The job's invoice property matched the input invoice. - The job's deleteExistingFile property was false.
Status	Pass
Severity	Medium

Test Case ID	GIPJ-004
Title	handle() calls generatePDF with invoice_number and deleteExistingFile true
Objective	Verify that handle() calls generatePDF on invoice with correct parameters when deleteExistingFile is true.
Preconditions	- Application running - Database seeded with invoices - Mockery available for mocking invoice object - GenerateInvoicePdfJob class available
Test Steps	1. Create a mock invoice object with invoice_number='INV-TEST-001'. 2. Setup a mock expectation for generatePDF to be called with ('invoice', 'INV-TEST-001', true). 3. Construct GenerateInvoicePdfJob with the mock invoice and deleteExistingFile as true. 4. Call job->handle(). 5. Assert that handle() returns 0.
Test Data	- Input invoice_number: 'INV-TEST-001' - Input deleteExistingFile: true - Expected generatePDF params: ('invoice', 'INV-TEST-001', true) - Expected handle() return: 0
Expected Result	- generatePDF is called once with proper arguments. - handle() returns 0.
Actual Result	- generatePDF was called as expected. - handle() returned 0.
Status	Pass
Severity	High

Test Case ID	GIPJ-005
Title	handle() calls generatePDF with invoice_number and deleteExistingFile false
Objective	Verify that handle() calls generatePDF on invoice with correct parameters when deleteExistingFile is false.
Preconditions	- Application running - Database seeded with invoices - Mockery available for mocking invoice object - GenerateInvoicePdfJob class available

Test Steps	1. Create a mock invoice object with invoice_number='INV-TEST-002'. 2. Setup a mock expectation for generatePDF to be called with ('invoice', 'INV-TEST-002', false). 3. Construct GenerateInvoicePdfJob with the mock invoice and deleteExistingFile as false. 4. Call job->handle(). 5. Assert that handle() returns 0.
Test Data	- Input invoice_number: 'INV-TEST-002' - Input deleteExistingFile: false - Expected generatePDF params: ('invoice', 'INV-TEST-002', false) - Expected handle() return: 0
Expected Result	- generatePDF is called once with proper arguments. - handle() returns 0.
Actual Result	- generatePDF was called as expected. - handle() returned 0.
Status	Pass
Severity	High

Test Case ID	GIPJ-006
Title	handle() calls generatePDF with default false for deleteExistingFile
Objective	Verify that handle() calls generatePDF on invoice with default false for deleteExistingFile.
Preconditions	- Application running - Database seeded with invoices - Mockery available for mocking invoice object - GenerateInvoicePdfJob class available
Test Steps	1. Create a mock invoice object with invoice_number='INV-TEST-003'. 2. Setup a mock expectation for generatePDF to be called with ('invoice', 'INV-TEST-003', false). 3. Construct GenerateInvoicePdfJob with only the mock invoice. 4. Call job->handle(). 5. Assert that handle() returns 0.
Test Data	- Input invoice_number: 'INV-TEST-003' - Input deleteExistingFile: not passed (should default to false) - Expected generatePDF params: ('invoice', 'INV-TEST-003', false) - Expected handle() return: 0
Expected Result	- generatePDF is called once with proper arguments. - handle() returns 0.
Actual Result	- generatePDF was called as expected. - handle() returned 0.
Status	Pass
Severity	High

Test Case ID	GIPJ-007
Title	handle() calls generatePDF with empty invoice_number when null
Objective	Verify that handle() passes an empty string for invoice_number to generatePDF when invoice_number is null.
Preconditions	- Application running - Database seeded with invoices - Mockery available for mocking invoice object - GenerateInvoicePdfJob class available

Test Steps	1. Create a mock invoice object with invoice_number=null. 2. Setup a mock expectation for generatePDF to be called with ('invoice', '', false). 3. Construct GenerateInvoicePdfJob with the mock invoice. 4. Call job->handle(). 5. Assert that handle() returns 0.
Test Data	- Input invoice_number: null - Input deleteExistingFile: not passed (should default to false) - Expected generatePDF params: ('invoice', '', false) - Expected handle() return: 0
Expected Result	- generatePDF is called once with proper arguments (empty string for invoice_number). - handle() returns 0.
Actual Result	- generatePDF was called as expected. - handle() returned 0.
Status	Pass
Severity	High

Test Case ID	GIPJ-008
Title	handle() calls generatePDF with empty invoice_number when empty string
Objective	Verify that handle() passes an empty string for invoice_number to generatePDF when invoice_number property is an empty string.
Preconditions	- Application running - Database seeded with invoices - Mockery available for mocking invoice object - GenerateInvoicePdfJob class available
Test Steps	1. Create a mock invoice object with invoice_number=''. 2. Setup a mock expectation for generatePDF to be called with ('invoice', '', false). 3. Construct GenerateInvoicePdfJob with the mock invoice. 4. Call job->handle(). 5. Assert that handle() returns 0.
Test Data	- Input invoice_number: '' - Input deleteExistingFile: not passed (should default to false) - Expected generatePDF params: ('invoice', '', false) - Expected handle() return: 0
Expected Result	- generatePDF is called once with proper arguments (empty string for invoice_number). - handle() returns 0.
Actual Result	- generatePDF was called as expected. - handle() returned 0.
Status	Pass
Severity	High

File: GeneratePaymentPdfJob-Test.txt

Test Case ID	GPPJ-001
Title	Constructor assigns payment and deleteExistingFile properties correctly
Objective	Verify that the constructor of GeneratePaymentPdfJob correctly assigns the payment and deleteExistingFile properties with both explicit and default values.
Preconditions	- Application running - Database seeded - GeneratePaymentPdfJob class available - Mockery library installed
Test Steps	1. Create a mock payment object. 2. Instantiate GeneratePaymentPdfJob with the mock payment and deleteExistingFile set to true. 3. Verify that the payment property matches the mock payment. 4. Verify that the deleteExistingFile property is true. 5. Instantiate GeneratePaymentPdfJob with only the mock payment (deleteExistingFile not provided). 6. Verify that the deleteExistingFile property is false by default.
Test Data	- Mock payment object - deleteExistingFile values: true and (default) false
Expected Result	- The payment property should match the mock payment object. - The deleteExistingFile property should be true when passed in the constructor. - The deleteExistingFile property should be false when not passed in the constructor.
Actual Result	All properties are set as expected. The constructor correctly assigns values.
Status	Pass
Severity	Medium

Test Case ID	GPPJ-002
Title	handle() calls generatePDF and returns 0 when deleteExistingFile is true
Objective	Verify that handle() method calls generatePDF with correct arguments and returns 0 when deleteExistingFile is set to true.
Preconditions	- Application running - Database seeded - GeneratePaymentPdfJob and payment class available - Mockery library installed
Test Steps	1. Create a mock payment object with payment_number "PAY-001". 2. Expect the generatePDF method to be called with arguments: ('payment', 'PAY-001', true). 3. Instantiate GeneratePaymentPdfJob with the mock payment object and deleteExistingFile set to true. 4. Call the handle() method on the job. 5. Assert the return value is 0.
Test Data	- payment_number: "PAY-001" - deleteExistingFile: true
Expected Result	- generatePDF method is called once with ('payment', 'PAY-001', true). - handle() method returns 0.
Actual Result	generatePDF called with correct arguments and handle() returned 0.
Status	Pass
Severity	High

Test Case ID	GPPJ-003

Title	handle() calls generatePDF and returns 0 when deleteExistingFile is false
Objective	Verify that handle() calls generatePDF with correct arguments and returns 0 when deleteExistingFile is false.
Preconditions	- Application running - Database seeded - GeneratePaymentPdfJob and payment class available - Mockery library installed
Test Steps	1. Create a mock payment object with payment_number "PAY-002". 2. Set expectation for generatePDF to be called with ('payment', 'PAY-002', false). 3. Instantiate GeneratePaymentPdfJob with the mock payment and deleteExistingFile = false. 4. Call handle() method on the job. 5. Assert the return value is 0.
Test Data	- payment_number: "PAY-002" - deleteExistingFile: false
Expected Result	- generatePDF method is called once with ('payment', 'PAY-002', false). - handle() method returns 0.
Actual Result	generatePDF invoked as expected and handle() returned 0.
Status	Pass
Severity	High

Test Case ID	GPPJ-004
Title	handle() uses default deleteExistingFile value (false) when not provided
Objective	Verify that handle() uses deleteExistingFile default value of false when not explicitly provided.
Preconditions	- Application running - Database seeded - GeneratePaymentPdfJob and payment class available - Mockery library installed
Test Steps	1. Create a mock payment object with payment_number "PAY-003". 2. Expect generatePDF to be called with ('payment', 'PAY-003', false). 3. Instantiate GeneratePaymentPdfJob with the mock payment (without deleteExistingFile). 4. Call handle() method on the job. 5. Assert the return value is 0.
Test Data	- payment_number: "PAY-003" - deleteExistingFile: (default) false
Expected Result	- generatePDF is called once with ('payment', 'PAY-003', false). - handle() returns 0.
Actual Result	generatePDF triggered with correct defaults and handle() returned 0.
Status	Pass
Severity	High

Test Case ID	GPPJ-005
Title	handle() works when payment_number is null
Objective	Verify that handle() processes job correctly when payment_number is null.
Preconditions	- Application running - Database seeded - GeneratePaymentPdfJob and payment class available - Mockery library installed

Test Steps	1. Create a mock payment object with payment_number set to null. 2. Expect generatePDF to be called with ('payment', null, false). 3. Instantiate GeneratePaymentPdfJob with the mock payment and deleteExistingFile set to false. 4. Call handle() method on the job. 5. Assert the return value is 0.
Test Data	- payment_number: null - deleteExistingFile: false
Expected Result	- generatePDF is called once with ('payment', null, false). - handle() returns 0.
Actual Result	generatePDF executed with null payment_number and handle() returned 0.
Status	Pass
Severity	High

Test Case ID	GPPJ-006
Title	handle() works when payment_number is an empty string
Objective	Verify that handle() processes job correctly when payment_number is an empty string.
Preconditions	- Application running - Database seeded - GeneratePaymentPdfJob and payment class available - Mockery library installed
Test Steps	1. Create a mock payment object with payment_number set to "" (empty string). 2. Expect generatePDF to be called with ('payment', '', true). 3. Instantiate GeneratePaymentPdfJob with the mock payment and deleteExistingFile set to true. 4. Call handle() method on the job. 5. Assert the return value is 0.
Test Data	- payment_number: "" - deleteExistingFile: true
Expected Result	- generatePDF is called once with ('payment', '', true). - handle() returns 0.
Actual Result	generatePDF executed with empty string payment_number and handle() returned 0.
Status	Pass
Severity	High

File: GeneratesMenuTrait-Test.txt

Test Case ID	GMT-001
Title	Trait can be used by a class
Objective	Verify that the GeneratesMenuTrait can be used in class instantiation.
Preconditions	- Application running - GeneratesMenuTrait is defined - DummyMenuGenerator class uses the trait
Test Steps	1. Instantiate DummyMenuGenerator. 2. Assert that the instance is of class DummyMenuGenerator.
Test Data	- Class: DummyMenuGenerator
Expected Result	- Instance is of DummyMenuGenerator.
Actual Result	Instance is of DummyMenuGenerator.
Status	Pass
Severity	Medium

Test Case ID	GMT-002
Title	Trait is in correct namespace
Objective	Verify GeneratesMenuTrait is in the Crater\Traits namespace.
Preconditions	- Application running - GeneratesMenuTrait is loaded
Test Steps	1. Use Reflection to access GeneratesMenuTrait. 2. Retrieve the namespace name. 3. Check if the namespace is 'Crater\Traits'.
Test Data	- Trait: GeneratesMenuTrait
Expected Result	- Namespace name is 'Crater\Traits'.
Actual Result	Namespace name is 'Crater\Traits'.
Status	Pass
Severity	Low

Test Case ID	GMT-003
Title	Trait is a trait
Objective	Ensure GeneratesMenuTrait is a trait, not a class or interface.
Preconditions	- Application running - GeneratesMenuTrait is loaded
Test Steps	1. Use Reflection to access GeneratesMenuTrait. 2. Check isTrait() property.
Test Data	- Trait: GeneratesMenuTrait
Expected Result	- isTrait() returns true.
Actual Result	isTrait() returns true.
Status	Pass
Severity	Low

Test Case ID	GMT-004
Title	Trait has generateMenu method
Objective	Verify the class using the trait has the generateMenu method.
Preconditions	- DummyMenuGenerator is instantiated

Test Steps	1. Instantiate DummyMenuGenerator. 2. Check if 'generateMenu' exists in the instance.
Test Data	- Class: DummyMenuGenerator
Expected Result	- 'generateMenu' method exists.
Actual Result	'generateMenu' method exists.
Status	Pass
Severity	High

Test Case ID	GMT-005
Title	generateMenu method is public
Objective	Ensure generateMenu is a public method.
Preconditions	- DummyMenuGenerator class uses GeneratesMenuTrait
Test Steps	1. Use Reflection to access DummyMenuGenerator class. 2. Retrieve 'generateMenu' method. 3. Assert method is public.
Test Data	- Method: generateMenu
Expected Result	- Method is public.
Actual Result	Method is public.
Status	Pass
Severity	High

Test Case ID	GMT-006
Title	generateMenu is not static
Objective	Assert generateMenu is not a static method.
Preconditions	- DummyMenuGenerator class uses GeneratesMenuTrait
Test Steps	1. Use Reflection to retrieve generateMenu method. 2. Assert method is not static.
Test Data	- Method: generateMenu
Expected Result	- isStatic() returns false.
Actual Result	isStatic() returns false.
Status	Pass
Severity	Medium

Test Case ID	GMT-007
Title	generateMenu accepts two parameters
Objective	Validate parameter structure of generateMenu (key, user).
Preconditions	- DummyMenuGenerator is loaded
Test Steps	1. Use Reflection to get generateMenu parameters. 2. Count and assert parameter names.
Test Data	- Method: generateMenu
Expected Result	- Two parameters named 'key' and 'user'.
Actual Result	Two parameters named 'key' and 'user'.
Status	Pass
Severity	High

Test Case ID	GMT-008

Title	Multiple instances using trait can be created
Objective	Ensure multiple DummyMenuGenerator instances are unique.
Preconditions	- Application running
Test Steps	1. Instantiate generator1 and generator2. 2. Assert both are DummyMenuGenerator. 3. Assert generator1 and generator2 are not the same object.
Test Data	- Class: DummyMenuGenerator
Expected Result	- Both instances are of DummyMenuGenerator. - generator1 is not the same as generator2.
Actual Result	Both instances are DummyMenuGenerator and unique.
Status	Pass
Severity	Medium

Test Case ID	GMT-009
Title	Class using trait can be cloned
Objective	Validate cloning works for class using GeneratesMenuTrait.
Preconditions	- DummyMenuGenerator class uses trait
Test Steps	1. Instantiate DummyMenuGenerator. 2. Clone the instance. 3. Assert clone is DummyMenuGenerator. 4. Assert clone is unique from original.
Test Data	- Instance of DummyMenuGenerator
Expected Result	- Clone is DummyMenuGenerator, and is not the original instance.
Actual Result	Clone is DummyMenuGenerator, and unique.
Status	Pass
Severity	Medium

Test Case ID	GMT-010
Title	Class using trait supports type hints
Objective	Ensure DummyMenuGenerator can be used in type hinting.
Preconditions	- DummyMenuGenerator class exists
Test Steps	1. Define a function with DummyMenuGenerator parameter. 2. Pass a DummyMenuGenerator instance. 3. Assert instance passes through type hint.
Test Data	- Instance of DummyMenuGenerator
Expected Result	- Instance is accepted via type hint.
Actual Result	Instance accepted.
Status	Pass
Severity	Low

Test Case ID	GMT-011
Title	Trait is not a class
Objective	Ensure GeneratesMenuTrait is not a class.
Preconditions	- Trait is loaded
Test Steps	1. Use Reflection to check isInterface() on trait.
Test Data	- Trait: GeneratesMenuTrait
Expected Result	- isInterface() returns false.

Actual Result	isInterface() returns false.
Status	Pass
Severity	Low

Test Case ID	GMT-012
Title	Trait is not an interface
Objective	Assert that GeneratesMenuTrait is not an interface.
Preconditions	- Trait is available
Test Steps	1. Use Reflection to check isInterface() on GeneratesMenuTrait.
Test Data	- Trait: GeneratesMenuTrait
Expected Result	- isInterface() returns false.
Actual Result	isInterface() returns false.
Status	Pass
Severity	Low

Test Case ID	GMT-013
Title	Trait is loaded
Objective	Confirm GeneratesMenuTrait is loaded in the system.
Preconditions	- Application running
Test Steps	1. Use trait_exists() function for GeneratesMenuTrait.
Test Data	- Trait name: Crater\Traits\GeneratesMenuTrait
Expected Result	- trait_exists() returns true.
Actual Result	trait_exists() returns true.
Status	Pass
Severity	Medium

Test Case ID	GMT-014
Title	Trait file has expected structure
Objective	Ensure trait source file contains expected components.
Preconditions	- GeneratesMenuTrait class file available
Test Steps	1. Use Reflection to get trait file content. 2. Check content contains 'trait GeneratesMenuTrait'. 3. Check content contains 'public function generateMenu'.
Test Data	- Trait source file
Expected Result	- Source contains trait declaration and public method.
Actual Result	Source contains required structure.
Status	Pass
Severity	Low

Test Case ID	GMT-015
Title	Trait has minimal line count
Objective	Ensure GeneratesMenuTrait file is concise.
Preconditions	- Trait file available
Test Steps	1. Load the trait file content. 2. Count lines in the file. 3. Assert total line count is less than 50.

Test Data	- Trait file content
Expected Result	- Line count < 50.
Actual Result	Line count is less than 50.
Status	Pass
Severity	Low

Test Case ID	GMT-016
Title	generateMenu uses Menu facade
Objective	Check trait method uses Menu facade in its logic.
Preconditions	- Trait source available
Test Steps	1. Load GeneratesMenuTrait file. 2. Assert file contains '\Menu::get' usage.
Test Data	- Trait source file
Expected Result	- File contains '\Menu::get'.
Actual Result	File contains '\Menu::get'.
Status	Pass
Severity	High

Test Case ID	GMT-017
Title	generateMenu uses foreach loop
Objective	Ensure trait method iterates over menu items.
Preconditions	- Trait source loaded
Test Steps	1. Load trait file content. 2. Assert 'foreach' loop is present.
Test Data	- Trait file content
Expected Result	- File contains 'foreach' loop.
Actual Result	File contains 'foreach' loop.
Status	Pass
Severity	High

Test Case ID	GMT-018
Title	generateMenu calls checkAccess on user
Objective	Confirm menu generation checks user access.
Preconditions	- Trait logic uses user object
Test Steps	1. Load trait file content. 2. Assert '$user->checkAccess' is called.
Test Data	- Trait logic
Expected Result	- '$user->checkAccess' is present in method.
Actual Result	'$user->checkAccess' call found.
Status	Pass
Severity	High

Test Case ID	GMT-019
Title	generateMenu builds array with title
Objective	Ensure output array includes item title.

Preconditions	- Trait file loaded
Test Steps	1. Check trait output structuring for 'title' key.
Test Data	- Trait file content
Expected Result	- Output array includes 'title' from data object.
Actual Result	Output array includes 'title'.
Status	Pass
Severity	Medium

Test Case ID	GMT-020
Title	generateMenu builds array with link
Objective	Ensure output includes item link information.
Preconditions	- Trait file and data structure defined
Test Steps	1. Check trait output structuring for 'link' key.
Test Data	- Trait file content
Expected Result	- Output array includes link (path['url']) from data object.
Actual Result	Output includes 'link' key.
Status	Pass
Severity	Medium

Test Case ID	GMT-021
Title	generateMenu builds array with icon
Objective	Ensure output contains item icon.
Preconditions	- Trait file loaded
Test Steps	1. Inspect trait output for 'icon' key.
Test Data	- Trait file content
Expected Result	- Output array includes 'icon' value.
Actual Result	Output array contains 'icon'.
Status	Pass
Severity	Low

Test Case ID	GMT-022
Title	generateMenu builds array with name
Objective	Ensure output includes item name.
Preconditions	- Trait file and menu item data available
Test Steps	1. Verify that trait output contains 'name' key.
Test Data	- Trait file content
Expected Result	- Output array includes 'name' value.
Actual Result	Output array contains 'name'.
Status	Pass
Severity	Low

Test Case ID	GMT-023
Title	generateMenu builds array with group
Objective	Ensure menu item group information is included.

Preconditions	- Trait file loaded
Test Steps	1. Inspect trait output for 'group' key.
Test Data	- Trait file content
Expected Result	- Output array includes 'group' value.
Actual Result	Output array contains 'group'.
Status	Pass
Severity	Low

Test Case ID	GMT-024
Title	generateMenu returns array
Objective	Confirm trait method returns an array.
Preconditions	- Trait file logic implemented
Test Steps	1. Inspect trait method for 'return $menu' statement.
Test Data	- Trait file content
Expected Result	- Method returns array variable.
Actual Result	Method returns array.
Status	Pass
Severity	High

Test Case ID	GMT-025
Title	generateMenu initializes empty menu array
Objective	Ensure trait starts with empty array before processing.
Preconditions	- Trait file logic present
Test Steps	1. Verify trait method sets $menu to [] before processing.
Test Data	- Trait file content
Expected Result	- Method includes '$menu = []'.
Actual Result	'$menu = []' present.
Status	Pass
Severity	Medium

Test Case ID	GMT-026
Title	DummyMenuGenerator uses GeneratesMenuTrait
Objective	Confirm class uses the trait as intended.
Preconditions	- DummyMenuGenerator class defined
Test Steps	1. Use Reflection to get trait names of DummyMenuGenerator. 2. Assert 'Crater\Traits\GeneratesMenuTrait' is present.
Test Data	- Class: DummyMenuGenerator
Expected Result	- Trait is listed in traits used by class.
Actual Result	Trait found in used traits.
Status	Pass
Severity	High

Test Case ID	GMT-027
Title	Trait provides generateMenu method to using class
Objective	Ensure trait injects generateMenu into DummyMenuGenerator.

Preconditions	- DummyMenuGenerator using trait
Test Steps	1. Use Reflection to check if method exists in class.
Test Data	- Class: DummyMenuGenerator
Expected Result	- 'generateMenu' is a method of DummyMenuGenerator.
Actual Result	Method exists.
Status	Pass
Severity	High

Test Case ID	GMT-028
Title	Trait has exactly one method
Objective	Check number and name of methods defined in trait.
Preconditions	- Trait loaded
Test Steps	1. Use Reflection to get all methods of trait. 2. Count methods. 3. Assert only one method exists and named 'generateMenu'.
Test Data	- Trait: GeneratesMenuTrait
Expected Result	- Only one method, called 'generateMenu'.
Actual Result	Only one method, 'generateMenu'.
Status	Pass
Severity	Low

Test Case ID	GMT-029
Title	generateMenu creates array with required keys
Objective	Verify output array includes all necessary keys.
Preconditions	- Trait file loaded
Test Steps	1. Inspect trait output for keys: 'title', 'link', 'icon', 'name', 'group'.
Test Data	- Trait file content
Expected Result	- Output array includes all required keys.
Actual Result	Output array includes keys: 'title', 'link', 'icon', 'name', 'group'.
Status	Pass
Severity	High

Test Case ID	GMT-030
Title	generateMenu uses if statement for access check
Objective	Confirm trait method uses if logic to restrict access.
Preconditions	- Trait logic present
Test Steps	1. Inspect trait method for 'if ($user->checkAccess($data))'.
Test Data	- Trait file content
Expected Result	- Access condition present.
Actual Result	Access condition present.
Status	Pass
Severity	High

Test Case ID	GMT-031
Title	generateMenu appends to menu array

Objective	Ensure menu items are added to output array.
Preconditions	- Trait source loaded
Test Steps	1. Inspect trait block for '$menu[]' statement.
Test Data	- Trait file content
Expected Result	- '$menu[]' used for appending data.
Actual Result	'$menu[]' found.
Status	Pass
Severity	Medium

Test Case ID	GMT-032
Title	generateMenu accesses items property
Objective	Confirm items property is accessed for menu generation.
Preconditions	- Trait logic available
Test Steps	1. Check trait method accesses '->items'.
Test Data	- Trait file content
Expected Result	- '->items' used in method.
Actual Result	'->items' accessed.
Status	Pass
Severity	High

Test Case ID	GMT-033
Title	generateMenu calls toArray on items
Objective	Validate trait converts menu items to array.
Preconditions	- Trait code implemented
Test Steps	1. Verify method calls '->toArray()' on items.
Test Data	- Trait file content
Expected Result	- '->toArray()' called.
Actual Result	'->toArray()' present.
Status	Pass
Severity	Medium

Test Case ID	GMT-034
Title	generateMenu accesses data object properties
Objective	Confirm trait accesses title, link, data of menu item object.
Preconditions	- Trait file loaded
Test Steps	1. Inspect method for access to $data->title, $data->link, $data->data.
Test Data	- Trait method body
Expected Result	- $data->title, $data->link, $data->data accessed.
Actual Result	Properties accessed as expected.
Status	Pass
Severity	Medium

Test Case ID	GMT-035
Title	Trait is in Traits namespace

Objective	Verify trait namespace is 'Crater\Traits'.
Preconditions	- Trait loaded
Test Steps	1. Use Reflection to get namespace of trait.
Test Data	- Trait: GeneratesMenuTrait
Expected Result	- Namespace is 'Crater\Traits'.
Actual Result	Namespace is correct.
Status	Pass
Severity	Low

Test Case ID	GMT-036
Title	Trait file declares namespace
Objective	Ensure source file declares the correct namespace.
Preconditions	- Trait file available
Test Steps	1. Load trait file content. 2. Assert it contains 'namespace Crater\Traits'.
Test Data	- Trait source file
Expected Result	- File declares namespace.
Actual Result	Namespace declaration present.
Status	Pass
Severity	Low

Test Case ID	GMT-037
Title	Trait file is concise
Objective	Check trait file size is under threshold.
Preconditions	- Trait source available
Test Steps	1. Measure trait file length. 2. Assert length is less than 1000 bytes.
Test Data	- Trait source file
Expected Result	- File size < 1000 bytes.
Actual Result	File is concise.
Status	Pass
Severity	Low

Test Case ID	GMT-038
Title	generateMenu uses descriptive variable names
Objective	Confirm variable naming conventions for clarity.
Preconditions	- Trait file available
Test Steps	1. Verify usage of $menu, $key, $user, $data in method.
Test Data	- Trait file content
Expected Result	- Descriptive variable names present.
Actual Result	Variable names are descriptive.
Status	Pass
Severity	Low

Test Case ID	GMT-039

Title	generateMenu iterates over menu items
Objective	Ensure foreach loop processes menu items from Menu facade.
Preconditions	- Trait file loaded
Test Steps	1. Check method contains 'foreach (\Menu::get($key)->items->toArray() as $data)'.
Test Data	- Trait file content
Expected Result	- Correct iteration over menu items.
Actual Result	Iteration implemented as expected.
Status	Pass
Severity	High

Test Case ID	GMT-040
Title	Trait has no properties
Objective	Verify trait does not define class properties.
Preconditions	- Trait loaded
Test Steps	1. Use Reflection to list properties. 2. Assert properties list is empty.
Test Data	- Trait: GeneratesMenuTrait
Expected Result	- No properties defined.
Actual Result	No properties present.
Status	Pass
Severity	Low

Test Case ID	GMT-041
Title	Trait has no constants
Objective	Confirm trait does not declare class constants.
Preconditions	- Trait loaded
Test Steps	1. Use Reflection to list trait constants. 2. Assert trait has no constants.
Test Data	- Trait: GeneratesMenuTrait
Expected Result	- No constants defined.
Actual Result	No constants present.
Status	Pass
Severity	Low

File: GetSettingRequest-Test.txt

Test Case ID	GSR-001
Title	Authorize method in GetSettingRequest returns true
Objective	To verify that the authorize() method of GetSettingRequest allows access.
Preconditions	- Application running - GetSettingRequest class available - No additional authentication restrictions
Test Steps	1. Instantiate the GetSettingRequest object. 2. Invoke the authorize() method on the object. 3. Capture the return value.
Test Data	- GetSettingRequest instance
Expected Result	- The authorize() method returns true.
Actual Result	The authorize() method returned true.
Status	Pass
Severity	Medium

Test Case ID	GSR-002
Title	rules method returns the correct validation rules structure
Objective	To verify that the rules() method in GetSettingRequest exposes the expected validation structure.
Preconditions	- Application running - GetSettingRequest class available
Test Steps	1. Instantiate the GetSettingRequest object. 2. Call the rules() method. 3. Compare output to the expected array: ['key' => ['required', 'string']].
Test Data	- Expected rules: ['key' => ['required', 'string']]
Expected Result	- The rules() method returns ['key' => ['required', 'string']].
Actual Result	The rules() method returned the expected validation array.
Status	Pass
Severity	Medium

Test Case ID	GSR-003
Title	rules method successfully validates data with a valid string key
Objective	To verify that valid data passes validation against rules().
Preconditions	- Application running - GetSettingRequest class available
Test Steps	1. Instantiate GetSettingRequest. 2. Prepare data: ['key' => 'valid_setting_key']. 3. Validate data using Laravel Validator with rules from GetSettingRequest. 4. Check validator passes() result.
Test Data	- Data: ['key' => 'valid_setting_key']
Expected Result	- Validation passes (validator->passes() returns true).
Actual Result	Validation passed successfully with a valid string key.
Status	Pass
Severity	Medium

Test Case ID	GSR-004

Title	rules method fails validation when the key is missing
Objective	To verify validation fails when required field 'key' is missing.
Preconditions	- Application running - GetSettingRequest class available
Test Steps	1. Instantiate GetSettingRequest. 2. Prepare data: [] (empty array). 3. Validate data using Laravel Validator with rules from GetSettingRequest. 4. Confirm validator->fails() returns true. 5. Check that validator->errors()->has('key') returns true. 6. Verify error message: validator->errors()->first('key').
Test Data	- Data: []
Expected Result	- Validation fails. - Error exists for 'key'. - Error message: 'The key field is required.'
Actual Result	Validation failed, 'key' error present, correct error message returned.
Status	Pass
Severity	High

Test Case ID	GSR-005
Title	rules method fails validation when the key is an empty string
Objective	To verify validation fails when required field 'key' is empty.
Preconditions	- Application running - GetSettingRequest class available
Test Steps	1. Instantiate GetSettingRequest. 2. Prepare data: ['key' => '']. 3. Validate data using Laravel Validator with rules from GetSettingRequest. 4. Confirm validator->fails() returns true. 5. Check that validator->errors()->has('key') returns true. 6. Verify error message: validator->errors()->first('key').
Test Data	- Data: ['key' => '']
Expected Result	- Validation fails. - Error exists for 'key'. - Error message: 'The key field is required.'
Actual Result	Validation failed, 'key' error present, correct error message returned.
Status	Pass
Severity	High

Test Case ID	GSR-006
Title	rules method fails validation when the key is not a string (integer)
Objective	To verify validation fails when field 'key' is of integer type.
Preconditions	- Application running - GetSettingRequest class available
Test Steps	1. Instantiate GetSettingRequest. 2. Prepare data: ['key' => 123]. 3. Validate data using Laravel Validator. 4. Confirm validator->fails() returns true. 5. Check that validator->errors()->has('key') returns true. 6. Verify error message: validator->errors()->first('key').
Test Data	- Data: ['key' => 123]
Expected Result	- Validation fails. - Error exists for 'key'. - Error message: 'The key must be a string.'

Actual Result	Validation failed for integer key, correct error message returned.
Status	Pass
Severity	High

Test Case ID	GSR-007
Title	rules method fails validation when the key is not a string (boolean)
Objective	To verify validation fails when field 'key' is a boolean.
Preconditions	- Application running - GetSettingRequest class available
Test Steps	1. Instantiate GetSettingRequest. 2. Prepare data: ['key' => true]. 3. Validate data using Laravel Validator. 4. Confirm validator->fails() returns true. 5. Check that validator->errors()->has('key') returns true. 6. Verify error message: validator->errors()->first('key').
Test Data	- Data: ['key' => true]
Expected Result	- Validation fails. - Error exists for 'key'. - Error message: 'The key must be a string.'
Actual Result	Validation failed for boolean key, correct error message returned.
Status	Pass
Severity	High

Test Case ID	GSR-008
Title	rules method fails validation when the key is not a string (array)
Objective	To verify validation fails when field 'key' is an array.
Preconditions	- Application running - GetSettingRequest class available
Test Steps	1. Instantiate GetSettingRequest. 2. Prepare data: ['key' => ['an', 'array']]. 3. Validate data using Laravel Validator. 4. Confirm validator->fails() returns true. 5. Check that validator->errors()->has('key') returns true. 6. Verify error message: validator->errors()->first('key').
Test Data	- Data: ['key' => ['an', 'array']]
Expected Result	- Validation fails. - Error exists for 'key'. - Error message: 'The key must be a string.'
Actual Result	Validation failed for array key, correct error message returned.
Status	Pass
Severity	High

Test Case ID	GSR-009
Title	rules method fails validation when the key is not a string (null)
Objective	To verify validation fails when field 'key' is null.
Preconditions	- Application running - GetSettingRequest class available

Test Steps	1. Instantiate GetSettingRequest. 2. Prepare data: ['key' => null]. 3. Validate data using Laravel Validator. 4. Confirm validator->fails() returns true. 5. Check that validator->errors()->has('key') returns true. 6. Verify error message: validator->errors()->first('key').
Test Data	- Data: ['key' => null]
Expected Result	- Validation fails. - Error exists for 'key'. - Error message: 'The key field is required.'
Actual Result	Validation failed for null key, required error message returned.
Status	Pass
Severity	High

File: GetSettingsRequest-Test.txt

Test Case ID	GSR-001
Title	Authorization method always returns true
Objective	Verify that the authorize() method of GetSettingsRequest always returns true
Preconditions	- Application is running - GetSettingsRequest class is loaded - No specific user authentication required
Test Steps	1. Instantiate a new GetSettingsRequest object. 2. Invoke the authorize() method on the object.
Test Data	- Method: authorize() - Instance: GetSettingsRequest
Expected Result	- The authorize() method returns true.
Actual Result	- The authorize() method returned true.
Status	Pass
Severity	Medium

Test Case ID	GSR-002
Title	Validation rules structure correctness
Objective	Ensure that rules() method in GetSettingsRequest returns the correct array structure for validation
Preconditions	- Application is running - GetSettingsRequest class is loaded
Test Steps	1. Instantiate a new GetSettingsRequest object. 2. Invoke the rules() method on the object. 3. Compare the returned array to the expected structure.
Test Data	- Expected validation rules: 'settings' => ['required'], 'settings.*' => ['required', 'string']
Expected Result	- The rules() method returns an array matching the expected validation rules structure.
Actual Result	- The returned array matches the expected validation rules structure.
Status	Pass
Severity	Medium

Test Case ID	GSR-003
Title	Valid settings data passes validation
Objective	Verify that correct settings input data passes validation using the rules()
Preconditions	- Application is running - GetSettingsRequest class is loaded
Test Steps	1. Instantiate a new GetSettingsRequest object. 2. Retrieve its validation rules. 3. Prepare input data with all required fields present and set to valid string values. 4. Validate the data using Laravel's Validator::make(). 5. Check if validation passes using passes().

Test Data	- Input data: ['settings' => ['app_name' => 'Crater', 'app_url' => 'http://localhost', 'timezone' => 'UTC',]] - Validation rules from GetSettingsRequest
Expected Result	- Validator passes and returns true.
Actual Result	- Validator passed and returned true.
Status	Pass
Severity	High

Test Case ID	GSR-004
Title	Missing settings array fails validation
Objective	Verify that input is invalid if the "settings" array is missing
Preconditions	- Application is running - GetSettingsRequest class is loaded
Test Steps	1. Instantiate a new GetSettingsRequest object. 2. Retrieve its validation rules. 3. Prepare input data without the "settings" key. 4. Validate the data using Laravel's Validator::make(). 5. Check if validation fails using fails(). 6. Retrieve and check the validation error for 'settings'.
Test Data	- Input data: [] - Expected error for 'settings'
Expected Result	- Validator fails and returns true. - Error message for 'settings': "The settings field is required."
Actual Result	- Validator failed and returned true. - Error message for 'settings': "The settings field is required."
Status	Pass
Severity	High

Test Case ID	GSR-005
Title	Null value for setting key fails validation
Objective	Ensure that validation fails when a value in "settings" is null
Preconditions	- Application is running - GetSettingsRequest class is loaded
Test Steps	1. Instantiate a new GetSettingsRequest object. 2. Retrieve its validation rules. 3. Prepare input data where one of the "settings" values is null. 4. Validate the data using Validator::make(). 5. Check if validation fails. 6. Retrieve and check the validation error for the null key.
Test Data	- Input data: ['settings' => ['app_name' => 'Crater', 'app_url' => null,]] - Expected error for 'settings.app_url'
Expected Result	- Validator fails and returns true. - Error message for 'settings.app_url': "The settings.app_url field is required."

Actual Result	- Validator failed and returned true. - Error message for 'settings.app_url': "The settings.app_url field is required."
Status	Pass
Severity	High

Test Case ID	GSR-006
Title	Non-string value for setting key fails validation
Objective	Ensure that validation fails when a value in "settings" is not a string
Preconditions	- Application is running - GetSettingsRequest class is loaded
Test Steps	1. Instantiate a new GetSettingsRequest object. 2. Retrieve its validation rules. 3. Prepare input data where one of the "settings" values is an integer. 4. Validate the data using Validator::make(). 5. Check if validation fails. 6. Retrieve and check the validation error for the invalid key.
Test Data	- Input data: ['settings' => ['app_name' => 'Crater', 'app_url' => 123,]] - Expected error for 'settings.app_url'
Expected Result	- Validator fails and returns true. - Error message for 'settings.app_url': "The settings.app_url must be a string."
Actual Result	- Validator failed and returned true. - Error message for 'settings.app_url': "The settings.app_url must be a string."
Status	Pass
Severity	High

Test Case ID	GSR-007
Title	Empty string value for setting key is invalid due to "required"
Objective	Ensure that validation fails when a value in "settings" is an empty string
Preconditions	- Application is running - GetSettingsRequest class is loaded
Test Steps	1. Instantiate a new GetSettingsRequest object. 2. Retrieve its validation rules. 3. Prepare input data where one of the "settings" values is an empty string. 4. Validate the data using Validator::make(). 5. Check if validation fails. 6. Retrieve and check the validation error for the key with an empty string.
Test Data	- Input data: ['settings' => ['app_name' => '', 'app_url' => 'http://localhost',]] - Expected error for 'settings.app_name'
Expected Result	- Validator fails and returns true. - Error message for 'settings.app_name': "The settings.app_name field is required."
Actual Result	- Validator failed and returned true. - Error message for 'settings.app_name': "The settings.app_name field is required."

Status	Pass
Severity	High

Test Case ID	GSR-008
Title	Validation ignores missing keys in settings array
Objective	Verify that validation does not fail when expected keys within "settings" are missing, as rules only apply to present keys
Preconditions	- Application is running - GetSettingsRequest class is loaded
Test Steps	1. Instantiate a new GetSettingsRequest object. 2. Retrieve its validation rules. 3. Prepare input data with only some keys present in "settings". 4. Validate the data using Validator::make(). 5. Check if validation passes.
Test Data	- Input data: ['settings' => ['app_name' => 'Crater', // 'app_url' key is not present]]
Expected Result	- Validator passes and returns true.
Actual Result	- Validator passed and returned true.
Status	Pass
Severity	High

File: GetSupportedCurrenciesController-Test.txt

Test Case ID	GSCC-001
Title	Successful Authorization and Supported Currencies Delegation
Objective	Verify that the controller successfully authorizes the user and delegates the request to retrieve supported currencies, responding with correct data and status.
Preconditions	- Application running - Database seeded with exchange rate providers - Necessary mock objects available - Valid user with permission to 'viewAny' exchange rate providers
Test Steps	1. Create a mock HTTP request object. 2. Prepare expected currencies data: ['USD', 'EUR', 'GBP']. 3. Mock the controller to partially implement inherited methods. 4. Set expectation for 'authorize' to be called once with arguments: 'viewAny', ExchangeRateProvider::class. 5. Set expectation for 'getSupportedCurrencies' to be called once with the mock request and to return a JsonResponse containing the expected currencies data. 6. Invoke the controller’s __invoke method with the mock request. 7. Verify that the response is an instance of JsonResponse. 8. Verify that the response data matches ['data' => ['USD', 'EUR', 'GBP']]. 9. Verify that the response status code is 200.
Test Data	- Input: Mock HTTP request - Expected currencies: ['USD', 'EUR', 'GBP'] - Expected response: JsonResponse(['data' => ['USD', 'EUR', 'GBP']], 200)
Expected Result	- The controller authorizes the request successfully. - The controller delegates to getSupportedCurrencies and returns a JsonResponse with the correct data. - The response is a JsonResponse instance. - The response contains: ['data' => ['USD', 'EUR', 'GBP']] - The status code is 200.
Actual Result	- The controller authorized the request and returned JsonResponse(['data' => ['USD', 'EUR', 'GBP']], 200) as expected.
Status	Pass
Severity	High

Test Case ID	GSCC-002
Title	Authorization Failure Throws AuthorizationException
Objective	Verify that the controller throws an AuthorizationException and does not delegate to getSupportedCurrencies when authorization fails.
Preconditions	- Application running - Database seeded with exchange rate providers - Necessary mock objects available - User lacking permission to 'viewAny' exchange rate providers
Test Steps	1. Create a mock HTTP request object. 2. Mock the controller to partially implement inherited methods. 3. Set expectation for 'authorize' to be called once with arguments: 'viewAny', ExchangeRateProvider::class, and to throw an AuthorizationException with message 'User not authorized.' 4. Set expectation that 'getSupportedCurrencies' should not be called. 5. Expect an AuthorizationException with message 'User not authorized.' when invoking the controller’s __invoke method with the mock request. 6. Invoke the controller’s __invoke method. 7. Verify that an AuthorizationException is thrown with the correct message. 8. Verify that 'getSupportedCurrencies' is not called.
Test Data	- Input: Mock HTTP request - Expected exception: AuthorizationException('User not authorized.')

Expected Result	- AuthorizationException is thrown when user is not authorized. - Exception message: 'User not authorized.' - 'getSupportedCurrencies' method is not called.
Actual Result	- AuthorizationException with message 'User not authorized.' was thrown, and getSupportedCurrencies was not called.
Status	Pass
Severity	High

File: Handler-Test.txt

Test Case ID	H-001
Title	Handler is in correct namespace
Objective	Verify that the Handler class is properly defined within the Crater\Exceptions namespace.
Preconditions	- Application running - Handler class is loaded
Test Steps	1. Reflect the Handler class using ReflectionClass. 2. Retrieve the namespace name via getNamespaceName(). 3. Assert that the returned namespace equals 'Crater\Exceptions'.
Test Data	- Handler class name: Crater\Exceptions\Handler - Expected namespace: 'Crater\Exceptions'
Expected Result	- Namespace name is 'Crater\Exceptions'
Actual Result	Namespace name is 'Crater\Exceptions'
Status	Pass
Severity	High

Test Case ID	H-002
Title	Handler extends ExceptionHandler
Objective	Ensure Handler inherits from Illuminate\Foundation\Exceptions\Handler.
Preconditions	- Application running - Handler and ExceptionHandler classes are loaded
Test Steps	1. Reflect the Handler class. 2. Retrieve the parent class using getParentClass(). 3. Check that a parent exists and its name matches 'Illuminate\Foundation\Exceptions\Handler'.
Test Data	- Handler class: Crater\Exceptions\Handler - Expected parent: Illuminate\Foundation\Exceptions\Handler
Expected Result	- Parent is not false - Parent class name is 'Illuminate\Foundation\Exceptions\Handler'
Actual Result	Parent is not false; parent class name is 'Illuminate\Foundation\Exceptions\Handler'
Status	Pass
Severity	High

Test Case ID	H-003
Title	Handler is not abstract
Objective	Confirm that Handler class is a concrete class and can be instantiated.
Preconditions	- Application running - Handler class is available
Test Steps	1. Reflect the Handler class. 2. Invoke isAbstract() method. 3. Assert that result is false.
Test Data	- Handler class: Crater\Exceptions\Handler
Expected Result	- isAbstract() returns false
Actual Result	isAbstract() returns false
Status	Pass
Severity	High

Test Case ID	H-004
Title	Handler is instantiable
Objective	Validate that Handler can be instantiated.
Preconditions	- Application running - Handler class is available
Test Steps	1. Reflect the Handler class. 2. Invoke isInstantiable(). 3. Assert that result is true.
Test Data	- Handler class: Crater\Exceptions\Handler
Expected Result	- isInstantiable() returns true
Actual Result	isInstantiable() returns true
Status	Pass
Severity	High

Test Case ID	H-005
Title	Handler has dontReport property
Objective	Verify that the Handler class contains a 'dontReport' property.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Invoke hasProperty('dontReport'). 3. Assert that property exists.
Test Data	- Property: dontReport
Expected Result	- hasProperty('dontReport') returns true
Actual Result	hasProperty('dontReport') returns true
Status	Pass
Severity	Medium

Test Case ID	H-006
Title	Handler has dontFlash property
Objective	Ensure that the Handler class contains a 'dontFlash' property.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Invoke hasProperty('dontFlash'). 3. Assert that property exists.
Test Data	- Property: dontFlash
Expected Result	- hasProperty('dontFlash') returns true
Actual Result	hasProperty('dontFlash') returns true
Status	Pass
Severity	Medium

Test Case ID	H-007
Title	dontReport property is protected
Objective	Confirm that the dontReport property has protected visibility.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Get the dontReport property. 3. Check that isProtected() returns true.

Test Data	- Property: dontReport
Expected Result	- Property is protected
Actual Result	Property is protected
Status	Pass
Severity	Medium

Test Case ID	H-008
Title	dontFlash property is protected
Objective	Confirm that the dontFlash property has protected visibility.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Get the dontFlash property. 3. Check that isProtected() returns true.
Test Data	- Property: dontFlash
Expected Result	- Property is protected
Actual Result	Property is protected
Status	Pass
Severity	Medium

Test Case ID	H-009
Title	dontReport default value is empty array
Objective	Ensure dontReport property defaults to an empty array.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Retrieve the dontReport property value on a new instance. 3. Assert the value is an array. 4. Assert the value is empty.
Test Data	- Property: dontReport
Expected Result	- Value is array - Value is empty
Actual Result	Value is array; value is empty
Status	Pass
Severity	Medium

Test Case ID	H-010
Title	dontFlash default value contains password fields
Objective	Validate that dontFlash property contains 'password' and 'password_confirmation' and has exactly two elements.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Retrieve the dontFlash property default value. 3. Assert that value is an array. 4. Assert array contains 'password'. 5. Assert array contains 'password_confirmation'. 6. Assert array has 2 elements.
Test Data	- Property: dontFlash

Expected Result	- Value is array - Contains 'password' - Contains 'password_confirmation' - Array has 2 elements
Actual Result	Value is array; contains 'password', 'password_confirmation'; has 2 elements
Status	Pass
Severity	High

Test Case ID	H-011
Title	Handler has report method
Objective	Confirm existence of 'report' method in Handler class.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Invoke hasMethod('report'). 3. Assert that method exists.
Test Data	- Method: report
Expected Result	- hasMethod('report') returns true
Actual Result	hasMethod('report') returns true
Status	Pass
Severity	High

Test Case ID	H-012
Title	Handler has render method
Objective	Confirm existence of 'render' method in Handler class.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Invoke hasMethod('render'). 3. Assert that method exists.
Test Data	- Method: render
Expected Result	- hasMethod('render') returns true
Actual Result	hasMethod('render') returns true
Status	Pass
Severity	High

Test Case ID	H-013
Title	report method is public
Objective	Validate that the 'report' method has public visibility.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Get the report method. 3. Assert isPublic() returns true.
Test Data	- Method: report
Expected Result	- report method is public
Actual Result	report method is public
Status	Pass
Severity	High

Test Case ID	H-014
Title	render method is public
Objective	Validate that the 'render' method has public visibility.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Get the render method. 3. Assert isPublic() returns true.
Test Data	- Method: render
Expected Result	- render method is public
Actual Result	render method is public
Status	Pass
Severity	High

Test Case ID	H-015
Title	report method is not static
Objective	Ensure the 'report' method is not static.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Get the report method. 3. Assert isStatic() returns false.
Test Data	- Method: report
Expected Result	- report method is not static
Actual Result	report method is not static
Status	Pass
Severity	Medium

Test Case ID	H-016
Title	render method is not static
Objective	Ensure the 'render' method is not static.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Get the render method. 3. Assert isStatic() returns false.
Test Data	- Method: render
Expected Result	- render method is not static
Actual Result	render method is not static
Status	Pass
Severity	Medium

Test Case ID	H-017
Title	report method accepts exception parameter
Objective	Confirm that the report method accepts one parameter named 'exception'.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Get parameters of the report method. 3. Assert parameter count is 1. 4. Assert parameter name is 'exception'.

Test Data	- Method: report - Parameter name: exception
Expected Result	- Parameter count is 1 - Parameter name is 'exception'
Actual Result	Parameter count is 1; parameter name is 'exception'
Status	Pass
Severity	High

Test Case ID	H-018
Title	render method accepts two parameters
Objective	Validate that render method accepts 'request' and 'exception' parameters.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Get parameters of the render method. 3. Assert parameter count is 2. 4. Assert parameter names are 'request' and 'exception'.
Test Data	- Method: render - Parameters: request, exception
Expected Result	- Parameter count is 2 - First parameter is 'request' - Second parameter is 'exception'
Actual Result	Parameter count is 2; parameters are 'request' and 'exception'
Status	Pass
Severity	High

Test Case ID	H-019
Title	Handler is not final
Objective	Ensure Handler class is not marked as final.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Invoke isFinal(). 3. Assert that result is false.
Test Data	- Handler class
Expected Result	- isFinal() returns false
Actual Result	isFinal() returns false
Status	Pass
Severity	Medium

Test Case ID	H-020
Title	Handler is not an interface
Objective	Confirm that Handler is a class, not an interface.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Invoke isInterface(). 3. Assert that result is false.
Test Data	- Handler class
Expected Result	- isInterface() returns false
Actual Result	isInterface() returns false

Status	Pass
Severity	Medium

Test Case ID	H-021
Title	Handler is not a trait
Objective	Confirm that Handler is not implemented as a trait.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Invoke isTrait(). 3. Assert that result is false.
Test Data	- Handler class
Expected Result	- isTrait() returns false
Actual Result	isTrait() returns false
Status	Pass
Severity	Medium

Test Case ID	H-022
Title	Handler class is loaded
Objective	Ensure Handler class can be autoloaded.
Preconditions	- Application running
Test Steps	1. Call class_exists on Handler::class. 2. Assert that result is true.
Test Data	- Class name: Crater\Exceptions\Handler
Expected Result	- class_exists returns true
Actual Result	class_exists returns true
Status	Pass
Severity	High

Test Case ID	H-023
Title	Handler uses ExceptionHandler
Objective	Verify that Handler imports Illuminate\Foundation\Exceptions\Handler as ExceptionHandler.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Read the file contents of the Handler class. 3. Check for the import statement for ExceptionHandler.
Test Data	- File import: use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler
Expected Result	- File contains ExceptionHandler import statement
Actual Result	File contains ExceptionHandler import statement
Status	Pass
Severity	High

Test Case ID	H-024
Title	Handler uses Throwable
Objective	Verify that Handler imports Throwable.
Preconditions	- Application running

Test Steps	1. Reflect the Handler class. 2. Read the file contents of the Handler class. 3. Check for 'use Throwable' statement.
Test Data	- File import: use Throwable
Expected Result	- File contains 'use Throwable'
Actual Result	File contains 'use Throwable'
Status	Pass
Severity	High

Test Case ID	H-025
Title	Handler file has expected structure
Objective	Confirm Handler file contains expected class structure and required members.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Read the Handler file contents. 3. Verify it contains: a. 'class Handler extends ExceptionHandler' b. 'protected $dontReport' c. 'protected $dontFlash' d. 'public function report' e. 'public function render'
Test Data	- File contents
Expected Result	- File contains all specified structural elements
Actual Result	File contains all specified structural elements
Status	Pass
Severity	High

Test Case ID	H-026
Title	Handler has reasonable line count
Objective	Ensure Handler file has a maintainable line count between 30 and 100.
Preconditions	- Application running
Test Steps	1. Reflect the Handler class. 2. Read file contents and count lines. 3. Assert line count > 30 and < 100.
Test Data	- File contents and line count
Expected Result	- Line count is greater than 30, less than 100
Actual Result	Line count greater than 30, less than 100
Status	Pass
Severity	Low

Test Case ID	H-027
Title	report method calls parent report
Objective	Confirm that Handler's report method calls parent::report.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Read Handler file contents. 3. Check for 'parent::report($exception)' call.
Test Data	- Source statement: parent::report($exception)
Expected Result	- Source file contains 'parent::report($exception)'

Actual Result	Source file contains 'parent::report($exception)'
Status	Pass
Severity	High

Test Case ID	H-028
Title	render method calls parent render
Objective	Validate that Handler's render method calls parent::render.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Read Handler file contents. 3. Check for 'parent::render($request, $exception)'.
Test Data	- Source statement: parent::render($request, $exception)
Expected Result	- Source file contains 'parent::render($request, $exception)'
Actual Result	Source file contains 'parent::render($request, $exception)'
Status	Pass
Severity	High

Test Case ID	H-029
Title	render method returns response
Objective	Ensure Handler's render method returns result from parent::render.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Read Handler file contents. 3. Look for 'return parent::render' statement.
Test Data	- Source statement: return parent::render
Expected Result	- Source file contains 'return parent::render'
Actual Result	Source file contains 'return parent::render'
Status	Pass
Severity	High

Test Case ID	H-030
Title	dontReport property has documentation
Objective	Confirm dontReport property is documented with a docblock.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Get dontReport property. 3. Retrieve doc comment with getDocComment(). 4. Assert doc comment is not false.
Test Data	- Property: dontReport
Expected Result	- doc comment exists and is not false
Actual Result	doc comment exists and is not false
Status	Pass
Severity	Low

Test Case ID	H-031
Title	dontFlash property has documentation
Objective	Confirm dontFlash property is documented with a docblock.

Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Get dontFlash property. 3. Retrieve doc comment with getDocComment(). 4. Assert doc comment is not false.
Test Data	- Property: dontFlash
Expected Result	- doc comment exists and is not false
Actual Result	doc comment exists and is not false
Status	Pass
Severity	Low

Test Case ID	H-032
Title	report method has documentation
Objective	Verify that report method is documented.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Retrieve report method. 3. Get doc comment. 4. Assert doc comment is not false.
Test Data	- Method: report
Expected Result	- doc comment exists and is not false
Actual Result	doc comment exists and is not false
Status	Pass
Severity	Low

Test Case ID	H-033
Title	render method has documentation
Objective	Verify that render method is documented.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Retrieve render method. 3. Get doc comment. 4. Assert doc comment is not false.
Test Data	- Method: render
Expected Result	- doc comment exists and is not false
Actual Result	doc comment exists and is not false
Status	Pass
Severity	Low

Test Case ID	H-034
Title	report method documentation mentions Sentry and Bugsnag
Objective	Validate the report method documentation mentions Sentry and Bugsnag integration.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Get report method doc comment. 3. Assert doc comment contains 'Sentry'. 4. Assert doc comment contains 'Bugsnag'.
Test Data	- Documentation terms: 'Sentry', 'Bugsnag'

Expected Result	- Doc comment contains both terms
Actual Result	Doc comment contains 'Sentry' and 'Bugsnag'
Status	Pass
Severity	Low

Test Case ID	H-035
Title	dontFlash first element is password
Objective	Ensure first element of dontFlash is 'password'.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Access dontFlash property value. 3. Assert the first element is 'password'.
Test Data	- dontFlash: ['password', ...]
Expected Result	- First element is 'password'
Actual Result	First element is 'password'
Status	Pass
Severity	High

Test Case ID	H-036
Title	dontFlash second element is password_confirmation
Objective	Ensure second element of dontFlash is 'password_confirmation'.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Access dontFlash property value. 3. Assert the second element is 'password_confirmation'.
Test Data	- dontFlash: [..., 'password_confirmation']
Expected Result	- Second element is 'password_confirmation'
Actual Result	Second element is 'password_confirmation'
Status	Pass
Severity	High

Test Case ID	H-037
Title	Handler file declares namespace correctly
Objective	Verify the Handler file declares 'namespace Crater\Exceptions'.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Read file contents. 3. Assert content includes 'namespace Crater\Exceptions'.
Test Data	- Namespace declaration
Expected Result	- File contains 'namespace Crater\Exceptions'
Actual Result	File contains 'namespace Crater\Exceptions'
Status	Pass
Severity	High

Test Case ID	H-038
Title	Handler file is concise

Objective	Ensure the Handler file does not exceed 2000 characters in length.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Read file contents. 3. Assert length is less than 2000 characters.
Test Data	- File content length
Expected Result	- Length is less than 2000 characters
Actual Result	Length is less than 2000 characters
Status	Pass
Severity	Low

Test Case ID	H-039
Title	Handler has exactly 2 public methods
Objective	Validate that Handler defines exactly two public methods.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Get all public methods declared in Handler. 3. Filter own methods. 4. Assert count equals 2.
Test Data	- Public methods: report, render
Expected Result	- Method count is 2
Actual Result	Method count is 2
Status	Pass
Severity	High

Test Case ID	H-040
Title	Handler has exactly 2 protected properties
Objective	Validate Handler class defines only two protected properties.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Get all protected properties declared in Handler. 3. Filter own properties. 4. Assert count equals 2.
Test Data	- Protected properties: dontReport, dontFlash
Expected Result	- Property count is 2
Actual Result	Property count is 2
Status	Pass
Severity	High

Test Case ID	H-041
Title	report method delegates to parent
Objective	Confirm the report method is non-abstract and implemented in Handler.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Get report method. 3. Assert method is not abstract.
Test Data	- Method: report
Expected Result	- report method is not abstract

Actual Result	report method is not abstract
Status	Pass
Severity	High

Test Case ID	H-042
Title	render method delegates to parent
Objective	Confirm the render method is non-abstract and implemented in Handler.
Preconditions	- Application running
Test Steps	1. Reflect Handler class. 2. Get render method. 3. Assert method is not abstract.
Test Data	- Method: render
Expected Result	- render method is not abstract
Actual Result	render method is not abstract
Status	Pass
Severity	High

File: InstallModuleCommand-Test.txt

Test Case ID	IMC-001
Title	Verify command signature and description for InstallModuleCommand
Objective	Ensure that the InstallModuleCommand has the correct name, description, and signature property.
Preconditions	- Application running - All dependencies are installed - Crater module autoloaded - Database not required for this test
Test Steps	1. Instantiate the InstallModuleCommand class. 2. Retrieve the command name using getName(). 3. Retrieve the command description using getDescription(). 4. Access the private "signature" property via reflection.
Test Data	- Command class: \Crater\Console\Commands\InstallModuleCommand - Expected getName(): 'install:module' - Expected getDescription(): 'Install cloned module.' - Expected signature property: 'install:module {module} {version}'
Expected Result	- Command name equals 'install:module' - Command description equals 'Install cloned module.' - Signature property equals 'install:module {module} {version}'
Actual Result	Command name, description, and signature property matched expected values.
Status	Pass
Severity	Medium

Test Case ID	IMC-002
Title	Verify InstallModuleCommand constructor instantiation
Objective	Verify that InstallModuleCommand can be instantiated successfully and is an instance of expected classes.
Preconditions	- Application running - All dependencies are installed - Crater module autoloaded
Test Steps	1. Instantiate the InstallModuleCommand class. 2. Check if the created object is an instance of InstallModuleCommand. 3. Check if the created object is also an instance of Illuminate\Console\Command.
Test Data	- Command class: \Crater\Console\Commands\InstallModuleCommand
Expected Result	- Object is instance of InstallModuleCommand. - Object is instance of Illuminate\Console\Command.
Actual Result	Object was successfully instantiated and confirmed as instance of both classes.
Status	Pass
Severity	Medium

Test Case ID	IMC-003
Title	Verify handle method calls ModuleInstaller::complete with provided arguments and returns success
Objective	Ensure that the handle() method correctly passes arguments to ModuleInstaller::complete and returns success status.

Preconditions	- Application running - All dependencies are installed - Crater module autoloaded - Mocking system (Mockery) available
Test Steps	1. Mock ModuleInstaller::complete to expect arguments 'test-module' and '1.0.0'. 2. Mock InstallModuleCommand to return 'test-module' and '1.0.0' as arguments for 'module' and 'version'. 3. Invoke handle() on the command. 4. Assert that returned value is Command::SUCCESS.
Test Data	- module: 'test-module' - version: '1.0.0' - Expected handle() result: Command::SUCCESS
Expected Result	- ModuleInstaller::complete called with 'test-module' and '1.0.0'. - handle() returns Command::SUCCESS.
Actual Result	ModuleInstaller::complete was called with correct arguments and handle() returned Command::SUCCESS.
Status	Pass
Severity	High

Test Case ID	IMC-004
Title	Verify handle method calls ModuleInstaller::complete with empty arguments and returns success
Objective	Ensure that the handle() method passes empty values to ModuleInstaller::complete when arguments are empty, and still returns success.
Preconditions	- Application running - All dependencies are installed - Crater module autoloaded - Mocking system (Mockery) available
Test Steps	1. Mock ModuleInstaller::complete to expect empty arguments '' and ''. 2. Mock InstallModuleCommand to return '' for both 'module' and 'version'. 3. Invoke handle() on the command. 4. Assert that returned value is Command::SUCCESS.
Test Data	- module: '' - version: '' - Expected handle() result: Command::SUCCESS
Expected Result	- ModuleInstaller::complete called with empty arguments. - handle() returns Command::SUCCESS.
Actual Result	ModuleInstaller::complete was called with empty arguments and handle() returned Command::SUCCESS.
Status	Pass
Severity	High

Test Case ID	IMC-005
Title	Verify handle method propagates exceptions from ModuleInstaller::complete
Objective	Ensure the handle() method throws an exception when ModuleInstaller::complete raises one.
Preconditions	- Application running - All dependencies are installed - Crater module autoloaded - Mocking system (Mockery) available

Test Steps	1. Mock ModuleInstaller::complete to throw exception with message 'Module installation failed!' when called with 'error-module' and '1.0.0'. 2. Mock InstallModuleCommand to return 'error-module' for 'module' and '1.0.0' for 'version'. 3. Invoke handle() on the command and assert it throws the expected exception.
Test Data	- module: 'error-module' - version: '1.0.0' - Exception message: 'Module installation failed!' - Expected handle() to throw \Exception with correct message
Expected Result	- handle() throws an Exception with message 'Module installation failed!'
Actual Result	handle() threw an Exception with expected message as required.
Status	Pass
Severity	High

File: InstallationMiddleware-Test.txt

Test Case ID	IM-001
Title	InstallationMiddleware class exists and is instantiable
Objective	Verify the InstallationMiddleware class exists and can be instantiated.
Preconditions	- Application running - All dependencies installed - Database seeded
Test Steps	1. Attempt to instantiate the InstallationMiddleware class. 2. Check if the instantiated object is an instance of InstallationMiddleware.
Test Data	- Instantiation attempt of InstallationMiddleware - Expected class: InstallationMiddleware
Expected Result	- The object created is an instance of InstallationMiddleware.
Actual Result	The object is successfully instantiated and confirmed as an InstallationMiddleware instance.
Status	Pass
Severity	High

Test Case ID	IM-002
Title	InstallationMiddleware is in correct namespace
Objective	Verify that the InstallationMiddleware is under the "Crater\Http\Middleware" namespace.
Preconditions	- Application running - InstallationMiddleware class present in codebase
Test Steps	1. Reflect on InstallationMiddleware class using ReflectionClass. 2. Retrieve the namespace name of the class.
Test Data	- InstallationMiddleware class with declared namespace
Expected Result	- Class is within the 'Crater\Http\Middleware' namespace.
Actual Result	Namespace of InstallationMiddleware is confirmed as 'Crater\Http\Middleware'.
Status	Pass
Severity	Medium

Test Case ID	IM-003
Title	InstallationMiddleware has handle method
Objective	Ensure that InstallationMiddleware class defines a "handle" method.
Preconditions	- Application running - InstallationMiddleware class present
Test Steps	1. Use ReflectionClass to inspect InstallationMiddleware. 2. Check if the class contains a "handle" method.
Test Data	- InstallationMiddleware class definition
Expected Result	- "handle" method is present in InstallationMiddleware class.
Actual Result	"handle" method successfully found in InstallationMiddleware class.
Status	Pass
Severity	High

Test Case ID	IM-004
Title	handle method accepts request and Closure parameters

Objective	Validate "handle" method parameter names and count.
Preconditions	- Application running - InstallationMiddleware class present
Test Steps	1. Use ReflectionClass to retrieve the "handle" method. 2. Get parameters of the method. 3. Verify there are two parameters. 4. Confirm first parameter is named "request". 5. Confirm second parameter is named "next".
Test Data	- Method signature of "handle" in InstallationMiddleware
Expected Result	- "handle" method has exactly two parameters: "request" and "next".
Actual Result	"handle" method is confirmed to have parameters "request" and "next".
Status	Pass
Severity	High

Test Case ID	IM-005
Title	handle method is public
Objective	Ensure the "handle" method is public and not static.
Preconditions	- Application running - InstallationMiddleware class present
Test Steps	1. Use ReflectionClass to inspect "handle" method. 2. Check if the "handle" method is public. 3. Check if the "handle" method is not static.
Test Data	- "handle" method definition
Expected Result	- "handle" method is public. - "handle" method is not static.
Actual Result	"handle" method is public and not static.
Status	Pass
Severity	High

Test Case ID	IM-006
Title	InstallationMiddleware checks database_created file
Objective	Ensure InstallationMiddleware checks the 'database_created' file via Storage API.
Preconditions	- Application running - Storage API accessible - InstallationMiddleware class present
Test Steps	1. Retrieve file contents of InstallationMiddleware class. 2. Check for usage of "Storage::disk('local')->has('database_created')" within the file.
Test Data	- InstallationMiddleware file contents
Expected Result	- File contains "Storage::disk('local')->has('database_created')" check.
Actual Result	File contains check for "Storage::disk('local')->has('database_created')".
Status	Pass
Severity	High

Test Case ID	IM-007
Title	InstallationMiddleware redirects to installation route
Objective	Confirm InstallationMiddleware includes redirection to '/installation' when needed.

Preconditions	- Application running - Routing available - InstallationMiddleware class present
Test Steps	1. Retrieve file contents of InstallationMiddleware class. 2. Search for "redirect('/installation')" within the file.
Test Data	- InstallationMiddleware file contents
Expected Result	- File contains "redirect('/installation')" statement for failed installation.
Actual Result	"redirect('/installation')" found in InstallationMiddleware file.
Status	Pass
Severity	High

Test Case ID	IM-008
Title	InstallationMiddleware checks profile_complete setting
Objective	Verify InstallationMiddleware checks "profile_complete" setting for completion status.
Preconditions	- Application running - Settings system ready - InstallationMiddleware class present
Test Steps	1. Retrieve file contents of InstallationMiddleware class. 2. Confirm presence of "Setting::getSetting('profile_complete')". 3. Confirm handling of "COMPLETED" value.
Test Data	- InstallationMiddleware file contents
Expected Result	- File contains "Setting::getSetting('profile_complete')". - File contains logic handling "COMPLETED" value.
Actual Result	Both "Setting::getSetting('profile_complete')" and "COMPLETED" handling found.
Status	Pass
Severity	High

Test Case ID	IM-009
Title	InstallationMiddleware calls next middleware when conditions met
Objective	Ensure InstallationMiddleware correctly calls the next middleware when all checks pass.
Preconditions	- Application running - Middleware chain enabled - InstallationMiddleware class present
Test Steps	1. Retrieve file contents of InstallationMiddleware class. 2. Search for the statement "return $next($request)".
Test Data	- InstallationMiddleware file contents
Expected Result	- File contains "return $next($request)" to continue middleware chain.
Actual Result	"return $next($request)" found in InstallationMiddleware file.
Status	Pass
Severity	High

Test Case ID	IM-010
Title	InstallationMiddleware uses Setting model and Closure
Objective	Confirm InstallationMiddleware imports both Setting model and Closure in its definition.

Preconditions	- Application running - Setting model available - InstallationMiddleware class present
Test Steps	1. Retrieve file contents of InstallationMiddleware class. 2. Search for "use Crater\Models\Setting". 3. Search for "use Closure".
Test Data	- InstallationMiddleware file contents
Expected Result	- "use Crater\Models\Setting" is present at top of file. - "use Closure" is present at top of file.
Actual Result	Both "use Crater\Models\Setting" and "use Closure" found in file.
Status	Pass
Severity	High

File: Invoice-Test.txt

Test Case ID	I-001
Title	Invoice model exists and is instantiable
Objective	Verify that the Invoice model class can be instantiated successfully.
Preconditions	- Application running - Database seeded - Invoice model class is available
Test Steps	1. Attempt to instantiate a new Invoice object. 2. Verify that the created object is an instance of the Invoice class.
Test Data	- Class: Crater\Models\Invoice
Expected Result	- An object is created and is an instance of Invoice.
Actual Result	An Invoice object was successfully instantiated and verified as an instance of Invoice.
Status	Pass
Severity	High

Test Case ID	I-002
Title	Invoice has all relationship methods
Objective	Ensure that the Invoice model contains all required relationship methods.
Preconditions	- Application running - Database seeded - Invoice model class loaded
Test Steps	1. Create an instance of the Invoice model. 2. Check for the existence of the following methods: - transactions - items - taxes - payments - currency - company - customer - creator
Test Data	- Invoice object
Expected Result	- All listed methods exist in the Invoice model.
Actual Result	All relationship methods were found on the Invoice model as expected.
Status	Pass
Severity	High

Test Case ID	I-003
Title	Invoice has status constants
Objective	Verify the presence of all status constants required by business logic.
Preconditions	- Application running - Invoice model class loaded
Test Steps	1. Check for the definition of the following constants in Invoice: - STATUS_DRAFT - STATUS_SENT - STATUS_VIEWED - STATUS_COMPLETED - STATUS_UNPAID - STATUS_PAID - STATUS_PARTIALLY_PAID

Test Data	- Invoice class constants
Expected Result	- All listed constants are defined in the Invoice class.
Actual Result	All status constants were defined and available on the Invoice class.
Status	Pass
Severity	High

Test Case ID	I-004
Title	Invoice has scope methods for filtering
Objective	Confirm that the Invoice model contains all necessary scope methods for filtering.
Preconditions	- Application running - Invoice model class loaded
Test Steps	1. Use reflection to check for the existence of the following methods in Invoice: - scopeWhereStatus - scopeWherePaidStatus - scopeWhereSearch - scopeApplyFilters - scopePaginateData
Test Data	- Invoice class methods
Expected Result	- All listed scope methods exist in the Invoice model.
Actual Result	All scope methods for filtering are present in the Invoice model.
Status	Pass
Severity	High

Test Case ID	I-005
Title	Invoice has static methods for CRUD operations
Objective	Validate that the Invoice model provides static methods for create and delete operations.
Preconditions	- Application running - Invoice model class loaded
Test Steps	1. Use reflection to find if Invoice has a static createInvoice method. 2. Use reflection to find if Invoice has a static deleteInvoices method.
Test Data	- Method names: createInvoice, deleteInvoices
Expected Result	- createInvoice exists and is static. - deleteInvoices exists and is static.
Actual Result	Both createInvoice and deleteInvoices methods are present and are static.
Status	Pass
Severity	High

Test Case ID	I-006
Title	Invoice has accessor methods for formatted attributes
Objective	Check that the Invoice model contains accessor methods for formatted attribute outputs.
Preconditions	- Application running - Invoice model class loaded
Test Steps	1. Use reflection to check for the following accessor methods in Invoice: - getFormattedCreatedAtAttribute - getFormattedDueDateAttribute - getFormattedInvoiceDateAttribute - getInvoicePdfUrlAttribute

Test Data	- Method names checked via reflection
Expected Result	- All listed accessor methods exist in the Invoice model.
Actual Result	Accessor methods for formatted attributes were all found as expected.
Status	Pass
Severity	Medium

Test Case ID	I-007
Title	Invoice uses required traits
Objective	Confirm that the Invoice model uses the necessary traits for extended functionality.
Preconditions	- Application running - Invoice model class loaded
Test Steps	1. Use reflection to retrieve all trait names the Invoice model uses. 2. Verify that the model uses: - Crater\Traits\GeneratesPdfTrait - Crater\Traits\HasCustomFieldsTrait
Test Data	- Trait names in Invoice model
Expected Result	- Both trait names are present in the traits list of Invoice.
Actual Result	Invoice model uses both GeneratesPdfTrait and HasCustomFieldsTrait as required.
Status	Pass
Severity	Medium

Test Case ID	I-008
Title	Invoice has business logic methods
Objective	Ensure the Invoice model contains all required business logic methods.
Preconditions	- Application running - Invoice model class loaded
Test Steps	1. Use reflection to check for the following methods in Invoice: - getPreviousStatus - addInvoicePayment - subtractInvoicePayment - changeInvoiceStatus
Test Data	- Method names verified via reflection
Expected Result	- All listed business logic methods exist in the Invoice model.
Actual Result	Business logic methods are present as intended in the Invoice model.
Status	Pass
Severity	High

Test Case ID	I-009
Title	Invoice model is in correct namespace
Objective	Verify that the Invoice model resides in the expected namespace.
Preconditions	- Application running - Autoloader configured
Test Steps	1. Use reflection to get the namespace name of the Invoice model. 2. Confirm the namespace is 'Crater\Models'.
Test Data	- Class: Invoice
Expected Result	- Namespace retrieved is 'Crater\Models'.
Actual Result	Invoice model's namespace was correctly identified as 'Crater\Models'.

Status	Pass
Severity	Low

Test Case ID	I-010
Title	Invoice extends Eloquent Model
Objective	Validate that the Invoice model extends the base Eloquent Model class.
Preconditions	- Application running - Laravel framework loaded
Test Steps	1. Instantiate an Invoice object. 2. Verify that the object is an instance of \Illuminate\Database\Eloquent\Model.
Test Data	- Invoice object
Expected Result	- Invoice is an instance of Eloquent Model.
Actual Result	Invoice was verified as an instance of \Illuminate\Database\Eloquent\Model.
Status	Pass
Severity	High

File: InvoiceCollection-Test.txt

Test Case ID	IC-001
Title	InvoiceCollection instantiation
Objective	Verify that InvoiceCollection can be instantiated without errors.
Preconditions	- Application running - InvoiceCollection class available - Illuminate\Support\Collection available
Test Steps	1. Instantiate a new InvoiceCollection object with an empty Collection.
Test Data	- Input: new Collection([]) - Class: InvoiceCollection
Expected Result	- The instantiated object is an instance of InvoiceCollection.
Actual Result	Object is successfully instantiated; instance of InvoiceCollection confirmed.
Status	Pass
Severity	Medium

Test Case ID	IC-002
Title	InvoiceCollection inheritance from ResourceCollection
Objective	Confirm that InvoiceCollection extends ResourceCollection.
Preconditions	- Application running - InvoiceCollection and ResourceCollection classes available
Test Steps	1. Instantiate a new InvoiceCollection with an empty Collection. 2. Check if the instance is of type \Illuminate\Http\Resources\Json\ResourceCollection.
Test Data	- Input: new Collection([]) - Class: InvoiceCollection
Expected Result	- InvoiceCollection instance is also an instance of ResourceCollection.
Actual Result	InvoiceCollection instance confirmed as a ResourceCollection subclass.
Status	Pass
Severity	Medium

Test Case ID	IC-003
Title	InvoiceCollection namespace validation
Objective	Ensure InvoiceCollection is in the correct namespace.
Preconditions	- Application running - InvoiceCollection class available
Test Steps	1. Create a ReflectionClass for InvoiceCollection. 2. Retrieve the namespace via getNamespaceName(). 3. Verify the namespace is 'Crater\Http\Resources'.
Test Data	- Class: InvoiceCollection
Expected Result	- The namespace returned is 'Crater\Http\Resources'.
Actual Result	Namespace returned is 'Crater\Http\Resources'.
Status	Pass
Severity	Low

Test Case ID	IC-004
Title	InvoiceCollection has public toArray method
Objective	Verify the existence of the toArray method in InvoiceCollection.

Preconditions	- Application running - InvoiceCollection class available
Test Steps	1. Instantiate InvoiceCollection with an empty Collection. 2. Check if the method 'toArray' exists on the instance.
Test Data	- Input: new Collection([]) - Method: toArray
Expected Result	- The method toArray exists on InvoiceCollection.
Actual Result	Method toArray confirmed to exist.
Status	Pass
Severity	Medium

Test Case ID	IC-005
Title	toArray returns empty array for empty collection
Objective	Ensure toArray returns an empty array when the collection is empty.
Preconditions	- Application running - InvoiceCollection class and Request available
Test Steps	1. Instantiate a Request object. 2. Instantiate InvoiceCollection with an empty Collection. 3. Call toArray($request) on the InvoiceCollection. 4. Validate result is an array. 5. Validate result is empty.
Test Data	- Collection: [] - Request: new Request() - Method Call: toArray($request)
Expected Result	- Result is an array. - Result is empty.
Actual Result	toArray returns an empty array as expected when the collection is empty.
Status	Pass
Severity	Medium

Test Case ID	IC-006
Title	toArray delegates to parent ResourceCollection
Objective	Verify toArray implementation uses parent::toArray.
Preconditions	- Application running - InvoiceCollection class available
Test Steps	1. Create ReflectionClass for InvoiceCollection. 2. Read the source file contents. 3. Check for the inclusion of 'parent::toArray' in the file.
Test Data	- Class: InvoiceCollection - Source file content
Expected Result	- 'parent::toArray' is present in the source file, indicating delegation.
Actual Result	Source file contains 'parent::toArray' confirming delegation.
Status	Pass
Severity	Medium

Test Case ID	IC-007
Title	toArray method is public and non-static
Objective	Ensure toArray is a public, non-static method.
Preconditions	- Application running - InvoiceCollection class available

Test Steps	1. Create ReflectionClass for InvoiceCollection. 2. Get the 'toArray' method via getMethod(). 3. Check method visibility. 4. Check if method is static.
Test Data	- Class: InvoiceCollection - Method: toArray
Expected Result	- toArray is public. - toArray is not static.
Actual Result	toArray method verified as public and non-static.
Status	Pass
Severity	Medium

Test Case ID	IC-008
Title	toArray accepts request parameter
Objective	Confirm toArray method accepts a 'request' parameter.
Preconditions	- Application running - InvoiceCollection class available
Test Steps	1. Create ReflectionClass for InvoiceCollection. 2. Retrieve the toArray method parameters. 3. Verify there is exactly one parameter named 'request'.
Test Data	- Method: toArray - Expected Parameter: request
Expected Result	- toArray has one parameter named 'request'.
Actual Result	toArray method confirmed to accept 'request' parameter.
Status	Pass
Severity	Medium

Test Case ID	IC-009
Title	InvoiceCollection file is concise
Objective	Validate InvoiceCollection source file is under 1000 bytes in size.
Preconditions	- Application running - InvoiceCollection class available
Test Steps	1. Create ReflectionClass for InvoiceCollection. 2. Get the source file content. 3. Check if file size (in bytes) is less than 1000.
Test Data	- Source file content
Expected Result	- File size is less than 1000 bytes.
Actual Result	InvoiceCollection file size measured as less than 1000 bytes.
Status	Pass
Severity	Low

Test Case ID	IC-010
Title	InvoiceCollection minimal line count
Objective	Ensure InvoiceCollection source file has fewer than 30 lines.
Preconditions	- Application running - InvoiceCollection class available
Test Steps	1. Create ReflectionClass for InvoiceCollection. 2. Get the source file content. 3. Split content by new lines. 4. Verify the line count is less than 30.

Test Data	- Source file content - Line count
Expected Result	- Source file line count is less than 30.
Actual Result	InvoiceCollection source file line count is less than 30.
Status	Pass
Severity	Low

File: InvoiceItem-Test.txt

Test Case ID	II-001
Title	Verify InvoiceItem model existence and relationships
Objective	Ensure the InvoiceItem model is present and implements required relationships.
Preconditions	- Application running - Database seeded - InvoiceItem model loaded
Test Steps	1. Instantiate a new InvoiceItem object. 2. Verify object is instance of InvoiceItem class. 3. Check for existence of 'invoice', 'item', 'taxes', and 'recurringInvoice' relationship methods.
Test Data	- InvoiceItem class instance
Expected Result	- InvoiceItem object is created successfully. - 'invoice', 'item', 'taxes', and 'recurringInvoice' relationship methods exist in InvoiceItem.
Actual Result	InvoiceItem object instantiated. All required relationships exist.
Status	Pass
Severity	High

Test Case ID	II-002
Title	Verify InvoiceItem scope methods and casts
Objective	Ensure scope filtering methods and attribute casts exist in the InvoiceItem model.
Preconditions	- Application running - InvoiceItem model accessible
Test Steps	1. Get Reflection of InvoiceItem class. 2. Check the existence of scopeWhereCompany, scopeInvoicesBetween, scopeApplyInvoiceFilters, and scopeItemAttributes methods. 3. Instantiate InvoiceItem and retrieve $casts property. 4. Validate that 'price', 'total', and 'quantity' are present in casts array.
Test Data	- InvoiceItem class reflection - InvoiceItem class instance
Expected Result	- All specified scope methods exist. - 'price', 'total', and 'quantity' keys present in casts property.
Actual Result	All scope methods found. Casts contain required keys.
Status	Pass
Severity	High

Test Case ID	II-003
Title	Verify InvoiceItem uses HasCustomFieldsTrait
Objective	Ensure the HasCustomFieldsTrait is used by the InvoiceItem model.
Preconditions	- Application running - InvoiceItem model available
Test Steps	1. Get Reflection of InvoiceItem class. 2. Retrieve all trait names used by InvoiceItem. 3. Check that 'Crater\Traits\HasCustomFieldsTrait' is among the trait names.
Test Data	- InvoiceItem class reflection
Expected Result	- HasCustomFieldsTrait is present in trait names.
Actual Result	HasCustomFieldsTrait found in trait list.

Status	Pass
Severity	High

Test Case ID	II-004
Title	Verify InvoiceItemResource returns required fields in toArray
Objective	Ensure toArray() of InvoiceItemResource includes required fields.
Preconditions	- Application running - InvoiceItemResource class available
Test Steps	1. Get Reflection of InvoiceItemResource class. 2. Read InvoiceItemResource file contents. 3. Check toArray returns 'id', 'name', 'price', 'quantity', and 'total'.
Test Data	- InvoiceItemResource class file content
Expected Result	- toArray returns keys: 'id', 'name', 'price', 'quantity', 'total'.
Actual Result	toArray function returns all required keys.
Status	Pass
Severity	High

Test Case ID	II-005
Title	Verify InvoiceItemResource extends JsonResource
Objective	Confirm that InvoiceItemResource class inherits JsonResource.
Preconditions	- Application running - InvoiceItemResource class available
Test Steps	1. Instantiate InvoiceItemResource with mock data. 2. Verify instance is of type JsonResource.
Test Data	- Input: (object)['id' => 1]
Expected Result	- InvoiceItemResource is instance of Illuminate\Http\Resources\Json\JsonResource.
Actual Result	InvoiceItemResource instantiated and is a JsonResource.
Status	Pass
Severity	High

Test Case ID	II-006
Title	Verify InvoiceItemResource namespace
Objective	Ensure InvoiceItemResource is located in Crater\Http\Resources namespace.
Preconditions	- Application running - InvoiceItemResource class available
Test Steps	1. Get Reflection of InvoiceItemResource. 2. Retrieve and verify namespace name.
Test Data	- InvoiceItemResource class reflection
Expected Result	- Namespace name is 'Crater\Http\Resources'.
Actual Result	Namespace name verified as Crater\Http\Resources.
Status	Pass
Severity	High

Test Case ID	II-007
Title	Verify InvoiceItemResource has toArray method
Objective	Confirm the existence of toArray method in InvoiceItemResource.

Preconditions	- Application running - InvoiceItemResource class available
Test Steps	1. Get Reflection of InvoiceItemResource class. 2. Check for presence of toArray method.
Test Data	- InvoiceItemResource class reflection
Expected Result	- toArray method exists in InvoiceItemResource.
Actual Result	toArray method presence verified.
Status	Pass
Severity	High

Test Case ID	II-008
Title	Verify InvoiceItemCollection extends ResourceCollection
Objective	Confirm InvoiceItemCollection class inherits ResourceCollection.
Preconditions	- Application running - InvoiceItemCollection class available
Test Steps	1. Instantiate InvoiceItemCollection with empty Collection. 2. Verify instance is of type ResourceCollection.
Test Data	- Input: new Collection([])
Expected Result	- InvoiceItemCollection is instance of Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	InvoiceItemCollection instantiated and is a ResourceCollection.
Status	Pass
Severity	High

Test Case ID	II-009
Title	Verify InvoiceItemCollection returns empty array for empty input
Objective	Confirm toArray method returns an empty array when InvoiceItemCollection is empty.
Preconditions	- Application running - InvoiceItemCollection class available
Test Steps	1. Instantiate Request object. 2. Instantiate InvoiceItemCollection with empty collection. 3. Call toArray with the request. 4. Verify result is an empty array.
Test Data	- Input: new Collection([]) - Request object
Expected Result	- toArray returns an empty array.
Actual Result	toArray returns an empty array as expected.
Status	Pass
Severity	High

Test Case ID	II-010
Title	Verify InvoiceItemCollection delegates to parent implementation
Objective	Ensure InvoiceItemCollection calls parent::toArray method.
Preconditions	- Application running - InvoiceItemCollection class available
Test Steps	1. Get Reflection of InvoiceItemCollection class. 2. Read file contents of InvoiceItemCollection. 3. Check for usage of 'parent::toArray'.

Test Data	- InvoiceItemCollection class file content
Expected Result	- 'parent::toArray' is present in class code.
Actual Result	Verified presence of 'parent::toArray' in InvoiceItemCollection code.
Status	Pass
Severity	High

File: InvoicePdfController-Test.txt

Test Case ID	IPC-001
Title	Returns PDF data when the request includes a preview parameter
Objective	Verify that the controller returns PDF data when the 'preview' parameter is present in the request.
Preconditions	- Application running - Database seeded - PDF generation feature enabled - Invoice model available - InvoicePdfController available
Test Steps	1. Create a mock Request object. 2. Set up the Request to expect the presence of the 'preview' parameter and return true. 3. Create a mock Invoice object. 4. Set up the Invoice to expect the getPDFData method call and return 'mocked_pdf_data_content'. 5. Instantiate the InvoicePdfController. 6. Invoke the controller with the mock Request and Invoice objects. 7. Capture the result.
Test Data	- Request: has('preview') returns true - Invoice: getPDFData returns 'mocked_pdf_data_content'
Expected Result	- The controller should return 'mocked_pdf_data_content' as the PDF data.
Actual Result	The controller returned 'mocked_pdf_data_content' as the PDF data.
Status	Pass
Severity	High

Test Case ID	IPC-002
Title	Returns generated PDF stream when preview parameter is absent
Objective	Verify that the controller returns a generated PDF stream when the 'preview' parameter is not present in the request.
Preconditions	- Application running - Database seeded - PDF generation feature enabled - Invoice model available - InvoicePdfController available
Test Steps	1. Create a mock Request object. 2. Set up the Request to expect the check for 'preview' parameter and return false. 3. Create a mock Invoice object. 4. Set up the Invoice to expect the getGeneratedPDFOrStream method call with 'invoice' and return 'mocked_pdf_stream_content'. 5. Instantiate the InvoicePdfController. 6. Invoke the controller with the mock Request and Invoice objects. 7. Capture the result.
Test Data	- Request: has('preview') returns false - Invoice: getGeneratedPDFOrStream('invoice') returns 'mocked_pdf_stream_content'
Expected Result	- The controller should return 'mocked_pdf_stream_content' as the PDF stream.
Actual Result	The controller returned 'mocked_pdf_stream_content' as the PDF stream.
Status	Pass
Severity	High

File: InvoiceViewedMail-Test.txt

Test Case ID	IVM-001
Title	Instantiation of InvoiceViewedMail
Objective	Verify that the InvoiceViewedMail class can be successfully instantiated.
Preconditions	- Application running - Database seeded - InvoiceViewedMail class available and autoloaded
Test Steps	1. Create an array with 'invoice_number' => 'INV-001'. 2. Instantiate the InvoiceViewedMail class with this array. 3. Verify that the created object is an instance of InvoiceViewedMail.
Test Data	- Input: ['invoice_number' => 'INV-001'] - Expected value: Instance of InvoiceViewedMail
Expected Result	- The created object is an instance of InvoiceViewedMail.
Actual Result	- The object was correctly instantiated as InvoiceViewedMail.
Status	Pass
Severity	High

Test Case ID	IVM-002
Title	InvoiceViewedMail Extends Mailable
Objective	Verify that InvoiceViewedMail extends the Illuminate\Mail\Mailable class.
Preconditions	- Application running - Database seeded - InvoiceViewedMail class available
Test Steps	1. Create an array with 'invoice_number' => 'INV-001'. 2. Instantiate InvoiceViewedMail with this array. 3. Verify that the object is an instance of Illuminate\Mail\Mailable.
Test Data	- Input: ['invoice_number' => 'INV-001'] - Expected value: Instance of Illuminate\Mail\Mailable
Expected Result	- InvoiceViewedMail is an instance of Illuminate\Mail\Mailable.
Actual Result	- Verified that InvoiceViewedMail extends the Mailable class.
Status	Pass
Severity	High

Test Case ID	IVM-003
Title	InvoiceViewedMail Namespace Verification
Objective	Verify that InvoiceViewedMail resides in the Crater\Mail namespace.
Preconditions	- Application running - InvoiceViewedMail class available
Test Steps	1. Create a ReflectionClass instance of InvoiceViewedMail. 2. Retrieve and check the namespace name.
Test Data	- Class: InvoiceViewedMail - Expected Namespace: 'Crater\Mail'
Expected Result	- The namespace of InvoiceViewedMail is 'Crater\Mail'.
Actual Result	- Namespace verified as 'Crater\Mail'.
Status	Pass
Severity	Medium

Test Case ID	IVM-004

Title	Use of Queueable Trait in InvoiceViewedMail
Objective	Verify that the InvoiceViewedMail class uses the Illuminate\Bus\Queueable trait.
Preconditions	- Application running - InvoiceViewedMail class available
Test Steps	1. Create a ReflectionClass instance of InvoiceViewedMail. 2. Retrieve associated trait names. 3. Check for the presence of 'Illuminate\Bus\Queueable'.
Test Data	- Class: InvoiceViewedMail - Expected Trait: 'Illuminate\Bus\Queueable'
Expected Result	- InvoiceViewedMail uses the Illuminate\Bus\Queueable trait.
Actual Result	- Trait 'Illuminate\Bus\Queueable' found in InvoiceViewedMail.
Status	Pass
Severity	Medium

Test Case ID	IVM-005
Title	Use of SerializesModels Trait in InvoiceViewedMail
Objective	Verify that the InvoiceViewedMail class uses the Illuminate\Queue\SerializesModels trait.
Preconditions	- Application running - InvoiceViewedMail class available
Test Steps	1. Create a ReflectionClass instance of InvoiceViewedMail. 2. Retrieve associated trait names. 3. Check for the presence of 'Illuminate\Queue\SerializesModels'.
Test Data	- Class: InvoiceViewedMail - Expected Trait: 'Illuminate\Queue\SerializesModels'
Expected Result	- InvoiceViewedMail uses the Illuminate\Queue\SerializesModels trait.
Actual Result	- Trait 'Illuminate\Queue\SerializesModels' found in InvoiceViewedMail.
Status	Pass
Severity	Medium

Test Case ID	IVM-006
Title	Existence and Visibility of Data Property in InvoiceViewedMail
Objective	Ensure InvoiceViewedMail has a public 'data' property.
Preconditions	- Application running - InvoiceViewedMail class available
Test Steps	1. Create a ReflectionClass instance of InvoiceViewedMail. 2. Check for the existence of the 'data' property. 3. Check that the 'data' property is public.
Test Data	- Class: InvoiceViewedMail - Expected property: 'data' - Expected visibility: public
Expected Result	- 'data' property exists and is public in InvoiceViewedMail.
Actual Result	- Verified existence and public visibility of 'data' property.
Status	Pass
Severity	Medium

Test Case ID	IVM-007
Title	Constructor Sets Data Property

Objective	Validate that the constructor of InvoiceViewedMail correctly sets the 'data' property.
Preconditions	- Application running - InvoiceViewedMail class available
Test Steps	1. Create an array: ['invoice_number' => 'INV-001', 'customer' => 'John Doe']. 2. Instantiate InvoiceViewedMail with this array. 3. Verify that $mail->data equals the input array. 4. Verify $mail->data['invoice_number'] is 'INV-001'. 5. Verify $mail->data['customer'] is 'John Doe'.
Test Data	- Input: ['invoice_number' => 'INV-001', 'customer' => 'John Doe'] - Expected values: - $mail->data = input array - $mail->data['invoice_number'] = 'INV-001' - $mail->data['customer'] = 'John Doe'
Expected Result	- The 'data' property equals the provided array. - 'invoice_number' and 'customer' values are correctly set in 'data'.
Actual Result	- Data property set as provided; values match input array.
Status	Pass
Severity	High

Test Case ID	IVM-008
Title	Existence and Visibility of Build Method
Objective	Verify that the InvoiceViewedMail class has a public build() method.
Preconditions	- Application running - InvoiceViewedMail class available
Test Steps	1. Create a ReflectionClass instance of InvoiceViewedMail. 2. Check for existence of the 'build' method. 3. Verify that the 'build' method is public.
Test Data	- Class: InvoiceViewedMail - Expected method: 'build' - Expected visibility: public
Expected Result	- InvoiceViewedMail has a public 'build' method.
Actual Result	- 'build' method exists and is public.
Status	Pass
Severity	High

Test Case ID	IVM-009
Title	Markdown View Usage in Build Method
Objective	Verify that the build method of InvoiceViewedMail uses a markdown view for emails.
Preconditions	- Application running - InvoiceViewedMail class available
Test Steps	1. Create a ReflectionClass instance of InvoiceViewedMail. 2. Read the file contents of the class. 3. Check for presence of '->markdown(' in build method. 4. Check that the view 'emails.viewed.invoice' is used.
Test Data	- Class file contents - Expected value: '->markdown(' and 'emails.viewed.invoice'
Expected Result	- The build method uses markdown with view 'emails.viewed.invoice'.
Actual Result	- Markdown usage with 'emails.viewed.invoice' confirmed in build method.
Status	Pass
Severity	Medium

Test Case ID	IVM-010
Title	Setting From Address in Build Method
Objective	Verify that the build method sets the from address using mail configuration.
Preconditions	- Application running - Mail configuration available - InvoiceViewedMail class available
Test Steps	1. Create a ReflectionClass instance of InvoiceViewedMail. 2. Read the file contents of the class. 3. Check for presence of '->from(' in build method. 4. Verify that config('mail.from.address') and config('mail.from.name') are referenced.
Test Data	- Class file contents - Expected values: '->from(', 'config('mail.from.address')', 'config('mail.from.name')'
Expected Result	- The build method sets the mail from address and name using configuration values.
Actual Result	- Build method correctly sets mail from address and name using config settings.
Status	Pass
Severity	High

File: Invoicing-Test.txt

Test Case ID	I-001
Title	InvoiceResource class extends JsonResource
Objective	Verify that the InvoiceResource class inherits from Laravel's JsonResource.
Preconditions	- Application running - InvoiceResource class available - Dependencies autoloaded
Test Steps	1. Instantiate InvoiceResource with an object containing 'id' => 1. 2. Check the instance type of the created resource.
Test Data	- Input: (object)['id' => 1] - Expected type: \Illuminate\Http\Resources\Json\JsonResource
Expected Result	- The InvoiceResource instance must be of type JsonResource.
Actual Result	- InvoiceResource instance is of type JsonResource.
Status	Pass
Severity	Medium

Test Case ID	I-002
Title	InvoiceResource is defined in the correct namespace
Objective	Verify that InvoiceResource is placed under 'Crater\Http\Resources'.
Preconditions	- Application running - InvoiceResource class present
Test Steps	1. Obtain InvoiceResource class via ReflectionClass. 2. Retrieve the namespace from the class reflection.
Test Data	- Class: InvoiceResource - Expected namespace: Crater\Http\Resources
Expected Result	- Namespace of InvoiceResource is "Crater\Http\Resources".
Actual Result	- Namespace is "Crater\Http\Resources".
Status	Pass
Severity	Low

Test Case ID	I-003
Title	InvoiceResource contains a toArray method
Objective	Ensure InvoiceResource has the toArray method implemented.
Preconditions	- Application running - InvoiceResource class present
Test Steps	1. Use ReflectionClass to analyze InvoiceResource. 2. Check if the toArray method exists.
Test Data	- Class: InvoiceResource
Expected Result	- toArray method exists in InvoiceResource.
Actual Result	- toArray method exists.
Status	Pass
Severity	Medium

Test Case ID	I-004
Title	InvoiceResource toArray output includes required invoice fields
Objective	Confirm toArray in InvoiceResource outputs all essential invoice fields.

Preconditions	- Application running - InvoiceResource class and source file present
Test Steps	1. Reflect InvoiceResource to get source file. 2. Read the file content. 3. Check for: id, invoice_date, due_date, invoice_number, status, paid_status in toArray's output.
Test Data	- Source code with required fields in toArray.
Expected Result	- toArray includes all required invoice fields in its output array.
Actual Result	- All required fields are present in toArray output.
Status	Pass
Severity	High

Test Case ID	I-005
Title	InvoiceResource includes financial fields in toArray
Objective	Confirm toArray adds all financial fields to the resource output.
Preconditions	- Application running - InvoiceResource source code accessible
Test Steps	1. Use ReflectionClass to get InvoiceResource source file. 2. Inspect the content for financial fields: total, sub_total, tax, discount.
Test Data	- Expected financial fields: 'total', 'sub_total', 'tax', 'discount'
Expected Result	- All specified financial fields are included in the toArray output.
Actual Result	- All financial fields present.
Status	Pass
Severity	High

Test Case ID	I-006
Title	InvoiceResource includes relationship fields in toArray
Objective	Verify the toArray method includes fields for relationships.
Preconditions	- Application running - InvoiceResource source code accessible
Test Steps	1. Use ReflectionClass to get InvoiceResource source file. 2. Inspect the content for relationship fields: customer_id, currency_id, creator_id.
Test Data	- Expected fields: 'customer_id', 'currency_id', 'creator_id'
Expected Result	- Relationship fields are present in the toArray array.
Actual Result	- Relationship fields present.
Status	Pass
Severity	Medium

Test Case ID	I-007
Title	InvoiceResource uses conditional loading with when()
Objective	Ensure InvoiceResource uses the when() method for conditional relationships.
Preconditions	- Application running - InvoiceResource source code accessible
Test Steps	1. Use ReflectionClass to obtain InvoiceResource source file. 2. Inspect for usage of $this->when() in the file.
Test Data	- Code line: $this->when(
Expected Result	- $this->when is present for conditional inclusion in resource output.

Actual Result	- $this->when found in source code.
Status	Pass
Severity	Medium

Test Case ID	I-008
Title	InvoicesController class extends Controller
Objective	Verify that InvoicesController inherits from the base Controller class.
Preconditions	- Application running - InvoicesController class available
Test Steps	1. Instantiate InvoicesController. 2. Check if the instance type matches \Crater\Http\Controllers\Controller.
Test Data	- Class: InvoicesController
Expected Result	- Instance is of type Controller.
Actual Result	- InvoicesController is an instance of Controller.
Status	Pass
Severity	Medium

Test Case ID	I-009
Title	InvoicesController is defined in the correct namespace
Objective	Verify InvoicesController is located under 'Crater\Http\Controllers\V1\Admin\Invoice'.
Preconditions	- Application running - InvoicesController class present
Test Steps	1. Use ReflectionClass on InvoicesController. 2. Retrieve and compare the namespace.
Test Data	- Class: InvoicesController - Expected namespace: Crater\Http\Controllers\V1\Admin\Invoice
Expected Result	- Namespace matches "Crater\Http\Controllers\V1\Admin\Invoice".
Actual Result	- Namespace is correct.
Status	Pass
Severity	Low

Test Case ID	I-010
Title	InvoicesController implements CRUD methods
Objective	Confirm that InvoicesController contains index, store, show, update, and delete methods.
Preconditions	- Application running - InvoicesController source code accessible
Test Steps	1. Use ReflectionClass to enumerate methods. 2. Ensure presence of methods: index, store, show, update, delete.
Test Data	- Method names: index, store, show, update, delete
Expected Result	- All CRUD methods are implemented.
Actual Result	- All CRUD methods found.
Status	Pass
Severity	High

Test Case ID	I-011

Title	InvoicesController index method uses authorization
Objective	Ensure that only authorized users can access the index method.
Preconditions	- Application running - InvoicesController source code accessible
Test Steps	1. Use ReflectionClass to get 'index' method from InvoicesController. 2. Inspect code for usage of $this->authorize('viewAny', Invoice::class).
Test Data	- Code line: $this->authorize('viewAny', Invoice::class)
Expected Result	- Authorization is checked within the index method.
Actual Result	- Authorization code is present.
Status	Pass
Severity	High

Test Case ID	I-012
Title	InvoicesController store method creates invoice and dispatches job
Objective	Ensure that the store method creates an invoice and dispatches the GenerateInvoicePdfJob.
Preconditions	- Application running - InvoicesController source code accessible
Test Steps	1. Use ReflectionClass to read the store method's source code. 2. Confirm it calls Invoice::createInvoice($request). 3. Confirm it dispatches GenerateInvoicePdfJob.
Test Data	- Method: store - Expected calls: Invoice::createInvoice($request), GenerateInvoicePdfJob::dispatch
Expected Result	- store method creates invoice and dispatches PDF job.
Actual Result	- Both operations confirmed in code.
Status	Pass
Severity	High

Test Case ID	I-013
Title	InvoicesController update method error handling
Objective	Verify that errors are handled correctly in the update method.
Preconditions	- Application running - InvoicesController source code accessible
Test Steps	1. Read the update method source code in InvoicesController. 2. Confirm error handling for is_string($invoice). 3. Confirm usage of respondJson for error responses.
Test Data	- Error condition: is_string($invoice) - Response: respondJson
Expected Result	- Errors are detected and proper JSON response is returned.
Actual Result	- Error handling and responding JSON found.
Status	Pass
Severity	High

Test Case ID	I-014
Title	InvoicesController delete method requires DeleteInvoiceRequest
Objective	Ensure the delete method's parameter enforces the DeleteInvoiceRequest type.

Preconditions	- Application running - InvoicesController source code accessible
Test Steps	1. Use ReflectionClass to get the delete method. 2. Inspect method parameters for type-hint. 3. Verify the parameter is of type DeleteInvoiceRequest.
Test Data	- Parameter type: DeleteInvoiceRequest
Expected Result	- Delete method parameter accepts only DeleteInvoiceRequest.
Actual Result	- Parameter correctly type-hinted as DeleteInvoiceRequest.
Status	Pass
Severity	High

Test Case ID	I-015
Title	InvoicesRequest class extends FormRequest
Objective	Verify that InvoicesRequest inherits from FormRequest.
Preconditions	- Application running - InvoicesRequest class available
Test Steps	1. Instantiate InvoicesRequest. 2. Confirm its instance is of type \Illuminate\Foundation\Http\FormRequest.
Test Data	- Input: InvoicesRequest instance
Expected Result	- InvoicesRequest is an instance of FormRequest.
Actual Result	- Instance confirmed to be FormRequest.
Status	Pass
Severity	Medium

Test Case ID	I-016
Title	InvoicesRequest is defined in the correct namespace
Objective	Ensure the InvoicesRequest class is located under 'Crater\Http\Requests'.
Preconditions	- Application running - InvoicesRequest class present
Test Steps	1. Use ReflectionClass on InvoicesRequest. 2. Retrieve and check the namespace.
Test Data	- Expected namespace: Crater\Http\Requests
Expected Result	- Namespace is "Crater\Http\Requests".
Actual Result	- Namespace matches expectation.
Status	Pass
Severity	Low

Test Case ID	I-017
Title	InvoicesRequest implements required methods
Objective	Ensure InvoicesRequest has authorize, rules, and getInvoicePayload methods.
Preconditions	- Application running - InvoicesRequest class accessible
Test Steps	1. Use ReflectionClass to check InvoicesRequest. 2. Confirm existence of authorize, rules, getInvoicePayload methods.
Test Data	- Method names: authorize, rules, getInvoicePayload
Expected Result	- All required methods are present.
Actual Result	- Methods exist in InvoicesRequest.

Status	Pass
Severity	High

Test Case ID	I-018
Title	InvoicesRequest authorize method allows request
Objective	Verify that the authorize method returns true.
Preconditions	- Application running - InvoicesRequest class available
Test Steps	1. Instantiate InvoicesRequest. 2. Call authorize(). 3. Check return value.
Test Data	- Call: authorize() - Expected output: true
Expected Result	- Authorize returns true, permitting the request.
Actual Result	- Authorize returns true.
Status	Pass
Severity	Medium

Test Case ID	I-019
Title	InvoicesRequest rules require essential invoice fields
Objective	Verify the rules method includes required fields for invoice creation.
Preconditions	- Application running - InvoicesRequest source code accessible
Test Steps	1. Use ReflectionClass to get source file for InvoicesRequest. 2. Check rules for: invoice_date, customer_id, invoice_number, items, total.
Test Data	- Rule keys: 'invoice_date', 'customer_id', 'invoice_number', 'items', 'total'
Expected Result	- All essential fields included in rules.
Actual Result	- All required fields present.
Status	Pass
Severity	High

Test Case ID	I-020
Title	InvoicesRequest getInvoicePayload merges default values
Objective	Confirm getInvoicePayload merges in appropriate default values.
Preconditions	- Application running - InvoicesRequest source code accessible
Test Steps	1. Read InvoicesRequest source code for getInvoicePayload logic. 2. Check for merging of: STATUS_SENT, STATUS_DRAFT, STATUS_UNPAID, exchange_rate, base_total.
Test Data	- Default values: Invoice::STATUS_SENT, Invoice::STATUS_DRAFT, Invoice::STATUS_UNPAID, exchange_rate, base_total
Expected Result	- getInvoicePayload merges all specified defaults correctly.
Actual Result	- Default values are merged as expected.
Status	Pass
Severity	Medium

File: ItemCollection-Test.txt

Test Case ID	IC-001
Title	ItemCollection Instantiation
Objective	Verify that ItemCollection can be instantiated successfully.
Preconditions	- Application running - Database seeded - ItemCollection and Collection classes available
Test Steps	1. Instantiate an ItemCollection with an empty Collection object. 2. Verify that the resulting object is an instance of ItemCollection.
Test Data	- Input: new Collection([]) - Expected: Object is ItemCollection
Expected Result	- ItemCollection instance is created without errors.
Actual Result	ItemCollection instance is created successfully.
Status	Pass
Severity	Medium

Test Case ID	IC-002
Title	ItemCollection Extends ResourceCollection
Objective	Confirm that ItemCollection inherits from ResourceCollection.
Preconditions	- Application running - Database seeded - ItemCollection and ResourceCollection classes available
Test Steps	1. Instantiate an ItemCollection with an empty Collection object. 2. Verify that the object is an instance of ResourceCollection.
Test Data	- Input: new Collection([]) - Expected: Object is ResourceCollection
Expected Result	- ItemCollection is an instance of \Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	ItemCollection correctly extends ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	IC-003
Title	ItemCollection Namespace Validation
Objective	Ensure ItemCollection class is defined in the correct namespace.
Preconditions	- Application running - ItemCollection class file accessible
Test Steps	1. Reflect the ItemCollection class. 2. Retrieve the namespace name. 3. Verify the namespace is 'Crater\Http\Resources'.
Test Data	- Input: ItemCollection class - Expected Namespace: Crater\Http\Resources
Expected Result	- ItemCollection is in the namespace 'Crater\Http\Resources'.
Actual Result	ItemCollection is in the correct namespace.
Status	Pass
Severity	Low

Test Case ID	IC-004

Title	ItemCollection has toArray Method
Objective	Verify the ItemCollection class contains the 'toArray' method.
Preconditions	- Application running - ItemCollection class file accessible
Test Steps	1. Reflect the ItemCollection class. 2. Check for existence of the 'toArray' method.
Test Data	- Input: ItemCollection class - Expected Methods: toArray
Expected Result	- Method 'toArray' exists in ItemCollection.
Actual Result	Method 'toArray' is present.
Status	Pass
Severity	Medium

Test Case ID	IC-005
Title	ItemCollection toArray is Public and Non-Static
Objective	Confirm the visibility and nature of the toArray method.
Preconditions	- Application running - ItemCollection class file accessible
Test Steps	1. Reflect the ItemCollection class. 2. Get the 'toArray' method reflection. 3. Check that the method is public. 4. Verify that the method is not static.
Test Data	- Input: ItemCollection class - Method: toArray
Expected Result	- 'toArray' is public and non-static.
Actual Result	'toArray' is public and non-static.
Status	Pass
Severity	Medium

Test Case ID	IC-006
Title	ItemCollection toArray Method Parameter Validation
Objective	Validate that the toArray method accepts a request parameter.
Preconditions	- Application running - ItemCollection class file accessible
Test Steps	1. Reflect the ItemCollection class. 2. Get the 'toArray' method reflection. 3. Inspect method parameters. 4. Verify there is one parameter named 'request'.
Test Data	- Input: ItemCollection class - Method parameters: ['request']
Expected Result	- The toArray method has one parameter named 'request'.
Actual Result	toArray accepts a single 'request' parameter.
Status	Pass
Severity	Medium

Test Case ID	IC-007
Title	ItemCollection Returns Empty Array for Empty Collection
Objective	Confirm that toArray returns an empty array when ItemCollection is instantiated with an empty array.

Preconditions	- Application running - ItemCollection class operational - Request class available
Test Steps	1. Instantiate a Request object. 2. Instantiate an ItemCollection with an empty Collection. 3. Call toArray passing the request. 4. Verify the result is an empty array.
Test Data	- Input: new Collection([]), new Request() - Output: []
Expected Result	- toArray returns an empty array.
Actual Result	toArray returns an empty array for empty collection.
Status	Pass
Severity	Medium

Test Case ID	IC-008
Title	ItemCollection Delegates to Parent toArray Method
Objective	Ensure the ItemCollection toArray method calls parent::toArray.
Preconditions	- Application running - ItemCollection and parent class files accessible
Test Steps	1. Reflect ItemCollection class. 2. Access the class contents. 3. Verify the file contains 'parent::toArray'.
Test Data	- Input: ItemCollection source code - Expected Function Call: parent::toArray
Expected Result	- ItemCollection toArray calls parent::toArray.
Actual Result	parent::toArray is called in ItemCollection.
Status	Pass
Severity	Medium

Test Case ID	IC-009
Title	ItemCollection File Size Is Concise
Objective	Validate that the ItemCollection source file contains less than 1000 characters.
Preconditions	- Application running - ItemCollection class file accessible
Test Steps	1. Reflect the ItemCollection class. 2. Read the class file contents. 3. Measure the file length. 4. Verify it is less than 1000 characters.
Test Data	- Input: ItemCollection source code - Expected length: <1000 characters
Expected Result	- The ItemCollection file has less than 1000 characters.
Actual Result	ItemCollection source file size is below 1000 characters.
Status	Pass
Severity	Low

Test Case ID	IC-010
Title	ItemCollection File Line Count Is Minimal
Objective	Ensure the ItemCollection class file has fewer than 30 lines.
Preconditions	- Application running - ItemCollection class file accessible

Test Steps	1. Reflect the ItemCollection class. 2. Read the class file contents. 3. Count number of lines. 4. Verify line count is less than 30.
Test Data	- Input: ItemCollection source code - Expected line count: <30
Expected Result	- The ItemCollection file contains fewer than 30 lines.
Actual Result	ItemCollection file contains less than 30 lines.
Status	Pass
Severity	Low

File: ItemPolicy-Test.txt

Test Case ID	IP-001
Title	Instantiation of ItemPolicy
Objective	Verify that the ItemPolicy class can be successfully instantiated.
Preconditions	- Application running - All dependencies autoloaded - Composer autoload files present
Test Steps	1. Attempt to create a new instance of the ItemPolicy class. 2. Assert that the created object is an instance of ItemPolicy.
Test Data	- Class: ItemPolicy
Expected Result	- ItemPolicy object is instantiated and is of type ItemPolicy.
Actual Result	ItemPolicy object was instantiated successfully and matched the expected type.
Status	Pass
Severity	High

Test Case ID	IP-002
Title	ItemPolicy is in the correct namespace
Objective	Verify that ItemPolicy class resides in the 'Crater\Policies' namespace.
Preconditions	- Application running - ItemPolicy class exists
Test Steps	1. Use reflection to inspect the namespace of ItemPolicy. 2. Assert the namespace matches 'Crater\Policies'.
Test Data	- Class: ItemPolicy - Expected namespace: Crater\Policies
Expected Result	- Namespace of ItemPolicy is 'Crater\Policies'.
Actual Result	Namespace was correctly detected as 'Crater\Policies'.
Status	Pass
Severity	Low

Test Case ID	IP-003
Title	ItemPolicy uses HandlesAuthorization trait
Objective	Ensure ItemPolicy class uses the Illuminate\Auth\Access\HandlesAuthorization trait.
Preconditions	- Application running - ItemPolicy class available
Test Steps	1. Use reflection to get trait names used by ItemPolicy. 2. Assert 'Illuminate\Auth\Access\HandlesAuthorization' is in the trait list.
Test Data	- Class: ItemPolicy - Trait: Illuminate\Auth\Access\HandlesAuthorization
Expected Result	- ItemPolicy uses the HandlesAuthorization trait.
Actual Result	Trait detected in ItemPolicy as expected.
Status	Pass
Severity	High

Test Case ID	IP-004
Title	ItemPolicy contains all required authorization methods

Objective	Verify the existence of all standard authorization methods in ItemPolicy.
Preconditions	- Application running - ItemPolicy class available
Test Steps	1. Use reflection to check for the following methods: viewAny, view, create, update, delete, restore, forceDelete, deleteMultiple. 2. Assert each method exists.
Test Data	- List of methods: viewAny, view, create, update, delete, restore, forceDelete, deleteMultiple
Expected Result	- All specified methods exist within ItemPolicy.
Actual Result	All expected methods were present in the class.
Status	Pass
Severity	High

Test Case ID	IP-005
Title	viewAny method parameter type check
Objective	Ensure that viewAny method in ItemPolicy accepts a User parameter.
Preconditions	- Application running - User and ItemPolicy classes available
Test Steps	1. Use reflection to retrieve parameters for ItemPolicy::viewAny. 2. Assert there is exactly one parameter. 3. Assert the parameter is named 'user'. 4. Assert the parameter type matches or contains 'User'.
Test Data	- Method: viewAny - Parameter: user (type User)
Expected Result	- viewAny method has one parameter named 'user' of type User.
Actual Result	viewAny correctly accepts one 'user' parameter of type User.
Status	Pass
Severity	High

Test Case ID	IP-006
Title	view method parameter type and order check
Objective	Verify that ItemPolicy::view method accepts 'user' and 'item' parameters.
Preconditions	- Application running - User and Item classes available
Test Steps	1. Use reflection to get parameters for ItemPolicy::view. 2. Assert there are two parameters. 3. Assert first parameter is 'user', second is 'item'.
Test Data	- Method: view - Parameters: user, item
Expected Result	- view method has two parameters: user and item.
Actual Result	view method parameters listed as 'user' and 'item'.
Status	Pass
Severity	High

Test Case ID	IP-007
Title	ItemPolicy methods use BouncerFacade for authorization
Objective	Validate that ItemPolicy methods use BouncerFacade::can() with correct item permissions.

Preconditions	- Application running - BouncerFacade accessible - ItemPolicy class available
Test Steps	1. Open ItemPolicy source file. 2. Inspect for usage of 'BouncerFacade::can(' in policy methods. 3. Check for permission strings: view-item, create-item, edit-item, delete-item.
Test Data	- Permission strings: 'view-item', 'create-item', 'edit-item', 'delete-item'
Expected Result	- ItemPolicy methods use BouncerFacade::can() calls for permissions on item actions.
Actual Result	All expected permission checks found using BouncerFacade::can().
Status	Pass
Severity	High

Test Case ID	IP-008
Title	view method company ownership check
Objective	Verify that ItemPolicy::view method checks that the user owns the item's company.
Preconditions	- Application running - User and Item classes seeded
Test Steps	1. Open ItemPolicy source file. 2. Search for statement '$user->hasCompany($item->company_id)' within view method.
Test Data	- Object: user, item - Property: item->company_id
Expected Result	- view method contains a check for $user->hasCompany($item->company_id).
Actual Result	Ownership check statement found in view method as expected.
Status	Pass
Severity	High

Test Case ID	IP-009
Title	All ItemPolicy methods return boolean values
Objective	Ensure that all methods in ItemPolicy return either true or false.
Preconditions	- Application running - ItemPolicy class available
Test Steps	1. Open ItemPolicy source file. 2. Search for 'return true' and 'return false' across methods.
Test Data	- Return values: true, false
Expected Result	- All methods are implemented to return boolean (true/false) values.
Actual Result	Boolean returns present in all policy methods.
Status	Pass
Severity	High

Test Case ID	IP-010
Title	create method permission check
Objective	Confirm that ItemPolicy::create method checks the 'create-item' permission using BouncerFacade.
Preconditions	- Application running - BouncerFacade available - Item class available

Test Steps	1. Use reflection to access ItemPolicy::create method. 2. Inspect for use of 'BouncerFacade::can('create-item', Item::class)'.
Test Data	- Method: create - Permission check: 'create-item', Item::class
Expected Result	- create method includes 'BouncerFacade::can('create-item', Item::class)' check.
Actual Result	Permission check found in create method implementation.
Status	Pass
Severity	High

Test Case ID	IP-011
Title	update method permission check
Objective	Ensure that ItemPolicy::update method validates 'edit-item' permission for the specific item.
Preconditions	- Application running - ItemPolicy and BouncerFacade available
Test Steps	1. Open ItemPolicy source file. 2. Search for 'BouncerFacade::can('edit-item', $item)' in update method.
Test Data	- Method: update - Permission check: 'edit-item', $item
Expected Result	- update method contains a 'BouncerFacade::can('edit-item', $item)' check.
Actual Result	edit-item permission check for item present in update method.
Status	Pass
Severity	High

Test Case ID	IP-012
Title	deleteMultiple method permission check
Objective	Verify that ItemPolicy::deleteMultiple method checks 'delete-item' permission using BouncerFacade and Item class.
Preconditions	- Application running - ItemPolicy and BouncerFacade available
Test Steps	1. Open ItemPolicy source file. 2. Find mention of 'BouncerFacade::can('delete-item', Item::class)' in deleteMultiple method.
Test Data	- Method: deleteMultiple - Permission check: 'delete-item', Item::class
Expected Result	- deleteMultiple method includes 'BouncerFacade::can('delete-item', Item::class)' check.
Actual Result	delete-item permission check correctly present in deleteMultiple method.
Status	Pass
Severity	High

File: ItemResource-Test.txt

Test Case ID	IR-001
Title	Instantiation of ItemResource
Objective	Verify that the ItemResource class can be instantiated successfully.
Preconditions	- Application running - ItemResource class available in codebase - Database seeded (not directly used, but ensures model integrity)
Test Steps	1. Instantiate ItemResource with an object containing an 'id'. 2. Verify that the instantiated object is of type ItemResource.
Test Data	- Input: (object)['id' => 1] - Expected Type: ItemResource
Expected Result	- The object created should be an instance of ItemResource.
Actual Result	- The object is successfully instantiated as ItemResource.
Status	Pass
Severity	Medium

Test Case ID	IR-002
Title	Inheritance from JsonResource
Objective	Verify that ItemResource extends the JsonResource class.
Preconditions	- Application running - ItemResource class and JsonResource available in codebase
Test Steps	1. Instantiate ItemResource with an object containing an 'id'. 2. Verify that ItemResource is an instance of \Illuminate\Http\Resources\Json\JsonResource.
Test Data	- Input: (object)['id' => 1] - Expected Parent Type: \Illuminate\Http\Resources\Json\JsonResource
Expected Result	- ItemResource instance is an instance of JsonResource.
Actual Result	- ItemResource instance is confirmed to extend JsonResource.
Status	Pass
Severity	Medium

Test Case ID	IR-003
Title	Namespace Verification of ItemResource
Objective	Ensure ItemResource resides in the correct namespace.
Preconditions	- Application running - ItemResource class available
Test Steps	1. Create a ReflectionClass instance for ItemResource. 2. Retrieve the namespace using getNamespaceName(). 3. Verify it equals 'Crater\Http\Resources'.
Test Data	- Class: ItemResource - Expected Namespace: 'Crater\Http\Resources'
Expected Result	- Retrieved namespace name is 'Crater\Http\Resources'.
Actual Result	- Namespace name is confirmed as 'Crater\Http\Resources'.
Status	Pass
Severity	Medium

Test Case ID	IR-004

Title	Existence of toArray Method in ItemResource
Objective	Verify that ItemResource declares a toArray method.
Preconditions	- Application running - ItemResource class available
Test Steps	1. Create a ReflectionClass instance for ItemResource. 2. Use hasMethod('toArray') to check if the method exists.
Test Data	- Method Checked: 'toArray' - Class: ItemResource
Expected Result	- ItemResource class contains a toArray method.
Actual Result	- toArray method exists in ItemResource.
Status	Pass
Severity	Medium

Test Case ID	IR-005
Title	toArray Method Includes Basic Item Fields
Objective	Verify that the toArray method of ItemResource includes basic item fields.
Preconditions	- Application running - ItemResource class implemented with toArray method
Test Steps	1. Create a ReflectionClass instance for ItemResource. 2. Get file contents of its implementation. 3. Check for the presence of the following fields within the toArray method: - 'id' - 'name' - 'description' - 'price'
Test Data	- toArray output fields: 'id', 'name', 'description', 'price' - Source checked: ItemResource implementation
Expected Result	- File contains: '\'id\' => $this->id', '\'name\' => $this->name', '\'description\' => $this->description', '\'price\' => $this->price'
Actual Result	- All expected item fields found in toArray output.
Status	Pass
Severity	High

Test Case ID	IR-006
Title	Inclusion of Relationship IDs in ItemResource
Objective	Verify that relationship IDs are included in the toArray output of ItemResource.
Preconditions	- Application running - ItemResource class implemented
Test Steps	1. Create a ReflectionClass for ItemResource. 2. Retrieve implementation file content. 3. Check for the following relationship IDs in the toArray method: - 'unit_id' - 'company_id' - 'creator_id' - 'currency_id'
Test Data	- toArray output fields: 'unit_id', 'company_id', 'creator_id', 'currency_id'
Expected Result	- File contains: '\'unit_id\' => $this->unit_id', '\'company_id\' => $this->company_id', '\'creator_id\' => $this->creator_id', '\'currency_id\' => $this->currency_id'
Actual Result	- All expected relationship IDs found in toArray output.
Status	Pass

Severity	High

Test Case ID	IR-007
Title	Inclusion of Timestamps in ItemResource
Objective	Verify toArray includes timestamp fields in ItemResource.
Preconditions	- Application running - ItemResource class implemented
Test Steps	1. Reflect ItemResource class and access its implementation file. 2. Confirm the toArray method outputs: - 'created_at' - 'updated_at' - 'formatted_created_at'
Test Data	- toArray output fields: 'created_at', 'updated_at', 'formatted_created_at'
Expected Result	- File contains: '\'created_at\' => $this->created_at', '\'updated_at\' => $this->updated_at', '\'formatted_created_at\''
Actual Result	- All timestamp fields found in toArray output.
Status	Pass
Severity	High

Test Case ID	IR-008
Title	Conditional Relationships via when() Method
Objective	Verify that ItemResource uses the when() helper for conditional relationships.
Preconditions	- Application running - ItemResource class implemented
Test Steps	1. Create ReflectionClass for ItemResource. 2. Obtain implementation file content. 3. Search for usage of '$this->when(' indicating conditional relationship handling.
Test Data	- Source checked: File content of ItemResource
Expected Result	- Implementation uses '$this->when(' for conditional relationships.
Actual Result	- '$this->when(' is present in implementation as expected.
Status	Pass
Severity	Medium

Test Case ID	IR-009
Title	Inclusion of Unit Relationship in ItemResource
Objective	Verify that the unit relationship is present in the toArray output via UnitResource.
Preconditions	- Application running - ItemResource and UnitResource classes implemented
Test Steps	1. Reflect on ItemResource and examine file content. 2. Check for inclusion of 'unit' key in toArray. 3. Verify usage of UnitResource for 'unit' relationship.
Test Data	- toArray output fields: 'unit' - Resource used: UnitResource
Expected Result	- File contains: '\'unit\' =>' and 'UnitResource'
Actual Result	- Unit relationship and UnitResource found in output.
Status	Pass
Severity	High

Test Case ID	IR-010
Title	Inclusion of Company and Currency Relationships
Objective	Verify ItemResource includes company and currency relationships via respective resources.
Preconditions	- Application running - ItemResource, CompanyResource, and CurrencyResource classes implemented
Test Steps	1. Reflect on ItemResource and acquire file contents. 2. Ensure 'company' and 'currency' are included as top-level keys. 3. Verify use of CompanyResource for 'company' and CurrencyResource for 'currency'.
Test Data	- toArray output fields: 'company', 'currency' - Resources used: CompanyResource, CurrencyResource
Expected Result	- File contains: '\'company\' =>', 'CompanyResource', '\'currency\' =>', 'CurrencyResource'
Actual Result	- Company and currency relationships included with correct resources.
Status	Pass
Severity	High

File: ItemSalesReportController-Test.txt

Test Case ID	ISRC-001
Title	Instantiating ItemSalesReportController
Objective	Verify that the ItemSalesReportController class can be instantiated without errors.
Preconditions	- Application running - Dependencies for ItemSalesReportController are available - PHP environment set up
Test Steps	1. Attempt to instantiate ItemSalesReportController. 2. Check if the instantiated object is of type ItemSalesReportController.
Test Data	- Class: ItemSalesReportController
Expected Result	- The object is successfully created and is an instance of ItemSalesReportController.
Actual Result	Object is successfully instantiated as ItemSalesReportController.
Status	Pass
Severity	Low

Test Case ID	ISRC-002
Title	ItemSalesReportController Extends Base Controller
Objective	Confirm that ItemSalesReportController inherits from the main Controller class.
Preconditions	- Application running - ItemSalesReportController and Controller classes available
Test Steps	1. Instantiate ItemSalesReportController. 2. Check if the created object is an instance of Controller.
Test Data	- Class: ItemSalesReportController
Expected Result	- The ItemSalesReportController object is an instance of the base Controller class.
Actual Result	ItemSalesReportController is correctly an instance of Controller.
Status	Pass
Severity	Low

Test Case ID	ISRC-003
Title	ItemSalesReportController Namespace Validation
Objective	Ensure ItemSalesReportController is under the designated namespace.
Preconditions	- Application running - ItemSalesReportController class available
Test Steps	1. Use ReflectionClass to inspect ItemSalesReportController. 2. Retrieve the namespace name for the class. 3. Verify it matches 'Crater\Http\Controllers\V1\Admin\Report'.
Test Data	- Class: ItemSalesReportController
Expected Result	- Namespace is 'Crater\Http\Controllers\V1\Admin\Report'.
Actual Result	Namespace matches 'Crater\Http\Controllers\V1\Admin\Report'.
Status	Pass
Severity	Low

Test Case ID	ISRC-004

Title	ItemSalesReportController Invokable Method Presence
Objective	Confirm that ItemSalesReportController implements the __invoke magic method.
Preconditions	- Application running - ItemSalesReportController class available
Test Steps	1. Use ReflectionClass on ItemSalesReportController. 2. Check if the class has a method named __invoke.
Test Data	- Class: ItemSalesReportController
Expected Result	- The __invoke method exists within the controller.
Actual Result	__invoke method is present in ItemSalesReportController.
Status	Pass
Severity	Medium

Test Case ID	ISRC-005
Title	__invoke Method Parameter Verification
Objective	Ensure the __invoke method accepts 'request' and 'hash' parameters, in correct order.
Preconditions	- Application running - ItemSalesReportController class available
Test Steps	1. Use ReflectionClass on ItemSalesReportController. 2. Retrieve the __invoke method. 3. Verify method parameters: first is 'request', second is 'hash'.
Test Data	- Method: __invoke - Parameters: 'request', 'hash'
Expected Result	- __invoke takes 'request' as first, 'hash' as second parameter.
Actual Result	__invoke method parameters are 'request' and 'hash' in correct order.
Status	Pass
Severity	Medium

Test Case ID	ISRC-006
Title	Authorization Call in ItemSalesReportController
Objective	Verify that ItemSalesReportController enforces authorization on report viewing.
Preconditions	- Application running - ItemSalesReportController class file accessible
Test Steps	1. Read the file contents of ItemSalesReportController. 2. Search for authorization call: $this->authorize('view report', $company).
Test Data	- Authorization string: 'view report'
Expected Result	- File contains $this->authorize('view report', $company).
Actual Result	Authorization call is present in ItemSalesReportController.
Status	Pass
Severity	High

Test Case ID	ISRC-007
Title	Company Lookup via unique_hash in Controller
Objective	Ensure that ItemSalesReportController queries the Company using the unique_hash field.
Preconditions	- Application running - ItemSalesReportController class file accessible

Test Steps	1. Read the file contents of ItemSalesReportController. 2. Search for Company::where('unique_hash', $hash)->first().
Test Data	- Query string: 'unique_hash'
Expected Result	- Controller queries Company by 'unique_hash' field.
Actual Result	Company is queried using 'unique_hash' in controller.
Status	Pass
Severity	High

Test Case ID	ISRC-008
Title	Locale Setting from Company Settings
Objective	Verify that ItemSalesReportController sets the locale as per company language preferences.
Preconditions	- Application running - CompanySetting class and settings available
Test Steps	1. Read the file contents of ItemSalesReportController. 2. Confirm retrieval of language from CompanySetting::getSetting('language'). 3. Confirm App::setLocale is called with the retrieved value.
Test Data	- Setting key: 'language'
Expected Result	- Language retrieved from company settings and locale set appropriately.
Actual Result	Locale set from company 'language' setting in controller.
Status	Pass
Severity	Medium

Test Case ID	ISRC-009
Title	Invoice Item Filtering and Querying
Objective	Ensure controller filters InvoiceItems by company, applies filters, and selects item attributes.
Preconditions	- Application running - InvoiceItem model available
Test Steps	1. Read the file contents of ItemSalesReportController. 2. Verify presence of InvoiceItem::whereCompany. 3. Confirm applyInvoiceFilters and itemAttributes methods are called.
Test Data	- Query: InvoiceItem filtering and attributes
Expected Result	- InvoiceItems are filtered by company, filtered by criteria, and relevant attributes selected.
Actual Result	Controller filters and retrieves InvoiceItems with necessary filters and attributes.
Status	Pass
Severity	High

Test Case ID	ISRC-010
Title	Calculation of Total Amount in ItemSalesReportController
Objective	Confirm the controller calculates total amount for all invoice items.
Preconditions	- Application running - InvoiceItem data available
Test Steps	1. Read the file contents of ItemSalesReportController. 2. Verify initialization of $totalAmount. 3. Confirm iteration over items and addition of $item->total_amount to $totalAmount.

Test Data	- Variable: $totalAmount
Expected Result	- $totalAmount is calculated by summing each item's total_amount value.
Actual Result	Total amount is calculated and summed correctly in controller.
Status	Pass
Severity	High

Test Case ID	ISRC-011
Title	Date Formatting and Currency Retrieval
Objective	Ensure dates are formatted per company settings and currency is retrieved correctly.
Preconditions	- Application running - CompanySetting and Currency models available
Test Steps	1. Read the file contents of ItemSalesReportController. 2. Confirm CompanySetting::getSetting('carbon_date_format') is used. 3. Validate usage of Carbon::createFromFormat for date formatting. 4. Confirm Currency::findOrFail is called to get currency.
Test Data	- Setting key: 'carbon_date_format' - Currency retrieval method
Expected Result	- Dates are formatted using company setting. - Currency data is retrieved successfully.
Actual Result	Controller formats dates and retrieves currency as expected.
Status	Pass
Severity	Medium

Test Case ID	ISRC-012
Title	Handling Preview, Download, and Stream Modes in Controller
Objective	Verify the controller can handle preview, download, and stream PDF output modes in responses.
Preconditions	- Application running - PDF generation functions available
Test Steps	1. Read the file contents of ItemSalesReportController. 2. Search for conditional handling of $request->has('preview'), $request->has('download'). 3. Confirm actions: pdf->download(), pdf->stream(), and correct return statements.
Test Data	- Request modes: 'preview', 'download'
Expected Result	- Controller returns correct output for preview, download, and stream modes.
Actual Result	Controller handles preview, download, and stream modes correctly for PDF outputs.
Status	Pass
Severity	High

File: Kernel-Test.txt

Test Case ID	K-001
Title	Verify Global Middleware Stack Definition
Objective	Ensure that the kernel's global middleware stack is defined with the expected middleware components in the correct order and count.
Preconditions	- Application is running - All necessary middleware classes are available and installed - No custom global middleware modifications in the environment
Test Steps	1. Instantiate the HTTP kernel. 2. Retrieve the value of the 'middleware' property using reflection. 3. Verify that the value is an array. 4. Verify that the array contains exactly 7 elements. 5. Check that the array elements match the expected list: - CheckForMaintenanceMode - ValidatePostSize - TrimStrings - ConvertEmptyStringsToNull - TrustProxies - ConfigMiddleware - HandleCors
Test Data	- HTTP kernel instance - Expected middleware stack: [CheckForMaintenanceMode, ValidatePostSize, TrimStrings, ConvertEmptyStringsToNull, TrustProxies, ConfigMiddleware, HandleCors]
Expected Result	- The global middleware stack is an array. - The stack contains exactly 7 elements. - The stack elements exactly match the expected middleware class names in the specified order.
Actual Result	The middleware stack was an array of 7 elements and matched exactly the expected middleware classes.
Status	Pass
Severity	High

Test Case ID	K-002
Title	Verify Web Middleware Group Definition
Objective	Confirm that the 'web' middleware group for the kernel is properly set up with the intended middleware and in the correct order.
Preconditions	- Application is running - All required 'web' middleware classes are correctly installed - No commented lines are counted in validation
Test Steps	1. Instantiate the HTTP kernel. 2. Retrieve the 'middlewareGroups' property using reflection. 3. Check that the value is an array and contains a 'web' key. 4. Verify that the 'web' group is an array. 5. Verify that the array contains exactly 6 middleware classes. 6. Ensure that the elements in the array are: - EncryptCookies - AddQueuedCookiesToResponse - StartSession - ShareErrorsFromSession - VerifyCsrfToken - SubstituteBindings
Test Data	- HTTP kernel instance - Expected 'web' middleware group: [EncryptCookies, AddQueuedCookiesToResponse, StartSession, ShareErrorsFromSession, VerifyCsrfToken, SubstituteBindings]

Expected Result	- ‘middlewareGroups’ is an array with a ‘web’ key. - The ‘web’ group equals an array of 6 specified middleware classes, in the expected order.
Actual Result	‘middlewareGroups’ contained a ‘web’ key with the 6 middleware classes in the proper order.
Status	Pass
Severity	Medium

Test Case ID	K-003
Title	Verify API Middleware Group Definition
Objective	Ensure the kernel’s ‘api’ middleware group is configured with the correct middleware elements and count.
Preconditions	- Application is running - All required ‘api’ middleware classes are available
Test Steps	1. Instantiate the HTTP kernel. 2. Retrieve the ‘middlewareGroups’ property using reflection. 3. Check that the array contains an ‘api’ key. 4. Verify the ‘api’ group is an array. 5. Ensure the array contains exactly 3 middleware entries: - EnsureFrontendRequestsAreStateful - 'throttle:180,1' - SubstituteBindings
Test Data	- HTTP kernel instance - Expected ‘api’ middleware group: [EnsureFrontendRequestsAreStateful, 'throttle:180,1', SubstituteBindings]
Expected Result	- ‘middlewareGroups’ is an array with an ‘api’ key. - The ‘api’ group is an array with the specified 3 middleware entries in the correct order.
Actual Result	The ‘api’ group contained all expected middleware entries in the correct order and count.
Status	Pass
Severity	Medium

Test Case ID	K-004
Title	Verify Route Middleware Definitions
Objective	Validate that the kernel’s route middleware definitions include all expected keys with the correct middleware class mappings.
Preconditions	- Application is running - All required route middleware classes installed

Test Steps	1. Instantiate the HTTP kernel. 2. Use reflection to get the ‘routeMiddleware’ property. 3. Confirm the value is an array. 4. Verify the array contains exactly 17 middleware mappings. 5. Check that the array keys and their mapped classes correspond to: - 'auth' => Authenticate - 'bouncer' => ScopeBouncer - 'auth.basic' => AuthenticateWithBasicAuth - 'bindings' => SubstituteBindings - 'can' => Authorize - 'guest' => RedirectIfAuthenticated - 'customer' => CustomerRedirectIfAuthenticated - 'throttle' => ThrottleRequests - 'verified' => EnsureEmailIsVerified - 'install' => InstallationMiddleware - 'redirect-if-installed' => RedirectIfInstalled - 'redirect-if-unauthenticated' => RedirectIfUnauthorized - 'customer-guest' => CustomerGuest - 'company' => CompanyMiddleware - 'pdf-auth' => PdfMiddleware - 'cron-job' => CronJobMiddleware - 'customer-portal' => CustomerPortalMiddleware
Test Data	- HTTP kernel instance - Expected route middleware mappings (as above)
Expected Result	- ‘routeMiddleware’ is an array of 17 elements. - All keys and mapped classes match the expected definitions.
Actual Result	The route middleware array contained 17 entries matching all expected keys and classes.
Status	Pass
Severity	High

Test Case ID	K-005
Title	Verify Middleware Priority List Definition
Objective	Ensure that the kernel’s middlewarePriority property is an array containing the correct middleware classes in expected order.
Preconditions	- Application is running - All required middleware classes present
Test Steps	1. Instantiate the HTTP kernel. 2. Use reflection to get ‘middlewarePriority’ property. 3. Confirm the property value is an array. 4. Verify the array contains exactly 6 middleware classes. 5. Ensure the array elements are: - StartSession - ShareErrorsFromSession - Authenticate - IlluminateAuthenticateSession - SubstituteBindings - Authorize
Test Data	- HTTP kernel instance - Expected middleware priority: [StartSession, ShareErrorsFromSession, Authenticate, IlluminateAuthenticateSession, SubstituteBindings, Authorize]
Expected Result	- ‘middlewarePriority’ is an array with 6 specified middleware classes in the expected order.
Actual Result	The middlewarePriority property contained all 6 expected middleware classes in the correct order.
Status	Pass
Severity	Medium

File: Listener-Test.txt

Test Case ID	L-001
Title	isListenerFired returns true when listener version is less than event old version
Objective	Verify that isListenerFired returns true when Listener::VERSION is less than event->old.
Preconditions	- Application running - Database seeded - Listener class available - No conflicting event listeners
Test Steps	1. Instantiate an anonymous class extending Listener with VERSION set to '1.0.0'. 2. Create an event object with 'old' property set to '1.0.1'. 3. Call publicIsListenerFired($event) method on the listener instance. 4. Assert that the result is true.
Test Data	- Listener::VERSION: '1.0.0' - event->old: '1.0.1'
Expected Result	- isListenerFired returns true.
Actual Result	isListenerFired returned true.
Status	Pass
Severity	High

Test Case ID	L-002
Title	isListenerFired returns true when listener version equals event old version
Objective	Verify that isListenerFired returns true when Listener::VERSION is equal to event->old.
Preconditions	- Application running - Database seeded - Listener class available
Test Steps	1. Instantiate an anonymous Listener class with VERSION set to '1.0.0'. 2. Create an event object with 'old' property set to '1.0.0'. 3. Call publicIsListenerFired($event) method. 4. Assert that the result is true.
Test Data	- Listener::VERSION: '1.0.0' - event->old: '1.0.0'
Expected Result	- isListenerFired returns true.
Actual Result	isListenerFired returned true.
Status	Pass
Severity	High

Test Case ID	L-003
Title	isListenerFired returns false when listener version is greater than event old version
Objective	Verify that isListenerFired returns false when Listener::VERSION is greater than event->old.
Preconditions	- Application running - Database seeded - Listener class available
Test Steps	1. Instantiate an anonymous Listener class with VERSION set to '1.0.1'. 2. Create an event object with 'old' property set to '1.0.0'. 3. Call publicIsListenerFired($event) method. 4. Assert that the result is false.

Test Data	- Listener::VERSION: '1.0.1' - event->old: '1.0.0'
Expected Result	- isListenerFired returns false.
Actual Result	isListenerFired returned false.
Status	Pass
Severity	High

Test Case ID	L-004
Title	isListenerFired handles complex version strings correctly (less than)
Objective	Verify that isListenerFired returns true when Listener::VERSION is '2.0.0-alpha.1' and event->old is '2.0.0-beta.1'.
Preconditions	- Application running - Database seeded - Listener class available
Test Steps	1. Instantiate an anonymous Listener class with VERSION set to '2.0.0-alpha.1'. 2. Create an event object with 'old' property set to '2.0.0-beta.1'. 3. Call publicIsListenerFired($event). 4. Assert that the result is true.
Test Data	- Listener::VERSION: '2.0.0-alpha.1' - event->old: '2.0.0-beta.1'
Expected Result	- isListenerFired returns true.
Actual Result	isListenerFired returned true.
Status	Pass
Severity	High

Test Case ID	L-005
Title	isListenerFired handles complex version strings correctly (equal)
Objective	Verify that isListenerFired returns true when Listener::VERSION and event->old are both '2.0.0-RC1'.
Preconditions	- Application running - Database seeded - Listener class available
Test Steps	1. Instantiate an anonymous Listener class with VERSION set to '2.0.0-RC1'. 2. Create an event object with 'old' property set to '2.0.0-RC1'. 3. Call publicIsListenerFired($event). 4. Assert that the result is true.
Test Data	- Listener::VERSION: '2.0.0-RC1' - event->old: '2.0.0-RC1'
Expected Result	- isListenerFired returns true.
Actual Result	isListenerFired returned true.
Status	Pass
Severity	High

Test Case ID	L-006
Title	isListenerFired handles complex version strings correctly (greater than)
Objective	Verify that isListenerFired returns false when Listener::VERSION is '2.0.0-RC2' and event->old is '2.0.0-RC1'.
Preconditions	- Application running - Database seeded - Listener class available

Test Steps	1. Instantiate an anonymous Listener class with VERSION set to '2.0.0-RC2'. 2. Create an event object with 'old' property set to '2.0.0-RC1'. 3. Call publicIsListenerFired($event). 4. Assert that the result is false.
Test Data	- Listener::VERSION: '2.0.0-RC2' - event->old: '2.0.0-RC1'
Expected Result	- isListenerFired returns false.
Actual Result	isListenerFired returned false.
Status	Pass
Severity	High

Test Case ID	L-007
Title	isListenerFired handles empty version strings (both empty, should be true)
Objective	Verify that isListenerFired returns true when both Listener::VERSION and event->old are empty strings.
Preconditions	- Application running - Database seeded - Listener class available
Test Steps	1. Instantiate an anonymous Listener class with VERSION set to '' (empty string). 2. Create an event object with 'old' property set to '' (empty string). 3. Call publicIsListenerFired($event). 4. Assert that the result is true.
Test Data	- Listener::VERSION: '' - event->old: ''
Expected Result	- isListenerFired returns true.
Actual Result	isListenerFired returned true.
Status	Pass
Severity	High

Test Case ID	L-008
Title	isListenerFired handles empty listener version with non-empty event old version
Objective	Verify that isListenerFired returns true when Listener::VERSION is empty and event->old is not empty.
Preconditions	- Application running - Database seeded - Listener class available
Test Steps	1. Instantiate an anonymous Listener class with VERSION set to '' (empty string). 2. Create an event object with 'old' property set to '1.0.0'. 3. Call publicIsListenerFired($event). 4. Assert that the result is true.
Test Data	- Listener::VERSION: '' - event->old: '1.0.0'
Expected Result	- isListenerFired returns true.
Actual Result	isListenerFired returned true.
Status	Pass
Severity	High

Test Case ID	L-009

Title	isListenerFired handles non-empty listener version with empty event old version
Objective	Verify that isListenerFired returns false when Listener::VERSION is non-empty and event->old is empty.
Preconditions	- Application running - Database seeded - Listener class available
Test Steps	1. Instantiate an anonymous Listener class with VERSION set to '1.0.0'. 2. Create an event object with 'old' property set to '' (empty string). 3. Call publicIsListenerFired($event). 4. Assert that the result is false.
Test Data	- Listener::VERSION: '1.0.0' - event->old: ''
Expected Result	- isListenerFired returns false.
Actual Result	isListenerFired returned false.
Status	Pass
Severity	High

File: LoginController-Test.txt

Test Case ID	LC-001
Title	Instantiation of LoginController
Objective	Verify that the LoginController class can be successfully instantiated.
Preconditions	- Application running - Database seeded - LoginController class available in codebase
Test Steps	1. Create a new instance of LoginController. 2. Check the type of the created object.
Test Data	- Class: LoginController
Expected Result	- The created object should be an instance of LoginController.
Actual Result	LoginController object was instantiated successfully and is of the correct class.
Status	Pass
Severity	High

Test Case ID	LC-002
Title	LoginController Extends Base Controller
Objective	Verify that LoginController extends the base Controller class.
Preconditions	- Application running - Database seeded - LoginController and Controller classes present
Test Steps	1. Create a new instance of LoginController. 2. Check if the object is also an instance of the Controller class.
Test Data	- Class: LoginController - Base class: Controller
Expected Result	- The LoginController instance should also be an instance of Controller.
Actual Result	LoginController is a subclass of Controller as expected.
Status	Pass
Severity	High

Test Case ID	LC-003
Title	LoginController Namespace Validation
Objective	Ensure that LoginController resides in the correct namespace.
Preconditions	- Application running - Database seeded - LoginController class available
Test Steps	1. Use reflection to inspect LoginController. 2. Retrieve and validate the namespace name.
Test Data	- Class: LoginController - Namespace: 'Crater\Http\Controllers\V1\Admin\Auth'
Expected Result	- The namespace should be 'Crater\Http\Controllers\V1\Admin\Auth'.
Actual Result	Namespace matches expected value.
Status	Pass
Severity	Medium

Test Case ID	LC-004
Title	AuthenticatesUsers Trait Usage

Objective	Verify that LoginController uses the AuthenticatesUsers trait.
Preconditions	- Application running - Database seeded - LoginController class definition with traits
Test Steps	1. Use reflection to view traits used by LoginController. 2. Check if 'Illuminate\Foundation\Auth\AuthenticatesUsers' is included.
Test Data	- Class: LoginController - Trait: 'Illuminate\Foundation\Auth\AuthenticatesUsers'
Expected Result	- List of traits should contain 'Illuminate\Foundation\Auth\AuthenticatesUsers'.
Actual Result	Trait is present as expected.
Status	Pass
Severity	High

Test Case ID	LC-005
Title	Protected redirectTo Property Presence
Objective	Verify that LoginController defines a protected property named 'redirectTo'.
Preconditions	- Application running - Database seeded - LoginController class available
Test Steps	1. Use reflection to check for 'redirectTo' property in LoginController. 2. Validate that the property exists. 3. Check that the property visibility is protected.
Test Data	- Class: LoginController - Property: 'redirectTo'
Expected Result	- Property 'redirectTo' exists in LoginController. - Property is protected.
Actual Result	'redirectTo' property exists and is protected.
Status	Pass
Severity	High

Test Case ID	LC-006
Title	redirectTo Property Assignment
Objective	Verify that redirectTo property uses RouteServiceProvider::HOME.
Preconditions	- Application running - Database seeded - RouteServiceProvider defined
Test Steps	1. Read the LoginController source file. 2. Search for assignment to 'protected $redirectTo = RouteServiceProvider::HOME'.
Test Data	- Class: LoginController - Expected line: 'protected $redirectTo = RouteServiceProvider::HOME'
Expected Result	- The 'redirectTo' property is assigned RouteServiceProvider::HOME.
Actual Result	Assignment to RouteServiceProvider::HOME found as expected.
Status	Pass
Severity	High

Test Case ID	LC-007
Title	Constructor Method Presence and Visibility
Objective	Confirm that LoginController has a public constructor method.

Preconditions	- Application running - Database seeded - LoginController definition
Test Steps	1. Use reflection to check for method '__construct' in LoginController. 2. Validate that the constructor exists. 3. Confirm the constructor is public.
Test Data	- Class: LoginController - Method: '__construct'
Expected Result	- '__construct' method exists and is public.
Actual Result	Constructor exists and is public.
Status	Pass
Severity	High

Test Case ID	LC-008
Title	Guest Middleware Applied in Constructor
Objective	Confirm that the constructor applies 'guest' middleware.
Preconditions	- Application running - Database seeded - Middleware configuration present
Test Steps	1. Read the LoginController source file. 2. Search for '$this->middleware('guest')' in the constructor.
Test Data	- Middleware: 'guest' - Source code content
Expected Result	- '$this->middleware('guest')' is present in the constructor.
Actual Result	Guest middleware applied in constructor as expected.
Status	Pass
Severity	High

Test Case ID	LC-009
Title	Middleware Excludes Logout Method
Objective	Verify that guest middleware excludes the 'logout' method from middleware application.
Preconditions	- Application running - Database seeded - Middleware configured
Test Steps	1. Read the LoginController source code. 2. Search for the line '->except('logout')' in the middleware definition.
Test Data	- Middleware: 'guest' - Exclusion: 'logout'
Expected Result	- Line '->except('logout')' exists, excluding logout from middleware.
Actual Result	Logout exclusion is present in middleware configuration.
Status	Pass
Severity	High

Test Case ID	LC-010
Title	LoginController Documentation Presence
Objective	Verify the LoginController file contains documentation for class and authentication purpose.

Preconditions	- Application running - Database seeded - LoginController code file available
Test Steps	1. Read the LoginController source file. 2. Search for text containing 'Login Controller'. 3. Search for text containing 'authenticating users'.
Test Data	- Source file content
Expected Result	- Source file includes 'Login Controller' in documentation. - Source file includes 'authenticating users' in documentation.
Actual Result	Documentation lines found in file as expected.
Status	Pass
Severity	Medium

File: LoginRequest-Test.txt

Test Case ID	LR-001
Title	Authorize Method Returns True
Objective	Verify that the authorize() method of the LoginRequest class always returns true.
Preconditions	- Application running - Database seeded - LoginRequest class available
Test Steps	1. Instantiate a new LoginRequest object. 2. Call the authorize() method on the LoginRequest instance. 3. Verify that the returned value is true.
Test Data	- LoginRequest instance
Expected Result	- The authorize() method returns true.
Actual Result	The authorize() method returned true as expected.
Status	Pass
Severity	High

Test Case ID	LR-002
Title	Rules Method Returns Correct Validation Rules
Objective	Verify that the rules() method of the LoginRequest class returns the expected validation rules for 'username', 'password', and 'device_name'.
Preconditions	- Application running - Database seeded - LoginRequest class available
Test Steps	1. Instantiate a new LoginRequest object. 2. Call the rules() method on the LoginRequest instance. 3. Check that the returned array contains the keys: 'username', 'password', 'device_name'. 4. Assert that each key ('username', 'password', 'device_name') has the value ['required']. 5. Verify that no additional validation rule keys are present (total count is 3).
Test Data	- LoginRequest instance
Expected Result	- The returned array contains the keys: 'username', 'password', 'device_name'. - The value for 'username' is ['required']. - The value for 'password' is ['required']. - The value for 'device_name' is ['required']. - The total number of keys in the array is 3.
Actual Result	All expected keys were present, each key had the value ['required'], and there were no extra keys.
Status	Pass
Severity	High

File: MailEnvironmentRequest-Test.txt

Test Case ID	MER-001
Title	Authorize method always returns true
Objective	Verify that the MailEnvironmentRequest::authorize() method consistently returns true, indicating authorization is always granted.
Preconditions	- Application running - MailEnvironmentRequest class is accessible - No specific request input required
Test Steps	1. Instantiate a new MailEnvironmentRequest object. 2. Call the authorize() method on the object. 3. Check the return value of the authorize() method.
Test Data	- No input data required
Expected Result	- The authorize() method returns true.
Actual Result	The authorize() method returned true.
Status	Pass
Severity	Medium

Test Case ID	MER-002
Title	Rules method returns correct rules for "smtp" mail driver
Objective	Verify that MailEnvironmentRequest::rules() returns the correct validation rules when 'mail_driver' is set to "smtp".
Preconditions	- Application running - Database seeded - MailEnvironmentRequest class is accessible
Test Steps	1. Create a partial mock MailEnvironmentRequest with 'mail_driver' set to "smtp". 2. Call the rules() method on the mock object. 3. Compare the returned rules to the expected array.
Test Data	- mail_driver: "smtp"
Expected Result	- rules() returns: ``` ['mail_driver' => ['required', 'string'], 'mail_host' => ['required', 'string'], 'mail_port' => ['required'], 'mail_encryption' => ['required', 'string'], 'from_name' => ['required', 'string'], 'from_mail' => ['required', 'string'],] ```
Actual Result	rules() returned the expected rules array for "smtp".
Status	Pass
Severity	High

Test Case ID	MER-003
Title	Rules method returns correct rules for "mailgun" mail driver
Objective	Verify that MailEnvironmentRequest::rules() returns correct validation rules when 'mail_driver' is set to "mailgun".
Preconditions	- Application running - Database seeded - MailEnvironmentRequest class is accessible

Test Steps	1. Create a partial mock MailEnvironmentRequest with 'mail_driver' set to "mailgun". 2. Call the rules() method on the mock object. 3. Compare the returned rules to the expected array.
Test Data	- mail_driver: "mailgun"
Expected Result	- rules() returns: ``` ['mail_driver' => ['required', 'string'], 'mail_mailgun_domain' => ['required', 'string'], 'mail_mailgun_secret' => ['required', 'string'], 'mail_mailgun_endpoint' => ['required', 'string'], 'from_name' => ['required', 'string'], 'from_mail' => ['required', 'string'],] ```
Actual Result	rules() returned the expected rules array for "mailgun".
Status	Pass
Severity	High

Test Case ID	MER-004
Title	Rules method returns correct rules for "ses" mail driver
Objective	Ensure that MailEnvironmentRequest::rules() returns the correct validation rules when 'mail_driver' is set to "ses", including correct handling of nullable values.
Preconditions	- Application running - Database seeded - MailEnvironmentRequest class is accessible
Test Steps	1. Create a partial mock MailEnvironmentRequest with 'mail_driver' set to "ses". 2. Call the rules() method on the mock object. 3. Verify the returned rules array matches expected, especially 'mail_encryption' is nullable.
Test Data	- mail_driver: "ses"
Expected Result	- rules() returns: ``` ['mail_driver' => ['required', 'string'], 'mail_host' => ['required', 'string'], 'mail_port' => ['required'], 'mail_ses_key' => ['required', 'string'], 'mail_ses_secret' => ['required', 'string'], 'mail_encryption' => ['nullable', 'string'], 'from_name' => ['required', 'string'], 'from_mail' => ['required', 'string'],] ```
Actual Result	rules() returned the expected rules array for "ses".
Status	Pass
Severity	High

Test Case ID	MER-005
Title	Rules method returns correct rules for "mail" mail driver
Objective	Verify that MailEnvironmentRequest::rules() returns proper rules when 'mail_driver' is set to "mail".

Preconditions	- Application running - Database seeded - MailEnvironmentRequest class is accessible
Test Steps	1. Create a partial mock MailEnvironmentRequest with 'mail_driver' set to "mail". 2. Call the rules() method on the mock object. 3. Compare the returned rules to the expected array.
Test Data	- mail_driver: "mail"
Expected Result	- rules() returns: ``` ['from_name' => ['required', 'string'], 'from_mail' => ['required', 'string'],] ```
Actual Result	rules() returned the expected rules array for "mail".
Status	Pass
Severity	High

Test Case ID	MER-006
Title	Rules method returns correct rules for "sendmail" mail driver
Objective	Verify that MailEnvironmentRequest::rules() returns correct rules when 'mail_driver' is set to "sendmail".
Preconditions	- Application running - Database seeded - MailEnvironmentRequest class is accessible
Test Steps	1. Create a partial mock MailEnvironmentRequest with 'mail_driver' set to "sendmail". 2. Call the rules() method on the mock object. 3. Compare the returned rules to the expected array.
Test Data	- mail_driver: "sendmail"
Expected Result	- rules() returns: ``` ['from_name' => ['required', 'string'], 'from_mail' => ['required', 'string'],] ```
Actual Result	rules() returned the expected rules array for "sendmail".
Status	Pass
Severity	High

Test Case ID	MER-007
Title	Rules method returns empty array for unknown mail driver
Objective	Confirm that MailEnvironmentRequest::rules() returns an empty array for an unknown mail driver value.
Preconditions	- Application running - MailEnvironmentRequest class is accessible
Test Steps	1. Create a partial mock MailEnvironmentRequest with 'mail_driver' set to "unknown_driver". 2. Call the rules() method on the mock object. 3. Verify the returned value is an empty array.
Test Data	- mail_driver: "unknown_driver"
Expected Result	- rules() returns: []

Actual Result	rules() returned an empty array for unknown mail driver.
Status	Pass
Severity	Medium

Test Case ID	MER-008
Title	Rules method returns empty array when mail_driver is null
Objective	Ensure that MailEnvironmentRequest::rules() returns an empty array when 'mail_driver' is not provided (null).
Preconditions	- Application running - MailEnvironmentRequest class is accessible
Test Steps	1. Create a partial mock MailEnvironmentRequest with 'mail_driver' set to null. 2. Call the rules() method on the mock object. 3. Verify the returned value is an empty array.
Test Data	- mail_driver: null
Expected Result	- rules() returns: []
Actual Result	rules() returned an empty array when mail_driver was null.
Status	Pass
Severity	Medium

Test Case ID	MER-009
Title	Rules method returns empty array when mail_driver is empty string
Objective	Verify MailEnvironmentRequest::rules() returns an empty array when 'mail_driver' is an empty string.
Preconditions	- Application running - MailEnvironmentRequest class is accessible
Test Steps	1. Create a partial mock MailEnvironmentRequest with 'mail_driver' set to '' (empty string). 2. Call the rules() method on the mock object. 3. Verify the returned value is an empty array.
Test Data	- mail_driver: ''
Expected Result	- rules() returns: []
Actual Result	rules() returned an empty array for empty string mail_driver.
Status	Pass
Severity	Medium

File: MailResetPasswordNotification-Test.txt

Test Case ID	MRPN-001
Title	Instantiating MailResetPasswordNotification with a token
Objective	Verify that the MailResetPasswordNotification can be instantiated using a token value.
Preconditions	- Application running - Crater\Notifications\MailResetPasswordNotification class available - PHP environment properly configured
Test Steps	1. Instantiate MailResetPasswordNotification with token 'test-token-123'. 2. Check the type of the created object.
Test Data	- Token: 'test-token-123'
Expected Result	- The instantiated object is an instance of MailResetPasswordNotification.
Actual Result	The object is successfully instantiated and is of type MailResetPasswordNotification.
Status	Pass
Severity	High

Test Case ID	MRPN-002
Title	MailResetPasswordNotification extends ResetPassword base class
Objective	Confirm that MailResetPasswordNotification inherits from ResetPassword.
Preconditions	- Application running - Required classes available
Test Steps	1. Instantiate MailResetPasswordNotification with token 'test-token'. 2. Check if the object is an instance of Illuminate\Auth\Notifications\ResetPassword.
Test Data	- Token: 'test-token'
Expected Result	- The object is an instance of Illuminate\Auth\Notifications\ResetPassword.
Actual Result	The instantiated object inherits from ResetPassword as expected.
Status	Pass
Severity	High

Test Case ID	MRPN-003
Title	Check MailResetPasswordNotification namespace
Objective	Ensure MailResetPasswordNotification is defined in the correct namespace.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to inspect MailResetPasswordNotification class. 2. Retrieve the namespace of the class.
Test Data	- Class: MailResetPasswordNotification
Expected Result	- The namespace of the class is 'Crater\Notifications'.
Actual Result	The namespace was verified as 'Crater\Notifications'.
Status	Pass
Severity	Medium

Test Case ID	MRPN-004
Title	MailResetPasswordNotification uses Queueable trait
Objective	Verify that the Queueable trait is used in MailResetPasswordNotification.

Preconditions	- Application running
Test Steps	1. Use ReflectionClass on MailResetPasswordNotification. 2. Retrieve traits used in the class. 3. Check for presence of 'Illuminate\Bus\Queueable' trait.
Test Data	- Class: MailResetPasswordNotification
Expected Result	- The trait 'Illuminate\Bus\Queueable' is present in the traits list.
Actual Result	Verified that 'Illuminate\Bus\Queueable' is used in the class.
Status	Pass
Severity	Medium

Test Case ID	MRPN-005
Title	MailResetPasswordNotification via method returns mail channel
Objective	Confirm that the via() method returns the mail channel for notifications.
Preconditions	- Application running
Test Steps	1. Instantiate MailResetPasswordNotification with token 'test-token'. 2. Invoke the via() method with null as the parameter. 3. Capture the returned channels.
Test Data	- Token: 'test-token' - Parameter: null
Expected Result	- Returned channels array is ['mail'].
Actual Result	The via() method returned ['mail'] as expected.
Status	Pass
Severity	High

Test Case ID	MRPN-006
Title	MailResetPasswordNotification toMail generates reset link with token
Objective	Verify the reset link in the mail message contains the passed token.
Preconditions	- Application running - Class source file available
Test Steps	1. Use ReflectionClass to get the file path for MailResetPasswordNotification. 2. Read contents of the class file. 3. Check for presence of 'url("/reset-password/".$this->token)' in file content.
Test Data	- Class: MailResetPasswordNotification
Expected Result	- 'url("/reset-password/".$this->token)' is present in the file content.
Actual Result	The reset link containing the token was found in the mail message.
Status	Pass
Severity	High

Test Case ID	MRPN-007
Title	MailResetPasswordNotification toMail includes required message lines
Objective	Ensure required lines appear in the MailResetPasswordNotification email message.
Preconditions	- Application running - Class source file available

Test Steps	1. Use ReflectionClass to get file path for MailResetPasswordNotification. 2. Read the contents of the class file. 3. Check for presence of each of: a. 'Reset Password Notification' b. 'password reset request' c. '->action(\'Reset Password\', $link)' d. 'expire in'
Test Data	- Class: MailResetPasswordNotification
Expected Result	- Each of the following is present in message composition: 1. 'Reset Password Notification' 2. 'password reset request' 3. '->action(\'Reset Password\', $link)' 4. 'expire in'
Actual Result	All required message lines were verified in the email message.
Status	Pass
Severity	High

Test Case ID	MRPN-008
Title	MailResetPasswordNotification toArray returns empty array
Objective	Verify that toArray() returns an empty array as expected.
Preconditions	- Application running
Test Steps	1. Instantiate MailResetPasswordNotification with token 'test-token'. 2. Call toArray() method with null as parameter. 3. Check result is an empty array.
Test Data	- Token: 'test-token' - Parameter: null
Expected Result	- Result is an empty array.
Actual Result	The method returned empty array as expected.
Status	Pass
Severity	Low

File: MigrateUpdateController-Test.txt

Test Case ID	MUC-001
Title	Returns 401 Unauthorized if no user is authenticated
Objective	Verify that the endpoint returns a 401 status code and appropriate error message when no user is authenticated, and ensure Updater::migrateUpdate is not called.
Preconditions	- Application is running - Database is seeded - No user is authenticated in the request
Test Steps	1. Create a mock request with no authenticated user. 2. Mock Updater class and ensure migrateUpdate static method should NOT be called. 3. Instantiate MigrateUpdateController. 4. Invoke the controller's __invoke method with the mock request. 5. Retrieve and check the response status code. 6. Validate the response message content.
Test Data	- Request user: null - Expected Status Code: 401 - Expected Data: { "success": false, "message": "You are not allowed to update this app." }
Expected Result	- The response status code should be 401. - The response body should be: { "success": false, "message": "You are not allowed to update this app." } - Updater::migrateUpdate static method should NOT be called.
Actual Result	- The response status code is 401. - The response body is: { "success": false, "message": "You are not allowed to update this app." } - Updater::migrateUpdate was not called.
Status	Pass
Severity	High

Test Case ID	MUC-002
Title	Returns 401 Unauthorized if authenticated user is not owner
Objective	Verify that the endpoint returns a 401 status code and appropriate error message when the authenticated user is not an owner, and ensure Updater::migrateUpdate is not called.
Preconditions	- Application is running - Database is seeded - Authenticated user exists but does NOT have owner privileges
Test Steps	1. Create a mock authenticated user with isOwner() returning false. 2. Create a mock request with this user as the authenticated user. 3. Mock Updater class and ensure migrateUpdate static method should NOT be called. 4. Instantiate MigrateUpdateController. 5. Invoke the controller's __invoke method with the mock request. 6. Retrieve and check the response status code. 7. Validate the response message content.
Test Data	- Authenticated user: isOwner() returns false - Expected Status Code: 401 - Expected Data: { "success": false, "message": "You are not allowed to update this app." }
Expected Result	- The response status code should be 401. - The response body should be: { "success": false, "message": "You are not allowed to update this app." } - Updater::migrateUpdate static method should NOT be called.

Actual Result	- The response status code is 401. - The response body is: { "success": false, "message": "You are not allowed to update this app." } - Updater::migrateUpdate was not called.
Status	Pass
Severity	High

Test Case ID	MUC-003
Title	Calls Updater::migrateUpdate and returns success if authenticated user is owner
Objective	Verify that Updater::migrateUpdate is called exactly once and the endpoint responds with 200 status and success when the authenticated user is owner.
Preconditions	- Application is running - Database is seeded - Authenticated user exists and has owner privileges
Test Steps	1. Create a mock authenticated user with isOwner() returning true. 2. Create a mock request with this user as the authenticated user. 3. Mock Updater class and configure migrateUpdate static method to expect one call and return null. 4. Instantiate MigrateUpdateController. 5. Invoke the controller's __invoke method with the mock request. 6. Retrieve and check the response status code. 7. Validate the response message content. 8. Verify that Updater::migrateUpdate was called exactly once.
Test Data	- Authenticated user: isOwner() returns true - Expected Status Code: 200 - Expected Data: { "success": true }
Expected Result	- The response status code should be 200. - The response body should be: { "success": true } - Updater::migrateUpdate static method should be called exactly once.
Actual Result	- The response status code is 200. - The response body is: { "success": true } - Updater::migrateUpdate was called exactly once.
Status	Pass
Severity	High

File: ModuleCollection-Test.txt

Test Case ID	MC-001
Title	Instantiation of ModuleCollection
Objective	Verify that the ModuleCollection class can be instantiated successfully.
Preconditions	- Application running - Database seeded - Necessary PHP dependencies installed
Test Steps	1. Create a new empty Collection object. 2. Instantiate a ModuleCollection object with the empty Collection. 3. Verify that the instantiated object is of type ModuleCollection.
Test Data	- Input: new Collection([]) - Expected class: ModuleCollection
Expected Result	- The created object is an instance of ModuleCollection.
Actual Result	The created object is an instance of ModuleCollection.
Status	Pass
Severity	Medium

Test Case ID	MC-002
Title	ModuleCollection Extends ResourceCollection
Objective	Ensure that ModuleCollection inherits from ResourceCollection.
Preconditions	- Application running - Database seeded - Necessary PHP dependencies installed
Test Steps	1. Create a new empty Collection object. 2. Instantiate a ModuleCollection object with the empty Collection. 3. Check that the object is also an instance of ResourceCollection.
Test Data	- Input: new Collection([]) - Expected parent class: Illuminate\Http\Resources\Json\ResourceCollection
Expected Result	- The created object extends ResourceCollection.
Actual Result	The created object extends ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	MC-003
Title	ModuleCollection Namespace Validation
Objective	Confirm that ModuleCollection is in the correct namespace.
Preconditions	- Application running - Database seeded - Necessary PHP dependencies installed
Test Steps	1. Use ReflectionClass on ModuleCollection. 2. Retrieve the namespace name of the class via getNamespaceName(). 3. Compare the namespace to the expected value.
Test Data	- Expected namespace: 'Crater\Http\Resources'
Expected Result	- The namespace name of ModuleCollection is 'Crater\Http\Resources'.
Actual Result	The namespace name is 'Crater\Http\Resources'.
Status	Pass
Severity	Low

Test Case ID	MC-004
Title	ModuleCollection Has toArray Method
Objective	Ensure that the ModuleCollection class implements a public toArray method.
Preconditions	- Application running - Database seeded - Necessary PHP dependencies installed
Test Steps	1. Use ReflectionClass on ModuleCollection. 2. Check for the existence of the method 'toArray'.
Test Data	- Expected method: 'toArray'
Expected Result	- ModuleCollection defines a method 'toArray'.
Actual Result	Method 'toArray' exists in ModuleCollection.
Status	Pass
Severity	Medium

Test Case ID	MC-005
Title	ModuleCollection toArray Returns Empty Array for Empty Collection
Objective	Verify that toArray() produces an empty array when the collection is empty.
Preconditions	- Application running - Database seeded - Necessary PHP dependencies installed
Test Steps	1. Create an empty Collection object. 2. Instantiate ModuleCollection with the empty collection. 3. Call toArray() on the ModuleCollection instance with a Request. 4. Check that the result is an empty array.
Test Data	- Input: new Collection([]) - Request: new Request() - Expected output: []
Expected Result	1. Output is an array. 2. Output is empty.
Actual Result	1. Output is an array. 2. Output is empty.
Status	Pass
Severity	Medium

Test Case ID	MC-006
Title	ModuleCollection Delegates to parent::toArray
Objective	Confirm that ModuleCollection's toArray method calls parent::toArray.
Preconditions	- Application running - Database seeded - Necessary PHP dependencies installed
Test Steps	1. Use ReflectionClass to obtain the filename of ModuleCollection. 2. Read the contents of the ModuleCollection file. 3. Search for the statement 'parent::toArray' in the file contents.
Test Data	- Input: ModuleCollection file content - Expected statement: 'parent::toArray'
Expected Result	- The file contains the statement 'parent::toArray'.
Actual Result	The file contains the statement 'parent::toArray'.
Status	Pass
Severity	Medium

Test Case ID	MC-007
Title	ModuleCollection Conciseness Verification
Objective	Verify that the ModuleCollection class file is concise, with less than 1000 characters.
Preconditions	- Application running - Database seeded - Necessary PHP dependencies installed
Test Steps	1. Use ReflectionClass to obtain the filename for ModuleCollection. 2. Read the entire ModuleCollection file contents. 3. Check if the length of the file content is less than 1000 characters.
Test Data	- Input: ModuleCollection file content - Expected maximum length: 999 characters
Expected Result	- File content length is less than 1000 characters.
Actual Result	File content length is less than 1000 characters.
Status	Pass
Severity	Low

File: ModuleDisabledEvent-Test.txt

Test Case ID	MDE-001
Title	Constructor assigns module property with a string
Objective	Verify that ModuleDisabledEvent constructor correctly assigns the module property when provided a string value.
Preconditions	- Application running - Necessary classes autoloaded (Crater\Events\ModuleDisabledEvent) - No prior dependency on database state
Test Steps	1. Set a variable $moduleName to 'CRM_Module'. 2. Instantiate ModuleDisabledEvent with $moduleName as the parameter. 3. Check the value of $event->module.
Test Data	- Input: 'CRM_Module' (string) - Expected value: $event->module equals 'CRM_Module'
Expected Result	- $event->module is assigned the string 'CRM_Module'
Actual Result	- $event->module is assigned the string 'CRM_Module'
Status	Pass
Severity	Medium

Test Case ID	MDE-002
Title	Constructor assigns module property with an integer
Objective	Confirm that ModuleDisabledEvent constructor correctly assigns the module property when provided an integer value.
Preconditions	- Application running - Necessary classes autoloaded (Crater\Events\ModuleDisabledEvent) - No prior dependency on database state
Test Steps	1. Set a variable $moduleId to 12345. 2. Instantiate ModuleDisabledEvent with $moduleId as the parameter. 3. Check the value of $event->module.
Test Data	- Input: 12345 (integer) - Expected value: $event->module equals 12345
Expected Result	- $event->module is assigned the integer 12345
Actual Result	- $event->module is assigned the integer 12345
Status	Pass
Severity	Medium

Test Case ID	MDE-003
Title	Constructor assigns module property with null
Objective	Ensure that ModuleDisabledEvent constructor correctly assigns the module property when provided a null value.
Preconditions	- Application running - Necessary classes autoloaded (Crater\Events\ModuleDisabledEvent) - No prior dependency on database state
Test Steps	1. Set a variable $module to null. 2. Instantiate ModuleDisabledEvent with $module as the parameter. 3. Check the value of $event->module.
Test Data	- Input: null - Expected value: $event->module is null
Expected Result	- $event->module is assigned null
Actual Result	- $event->module is assigned null

Status	Pass
Severity	Medium

Test Case ID	MDE-004
Title	Constructor assigns module property with an object
Objective	Validate that ModuleDisabledEvent constructor correctly assigns the module property when provided an object and preserves object properties.
Preconditions	- Application running - Necessary classes autoloaded (Crater\Events\ModuleDisabledEvent) - No prior dependency on database state
Test Steps	1. Create $moduleObject of type stdClass. 2. Set $moduleObject->id to 1. 3. Set $moduleObject->name to 'Inventory_Module'. 4. Instantiate ModuleDisabledEvent with $moduleObject as the parameter. 5. Check the value of $event->module. 6. Check $event->module->id. 7. Check $event->module->name.
Test Data	- Input: Object with properties id=1, name='Inventory_Module' - Expected values: - $event->module is identical to $moduleObject - $event->module->id equals 1 - $event->module->name equals 'Inventory_Module'
Expected Result	- $event->module is assigned $moduleObject - $event->module->id equals 1 - $event->module->name equals 'Inventory_Module'
Actual Result	- $event->module is assigned the object with id=1 and name='Inventory_Module'
Status	Pass
Severity	Medium

Test Case ID	MDE-005
Title	Module property is public and accessible after construction
Objective	Ensure the module property of ModuleDisabledEvent is publicly accessible after object creation.
Preconditions	- Application running - Necessary classes autoloaded (Crater\Events\ModuleDisabledEvent) - No prior dependency on database state
Test Steps	1. Set a variable $moduleValue to 'Settings_Module'. 2. Instantiate ModuleDisabledEvent with $moduleValue as parameter. 3. Use reflection to examine the visibility of the 'module' property. 4. Assert that 'module' property is public. 5. Check the value of $event->module.
Test Data	- Input: 'Settings_Module' (string) - Expected values: - ReflectionProperty of 'module' returns isPublic() as true - $event->module equals 'Settings_Module'
Expected Result	- The 'module' property is public - $event->module is assigned 'Settings_Module'
Actual Result	- The 'module' property is public - $event->module is assigned 'Settings_Module'
Status	Pass
Severity	Medium

File: ModuleEnabledEvent-Test.txt

Test Case ID	MEE-001
Title	Instantiation of ModuleEnabledEvent with string module name
Objective	Verify that ModuleEnabledEvent can be instantiated using a valid string as the module name and stores the value correctly.
Preconditions	- Application running - ModuleEnabledEvent class available - PHP environment configured
Test Steps	1. Set the module name to 'blog-module' (string). 2. Instantiate ModuleEnabledEvent with the string module name. 3. Verify the object is an instance of ModuleEnabledEvent. 4. Verify that the object's 'module' property equals 'blog-module'.
Test Data	- Input: 'blog-module' as module name - Expected module property value: 'blog-module'
Expected Result	1. An object of type ModuleEnabledEvent is created. 2. The 'module' property of the object equals 'blog-module'.
Actual Result	1. An object of type ModuleEnabledEvent was instantiated successfully. 2. The 'module' property stored the value 'blog-module' as expected.
Status	Pass
Severity	Medium

Test Case ID	MEE-002
Title	Instantiation of ModuleEnabledEvent with empty string module name
Objective	Verify that ModuleEnabledEvent can be instantiated using an empty string as the module name and stores it correctly.
Preconditions	- Application running - ModuleEnabledEvent class available - PHP environment configured
Test Steps	1. Set the module name to an empty string (''). 2. Instantiate ModuleEnabledEvent with the empty string. 3. Verify the object is an instance of ModuleEnabledEvent. 4. Verify that the object's 'module' property equals ''.
Test Data	- Input: '' (empty string) as module name - Expected module property value: ''
Expected Result	1. An object of type ModuleEnabledEvent is created. 2. The 'module' property of the object equals '' (empty string).
Actual Result	1. An object of type ModuleEnabledEvent was instantiated successfully. 2. The 'module' property stored the empty string as expected.
Status	Pass
Severity	Medium

Test Case ID	MEE-003
Title	Instantiation of ModuleEnabledEvent with numeric module name
Objective	Verify that ModuleEnabledEvent can be instantiated using a numeric value as the module name and stores it correctly.
Preconditions	- Application running - ModuleEnabledEvent class available - PHP environment configured
Test Steps	1. Set the module name to 123 (integer). 2. Instantiate ModuleEnabledEvent with the numeric module name. 3. Verify the object is an instance of ModuleEnabledEvent. 4. Verify that the object's 'module' property equals 123.

Test Data	- Input: 123 (integer) as module name - Expected module property value: 123
Expected Result	1. An object of type ModuleEnabledEvent is created. 2. The 'module' property of the object equals 123.
Actual Result	1. An object of type ModuleEnabledEvent was instantiated successfully. 2. The 'module' property stored the value 123 as expected.
Status	Pass
Severity	Medium

Test Case ID	MEE-004
Title	Instantiation of ModuleEnabledEvent with null module name
Objective	Verify that ModuleEnabledEvent can be instantiated using null as the module name and stores it correctly.
Preconditions	- Application running - ModuleEnabledEvent class available - PHP environment configured
Test Steps	1. Set the module name to null. 2. Instantiate ModuleEnabledEvent with null as the module name. 3. Verify the object is an instance of ModuleEnabledEvent. 4. Verify that the object's 'module' property is null.
Test Data	- Input: null as module name - Expected module property value: null
Expected Result	1. An object of type ModuleEnabledEvent is created. 2. The 'module' property of the object is null.
Actual Result	1. An object of type ModuleEnabledEvent was instantiated successfully. 2. The 'module' property was null as expected.
Status	Pass
Severity	Medium

Test Case ID	MEE-005
Title	Instantiation of ModuleEnabledEvent with object as module name
Objective	Verify that ModuleEnabledEvent can be instantiated using an object as the module name and stores it correctly.
Preconditions	- Application running - ModuleEnabledEvent class available - PHP environment configured
Test Steps	1. Create a stdClass object and set its 'name' property to 'object-module'. 2. Instantiate ModuleEnabledEvent with the object. 3. Verify the object is an instance of ModuleEnabledEvent. 4. Verify that the object's 'module' property is the same object passed in.
Test Data	- Input: stdClass object with property name = 'object-module' - Expected module property value: The same stdClass object used for instantiation
Expected Result	1. An object of type ModuleEnabledEvent is created. 2. The 'module' property of the object is the same object passed as the module name.
Actual Result	1. An object of type ModuleEnabledEvent was instantiated successfully. 2. The 'module' property stored the provided stdClass object as expected.
Status	Pass
Severity	Medium

Test Case ID	MEE-006

Title	ModuleEnabledEvent uses Dispatchable trait
Objective	Verify that the ModuleEnabledEvent class utilizes the Dispatchable trait.
Preconditions	- Application running - ModuleEnabledEvent class available
Test Steps	1. Retrieve all traits used by the ModuleEnabledEvent class. 2. Confirm that the Dispatchable trait is included in the list.
Test Data	- Input: ModuleEnabledEvent class - Expected trait: Dispatchable
Expected Result	1. The Dispatchable trait is present in the list of traits used by ModuleEnabledEvent.
Actual Result	1. The Dispatchable trait was detected in the set of traits used by ModuleEnabledEvent.
Status	Pass
Severity	Low

Test Case ID	MEE-007
Title	ModuleEnabledEvent uses InteractsWithSockets trait
Objective	Verify that the ModuleEnabledEvent class utilizes the InteractsWithSockets trait.
Preconditions	- Application running - ModuleEnabledEvent class available
Test Steps	1. Retrieve all traits used by the ModuleEnabledEvent class. 2. Confirm that the InteractsWithSockets trait is included in the list.
Test Data	- Input: ModuleEnabledEvent class - Expected trait: InteractsWithSockets
Expected Result	1. The InteractsWithSockets trait is present in the list of traits used by ModuleEnabledEvent.
Actual Result	1. The InteractsWithSockets trait was detected in the set of traits used by ModuleEnabledEvent.
Status	Pass
Severity	Low

Test Case ID	MEE-008
Title	ModuleEnabledEvent uses SerializesModels trait
Objective	Verify that the ModuleEnabledEvent class utilizes the SerializesModels trait.
Preconditions	- Application running - ModuleEnabledEvent class available
Test Steps	1. Retrieve all traits used by the ModuleEnabledEvent class. 2. Confirm that the SerializesModels trait is included in the list.
Test Data	- Input: ModuleEnabledEvent class - Expected trait: SerializesModels
Expected Result	1. The SerializesModels trait is present in the list of traits used by ModuleEnabledEvent.
Actual Result	1. The SerializesModels trait was detected in the set of traits used by ModuleEnabledEvent.
Status	Pass
Severity	Low

File: ModuleFacade-Test.txt

Test Case ID	MF-001
Title	Verify getFacadeAccessor returns correct Module class name string
Objective	Ensure that the getFacadeAccessor static method on ModuleFacade returns the fully qualified class name of the Module service as a string.
Preconditions	- Application running - Database seeded - ModuleFacade class is available and autoloaded - No interfering mocks registered
Test Steps	1. Use PHP ReflectionMethod to access the protected static method getFacadeAccessor on the ModuleFacade class. 2. Set the ReflectionMethod to accessible to allow calling the protected method. 3. Invoke getFacadeAccessor statically via Reflection, passing null for instance. 4. Capture the returned string from getFacadeAccessor. 5. Assert that the returned string equals 'Crater\Services\Module\Module'.
Test Data	- Input: Call to ModuleFacade::getFacadeAccessor (via Reflection) - Expected value: 'Crater\Services\Module\Module'
Expected Result	- getFacadeAccessor returns 'Crater\Services\Module\Module'.
Actual Result	getFacadeAccessor returned 'Crater\Services\Module\Module'.
Status	Pass
Severity	High

File: ModulesPolicy-Test.txt

Test Case ID	MP-001
Title	Verify manageModules returns true for owner user
Objective	To confirm that the manageModules policy method correctly grants access when the user is an owner.
Preconditions	- Application running - Database seeded - Mock User instance available - ModulesPolicy class loaded
Test Steps	1. Create a mock User instance. 2. Configure the User mock so that isOwner() returns true. 3. Instantiate a ModulesPolicy object. 4. Call the manageModules() method with the User mock. 5. Check that the returned result is true. 6. Verify that isOwner() method on User mock was called once.
Test Data	- User::isOwner() setup: returns true
Expected Result	- The result from manageModules($user) is TRUE. - The isOwner() method on the User mock is called once.
Actual Result	The result from manageModules($user) is TRUE, and isOwner() was called once as expected.
Status	Pass
Severity	High

Test Case ID	MP-002
Title	Verify manageModules returns false for non-owner user
Objective	To confirm that the manageModules policy method correctly denies access when the user is not an owner.
Preconditions	- Application running - Database seeded - Mock User instance available - ModulesPolicy class loaded
Test Steps	1. Create a mock User instance. 2. Configure the User mock so that isOwner() returns false. 3. Instantiate a ModulesPolicy object. 4. Call the manageModules() method with the User mock. 5. Check that the returned result is false. 6. Verify that isOwner() method on User mock was called once.
Test Data	- User::isOwner() setup: returns false
Expected Result	- The result from manageModules($user) is FALSE. - The isOwner() method on the User mock is called once.
Actual Result	The result from manageModules($user) is FALSE, and isOwner() was called once as expected.
Status	Pass
Severity	High

File: NextNumberController-Test.txt

Test Case ID	NNC-001
Title	Instantiation of NextNumberController
Objective	Verify that the NextNumberController class can be instantiated successfully.
Preconditions	- Application running - Codebase contains NextNumberController definition - Appropriate autoloading set up
Test Steps	1. Attempt to instantiate NextNumberController. 2. Check if the created object is an instance of NextNumberController.
Test Data	- Instantiation: new NextNumberController()
Expected Result	- The created object should be of type NextNumberController.
Actual Result	Object is successfully created and is of type NextNumberController.
Status	Pass
Severity	Medium

Test Case ID	NNC-002
Title	NextNumberController extends the base Controller
Objective	Ensure NextNumberController inherits from the base Controller class.
Preconditions	- Application running - NextNumberController defined in codebase - Controller base class available
Test Steps	1. Instantiate NextNumberController. 2. Verify that NextNumberController is an instance of Controller.
Test Data	- Instantiation: new NextNumberController()
Expected Result	- NextNumberController object is an instance of Controller.
Actual Result	NextNumberController correctly extends Controller.
Status	Pass
Severity	Medium

Test Case ID	NNC-003
Title	NextNumberController has correct namespace
Objective	Validate that NextNumberController is defined in Crater\Http\Controllers\V1\Admin\General namespace.
Preconditions	- Application running - NextNumberController class exists
Test Steps	1. Use reflection to inspect the namespace of NextNumberController. 2. Retrieve and compare the namespace string.
Test Data	- Class: NextNumberController
Expected Result	- Namespace retrieved is 'Crater\Http\Controllers\V1\Admin\General'.
Actual Result	Namespace returned is 'Crater\Http\Controllers\V1\Admin\General' as expected.
Status	Pass
Severity	Medium

Test Case ID	NNC-004
Title	NextNumberController is invokable

Objective	Confirm that NextNumberController implements an __invoke() method, making it invokable.
Preconditions	- Application running - NextNumberController defined
Test Steps	1. Use reflection to check if NextNumberController contains the __invoke method.
Test Data	- Class: NextNumberController
Expected Result	- The class contains a public __invoke() method.
Actual Result	__invoke method found; NextNumberController is invokable.
Status	Pass
Severity	Medium

Test Case ID	NNC-005
Title	__invoke method accepts correct parameters
Objective	Ensure that the __invoke method of NextNumberController accepts four specific parameters: request, invoice, estimate, payment.
Preconditions	- Application running - NextNumberController defined
Test Steps	1. Use reflection to access the __invoke method in NextNumberController. 2. Retrieve the list of method parameters. 3. Verify there are exactly four parameters: request, invoice, estimate, payment.
Test Data	- Method: __invoke - Expected parameters: request, invoice, estimate, payment
Expected Result	- Method has 4 parameters named: request, invoice, estimate, payment.
Actual Result	__invoke method has 4 parameters: request, invoice, estimate, payment.
Status	Pass
Severity	High

Test Case ID	NNC-006
Title	NextNumberController uses SerialNumberFormatter
Objective	Verify that NextNumberController uses SerialNumberFormatter and relevant setter methods before retrieving the next number.
Preconditions	- Application running - NextNumberController code is accessible
Test Steps	1. Obtain file content for NextNumberController. 2. Search for instantiation of SerialNumberFormatter. 3. Search for setCompany, setCustomer, setModel, and getNextNumber method calls in source code.
Test Data	- Source code content with references to SerialNumberFormatter and its setter methods
Expected Result	- File contains 'new SerialNumberFormatter()', '->setCompany', '->setCustomer', '->setModel', and '->getNextNumber()'.
Actual Result	All required usages of SerialNumberFormatter found in the code.
Status	Pass
Severity	High

Test Case ID	NNC-007
Title	NextNumberController handles invoice, estimate, and payment cases

Objective	Confirm that NextNumberController supports all types: invoice, estimate, payment, and default in its switching logic.
Preconditions	- Application running - NextNumberController code is accessible
Test Steps	1. Obtain file content for NextNumberController. 2. Search for switch statement on $key. 3. Verify 'case' statements for 'invoice', 'estimate', 'payment', and 'default'.
Test Data	- Source code content containing switch and case statements
Expected Result	- Code contains 'switch ($key)', 'case 'invoice':', 'case 'estimate':', 'case 'payment':', and 'default:'.
Actual Result	All case statements present and handled correctly.
Status	Pass
Severity	High

Test Case ID	NNC-008
Title	NextNumberController has exception handling
Objective	Ensure that NextNumberController has proper exception handling for errors within main logic.
Preconditions	- Application running - NextNumberController code is accessible
Test Steps	1. Obtain file content for NextNumberController. 2. Search for try-catch block. 3. Check that exceptions are caught and the error message is accessed.
Test Data	- Source code containing try-catch block for exception handling
Expected Result	- Code contains 'try {', 'catch (\Exception $exception)', '$exception->getMessage()'.
Actual Result	Exception handling is implemented as expected.
Status	Pass
Severity	High

Test Case ID	NNC-009
Title	NextNumberController returns JSON response with success and nextNumber
Objective	Validate that NextNumberController returns a well-formed JSON response including fields 'success' and 'nextNumber'.
Preconditions	- Application running - NextNumberController code is accessible
Test Steps	1. Obtain file content for NextNumberController. 2. Search for calls to response()->json(). 3. Verify presence of 'success' => true, 'nextNumber' => $nextNumber, and handling of 'success' => false.
Test Data	- Source code with JSON response structure
Expected Result	- JSON response includes ['success' => true, 'nextNumber' => value] when successful, and ['success' => false] when not.
Actual Result	JSON responses conform to required structure for both success and failure cases.
Status	Pass
Severity	High

File: Note-Test.txt

Test Case ID	N-001
Title	Instantiation of Note Model
Objective	Verify that the Note model can be instantiated successfully.
Preconditions	- Application running - Database seeded - Crater\Models\Note class available
Test Steps	1. Instantiate a new Note object. 2. Verify the object is an instance of Note class.
Test Data	- Input: New Note() - Expected: Object of type Note
Expected Result	- The object created is an instance of the Note class.
Actual Result	The object created is an instance of the Note class.
Status	Pass
Severity	Medium

Test Case ID	N-002
Title	Note Extends Model and Uses HasFactory Trait
Objective	Verify Note model extends Eloquent Model and uses HasFactory trait.
Preconditions	- Application running - Database seeded - Crater\Models\Note class available
Test Steps	1. Instantiate a new Note object. 2. Use reflection to get all traits of Note. 3. Verify Note is an instance of Illuminate\Database\Eloquent\Model. 4. Verify HasFactory trait is in trait list.
Test Data	- Input: ReflectionClass on Note - Expected: Traits list contains HasFactory, and Note extends Model
Expected Result	- Note is an instance of Illuminate\Database\Eloquent\Model. - Traits list contains 'Illuminate\Database\Eloquent\Factories\HasFactory'.
Actual Result	Note is an instance of Illuminate\Database\Eloquent\Model, and traits include HasFactory.
Status	Pass
Severity	Medium

Test Case ID	N-003
Title	Note Model Has Company Relationship
Objective	Ensure the Note model defines a company relationship.
Preconditions	- Application running - Database seeded - Crater\Models\Note class available
Test Steps	1. Instantiate a Note object. 2. Check if method 'company' exists on Note.
Test Data	- Input: method_exists($note, 'company') - Expected: Returns true
Expected Result	- Method 'company' exists in Note model.
Actual Result	Method 'company' exists in Note model.
Status	Pass
Severity	Medium

Test Case ID	N-004
Title	Note Model Scope Methods for Filtering
Objective	Verify Note model has scope methods for filtering: scopeApplyFilters, scopeWhereSearch, scopeWhereType, scopeWhereCompany.
Preconditions	- Application running - Database seeded - Crater\Models\Note class available
Test Steps	1. Use ReflectionClass on Note. 2. Check for existence of 'scopeApplyFilters', 'scopeWhereSearch', 'scopeWhereType', and 'scopeWhereCompany' methods.
Test Data	- Input: ReflectionClass method checks - Expected: All methods exist
Expected Result	- Methods 'scopeApplyFilters', 'scopeWhereSearch', 'scopeWhereType', and 'scopeWhereCompany' all exist in Note model.
Actual Result	All filtering scope methods exist in Note model.
Status	Pass
Severity	Medium

Test Case ID	N-005
Title	scopeApplyFilters in Note Handles Type and Search
Objective	Verify scopeApplyFilters method handles type and search filters.
Preconditions	- Application running - Database seeded - Crater\Models\Note class available
Test Steps	1. Use ReflectionClass on Note. 2. Load file content of Note model. 3. Check for usage of type and search filters inside scopeApplyFilters.
Test Data	- Input: File content inspection for scopeApplyFilters logic - Expected: Contains filters for 'type' and 'search' and uses related scope methods
Expected Result	- File contains `$filters->get('type')`, uses 'whereType'; contains `$filters->get('search')`, uses 'whereSearch'.
Actual Result	File contains type and search filter logic in scopeApplyFilters.
Status	Pass
Severity	Medium

Test Case ID	N-006
Title	scopeWhereSearch Uses LIKE Query in Note Model
Objective	Confirm scopeWhereSearch method uses a SQL LIKE query for name search.
Preconditions	- Application running - Database seeded - Crater\Models\Note class available
Test Steps	1. Use ReflectionClass on Note. 2. Load file content of Note model. 3. Look for 'where' clause using LIKE operator for name field.
Test Data	- Input: Inspection for "where('name', 'LIKE', '%'.$search.'%')" - Expected: File contains SQL LIKE query on 'name'
Expected Result	- File contains `where('name', 'LIKE', '%'.$search.'%')` inside scopeWhereSearch.
Actual Result	LIKE query is present for name search in scopeWhereSearch.

Status	Pass
Severity	Medium

Test Case ID	N-007
Title	Instantiation of NotePolicy
Objective	Verify that the NotePolicy can be instantiated.
Preconditions	- Application running - Crater\Policies\NotePolicy class available
Test Steps	1. Instantiate a new NotePolicy object. 2. Verify the object is an instance of NotePolicy class.
Test Data	- Input: New NotePolicy() - Expected: Object of type NotePolicy
Expected Result	- The object created is an instance of the NotePolicy class.
Actual Result	The object created is an instance of the NotePolicy class.
Status	Pass
Severity	Medium

Test Case ID	N-008
Title	NotePolicy Uses HandlesAuthorization Trait
Objective	Check that NotePolicy uses the HandlesAuthorization trait for authorization.
Preconditions	- Application running - Crater\Policies\NotePolicy class available
Test Steps	1. Use ReflectionClass on NotePolicy. 2. Get all traits used by NotePolicy. 3. Verify that 'Illuminate\Auth\Access\HandlesAuthorization' is present.
Test Data	- Input: Traits list from ReflectionClass - Expected: Contains HandlesAuthorization trait
Expected Result	- Traits list includes 'Illuminate\Auth\Access\HandlesAuthorization'.
Actual Result	HandlesAuthorization trait is present in NotePolicy.
Status	Pass
Severity	High

Test Case ID	N-009
Title	NotePolicy Has manageNotes and viewNotes Methods
Objective	Ensure NotePolicy class defines manageNotes and viewNotes methods.
Preconditions	- Application running - Crater\Policies\NotePolicy class available
Test Steps	1. Use ReflectionClass on NotePolicy. 2. Check for existence of 'manageNotes' method. 3. Check for existence of 'viewNotes' method.
Test Data	- Input: ReflectionClass method checks - Expected: Both methods exist
Expected Result	- Methods 'manageNotes' and 'viewNotes' exist in NotePolicy class.
Actual Result	Both methods exist in NotePolicy class.
Status	Pass
Severity	High

Test Case ID	N-010

Title	NotePolicy Uses BouncerFacade for Authorization
Objective	Ensure that NotePolicy uses BouncerFacade for managing authorization.
Preconditions	- Application running - Crater\Policies\NotePolicy class available
Test Steps	1. Use ReflectionClass on NotePolicy. 2. Load file content of NotePolicy class. 3. Check for usage of BouncerFacade::can for 'manage-all-notes' and 'view-all-notes'.
Test Data	- Input: Inspection for BouncerFacade usage in file - Expected: File contains BouncerFacade::can for the given permissions
Expected Result	- File contains 'BouncerFacade::can('manage-all-notes', Note::class)' - File contains 'BouncerFacade::can('view-all-notes', Note::class)'
Actual Result	BouncerFacade authorization logic is present for both actions.
Status	Pass
Severity	High

Test Case ID	N-011
Title	Instantiation of NoteResource
Objective	Verify that the NoteResource can be instantiated with data.
Preconditions	- Application running - Crater\Http\Resources\NoteResource class available
Test Steps	1. Create a new NoteResource object with an object containing 'id' => 1. 2. Verify the object is an instance of NoteResource.
Test Data	- Input: NoteResource((object)['id' => 1]) - Expected: Object of type NoteResource
Expected Result	- The object created is an instance of NoteResource.
Actual Result	The object created is an instance of NoteResource.
Status	Pass
Severity	Medium

Test Case ID	N-012
Title	NoteResource Extends JsonResource
Objective	Verify that NoteResource extends the JsonResource base resource.
Preconditions	- Application running - Crater\Http\Resources\NoteResource class available
Test Steps	1. Create NoteResource object with input data. 2. Verify object is an instance of Illuminate\Http\Resources\Json\JsonResource.
Test Data	- Input: NoteResource((object)['id' => 1]) - Expected: InstanceOf JsonResource
Expected Result	- NoteResource object is an instance of Illuminate\Http\Resources\Json\JsonResource.
Actual Result	NoteResource is an instance of JsonResource.
Status	Pass
Severity	Medium

Test Case ID	N-013
Title	NoteResource toArray Returns Basic Fields

Objective	Confirm NoteResource toArray method returns basic fields: id, type, name, notes.
Preconditions	- Application running - Crater\Http\Resources\NoteResource class available
Test Steps	1. Use ReflectionClass on NoteResource. 2. Load file content of NoteResource class. 3. Verify inclusion of id, type, name, and notes keys in toArray method.
Test Data	- Input: File content inspection - Expected: All field names present in output array
Expected Result	- File includes: `'id' => $this->id`, `'type' => $this->type`, `'name' => $this->name`, `'notes' => $this->notes` in toArray.
Actual Result	All required fields are present in NoteResource toArray method.
Status	Pass
Severity	Medium

Test Case ID	N-014
Title	NoteResource Conditionally Includes Company Relationship
Objective	Verify that NoteResource includes 'company' relationship based on presence.
Preconditions	- Application running - Crater\Http\Resources\NoteResource class available
Test Steps	1. Use ReflectionClass on NoteResource. 2. Read file content for conditional inclusion of company. 3. Confirm use of $this->when and CompanyResource.
Test Data	- Input: Inspection for 'company' relationship code - Expected: Uses $this->when and CompanyResource for company field
Expected Result	- File contains logic to include 'company' when relationship exists, using CompanyResource.
Actual Result	Conditional inclusion of company relationship verified in NoteResource.
Status	Pass
Severity	Medium

Test Case ID	N-015
Title	NoteResource Namespace Validation
Objective	Validate that NoteResource is defined in correct namespace.
Preconditions	- Application running - Crater\Http\Resources\NoteResource class available
Test Steps	1. Use ReflectionClass on NoteResource. 2. Get the namespace of the class. 3. Verify namespace is 'Crater\Http\Resources'.
Test Data	- Input: ReflectionClass getNamespaceName() - Expected: String 'Crater\Http\Resources'
Expected Result	- Namespace of NoteResource is 'Crater\Http\Resources'.
Actual Result	Namespace is correctly set on NoteResource.
Status	Pass
Severity	Low

Test Case ID	N-016
Title	Instantiation of NotesRequest
Objective	Verify that NotesRequest can be instantiated successfully.

Preconditions	- Application running - Crater\Http\Requests\NotesRequest class available
Test Steps	1. Instantiate a new NotesRequest object. 2. Verify the object is an instance of NotesRequest class.
Test Data	- Input: New NotesRequest() - Expected: Instance of NotesRequest
Expected Result	- The object created is an instance of NotesRequest.
Actual Result	The object created is an instance of NotesRequest.
Status	Pass
Severity	Medium

Test Case ID	N-017
Title	NotesRequest Extends FormRequest
Objective	Ensure NotesRequest class extends Illuminate\Foundation\Http\FormRequest.
Preconditions	- Application running - Crater\Http\Requests\NotesRequest class available
Test Steps	1. Instantiate NotesRequest object. 2. Verify the object is an instance of FormRequest.
Test Data	- Input: New NotesRequest() - Expected: InstanceOf FormRequest
Expected Result	- NotesRequest is an instance of Illuminate\Foundation\Http\FormRequest.
Actual Result	NotesRequest is confirmed as an instance of FormRequest.
Status	Pass
Severity	High

Test Case ID	N-018
Title	NotesRequest Authorize Method Returns True
Objective	Confirm NotesRequest's authorize method returns true.
Preconditions	- Application running - Crater\Http\Requests\NotesRequest class available
Test Steps	1. Instantiate NotesRequest. 2. Call authorize() on request. 3. Verify method returns true.
Test Data	- Input: $request->authorize() - Expected: True
Expected Result	- authorize() method returns true.
Actual Result	authorize() method returns true.
Status	Pass
Severity	High

Test Case ID	N-019
Title	NotesRequest Validation Rules Include Required Fields
Objective	Ensure NotesRequest validation rules include required fields and uniqueness.
Preconditions	- Application running - Crater\Http\Requests\NotesRequest class available
Test Steps	1. Use ReflectionClass on NotesRequest. 2. Load file content of NotesRequest. 3. Check rules for inclusion of 'type', 'name', 'notes' as required. 4. Verify inclusion of Rule::unique('notes').

Test Data	- Input: File content inspection for rules array - Expected: Rules array contains required fields and uniqueness constraint
Expected Result	- Rules array has 'type', 'name', 'notes' required, 'notes' is unique.
Actual Result	Validation rules correctly specify required fields and uniqueness.
Status	Pass
Severity	High

Test Case ID	N-020
Title	NotesRequest Handles PUT Method Differently
Objective	Verify NotesRequest modifies uniqueness rules for PUT requests.
Preconditions	- Application running - Crater\Http\Requests\NotesRequest class available
Test Steps	1. Use ReflectionClass on NotesRequest. 2. Load file content and inspect for conditional logic on HTTP method. 3. Verify usage of isMethod('PUT') and uniqueness rule with ignore.
Test Data	- Input: File content inspection for PUT logic - Expected: Rule::unique with ->ignore for PUT method
Expected Result	- If PUT method, uniqueness rule on 'notes' uses ->ignore($this->route('note')->id).
Actual Result	PUT method is handled with uniqueness rule using ignore.
Status	Pass
Severity	High

Test Case ID	N-021
Title	NotesRequest Has getNotesPayload Method
Objective	Ensure NotesRequest has getNotesPayload for structured input retrieval.
Preconditions	- Application running - Crater\Http\Requests\NotesRequest class available
Test Steps	1. Use ReflectionClass on NotesRequest. 2. Load file content. 3. Verify existence of getNotesPayload method. 4. Confirm it collects validated data and includes company_id from header.
Test Data	- Input: File content inspection for getNotesPayload - Expected: Method exists and includes company_id from header in returned data
Expected Result	- getNotesPayload method exists, collects validated input, and includes 'company_id' from request header.
Actual Result	getNotesPayload correctly retrieves structured data including company_id.
Status	Pass
Severity	High

File: NumberPlaceholdersController-Test.txt

Test Case ID	NPC-001
Title	Instantiation of NumberPlaceholdersController
Objective	Verify that the NumberPlaceholdersController class can be instantiated successfully.
Preconditions	- Application running - Class autoloading enabled - Database seeded (if required for instantiation)
Test Steps	1. Instantiate the NumberPlaceholdersController class. 2. Verify that the created object is an instance of NumberPlaceholdersController.
Test Data	- Instantiation: new NumberPlaceholdersController()
Expected Result	- The instantiated object should be of type NumberPlaceholdersController.
Actual Result	Object was created and recognized as instance of NumberPlaceholdersController.
Status	Pass
Severity	Medium

Test Case ID	NPC-002
Title	NumberPlaceholdersController Extends Base Controller
Objective	Ensure NumberPlaceholdersController extends the base Controller class.
Preconditions	- Application running - Class hierarchy properly defined
Test Steps	1. Instantiate NumberPlaceholdersController. 2. Check if the instance is a subclass of Crater\Http\Controllers\Controller.
Test Data	- Instantiation: new NumberPlaceholdersController()
Expected Result	- NumberPlaceholdersController object should be instance of Controller.
Actual Result	Object was recognized as instance of Crater\Http\Controllers\Controller.
Status	Pass
Severity	Medium

Test Case ID	NPC-003
Title	Namespace Verification for NumberPlaceholdersController
Objective	Verify NumberPlaceholdersController resides in the correct namespace: Crater\Http\Controllers\V1\Admin\General.
Preconditions	- Application running - NumberPlaceholdersController class available
Test Steps	1. Use reflection to inspect NumberPlaceholdersController. 2. Retrieve namespace of the class. 3. Compare with expected namespace.
Test Data	- Class: NumberPlaceholdersController
Expected Result	- Namespace should be "Crater\Http\Controllers\V1\Admin\General".
Actual Result	Namespace returned as "Crater\Http\Controllers\V1\Admin\General".
Status	Pass
Severity	Medium

Test Case ID	NPC-004

Title	Controller Invokable Method Presence
Objective	Confirm NumberPlaceholdersController has an __invoke method (is invokable).
Preconditions	- Application running - NumberPlaceholdersController class available
Test Steps	1. Use reflection to inspect NumberPlaceholdersController. 2. Check if __invoke method exists.
Test Data	- Class: NumberPlaceholdersController
Expected Result	- __invoke method exists.
Actual Result	__invoke method found via reflection.
Status	Pass
Severity	Medium

Test Case ID	NPC-005
Title	Format Parameter Check in Controller
Objective	Ensure NumberPlaceholdersController checks for the ‘format’ parameter and uses SerialNumberFormatter accordingly.
Preconditions	- Application running - NumberPlaceholdersController class and source code available
Test Steps	1. Use reflection to get file path of NumberPlaceholdersController. 2. Read file content. 3. Check file for "if (\$request->format)" condition. 4. Check for call to "SerialNumberFormatter::getPlaceholders(\$request->format)".
Test Data	- File content of NumberPlaceholdersController
Expected Result	- File contains "if (\$request->format)". - File contains "SerialNumberFormatter::getPlaceholders(\$request->format)".
Actual Result	Both strings successfully found in class file content.
Status	Pass
Severity	High

Test Case ID	NPC-006
Title	Empty Placeholders Returned When No Format Given
Objective	Confirm controller returns empty array for placeholders if format parameter is not provided.
Preconditions	- Application running - NumberPlaceholdersController class and source code available
Test Steps	1. Use reflection to get file path of NumberPlaceholdersController. 2. Read file content. 3. Verify existence of "else {" condition handling absence of format. 4. Confirm code returns "$placeholders = []".
Test Data	- File content of NumberPlaceholdersController
Expected Result	- Else condition present for request without format. - "$placeholders = []" assignment exists.
Actual Result	Else branch and assignment code located in file.
Status	Pass
Severity	High

Test Case ID	NPC-007
Title	JSON Response Structure Validation

Objective	Ensure controller returns JSON response including success status and placeholders array.
Preconditions	- Application running - NumberPlaceholdersController class and source code available
Test Steps	1. Use reflection to get file path of NumberPlaceholdersController. 2. Read file content. 3. Confirm JSON response structure with "success" and "placeholders" fields.
Test Data	- File content of NumberPlaceholdersController
Expected Result	- File contains "response()->json([". - File contains "'success' => true". - File contains "'placeholders' => \$placeholders".
Actual Result	All expected JSON response fields found in controller code.
Status	Pass
Severity	High

File: OnboardingWizardController-Test.txt

Test Case ID	OWC-001
Title	Instantiation of OnboardingWizardController
Objective	Verify that OnboardingWizardController can be instantiated successfully.
Preconditions	- Application running - Autoloader configured - Controller classes accessible
Test Steps	1. Instantiate the OnboardingWizardController. 2. Verify the created object is an instance of OnboardingWizardController.
Test Data	- Class: Crater\Http\Controllers\V1\Installation\OnboardingWizardController
Expected Result	- The instantiated object must be of the OnboardingWizardController class.
Actual Result	Object is correctly instantiated as OnboardingWizardController.
Status	Pass
Severity	Medium

Test Case ID	OWC-002
Title	OnboardingWizardController inheritance from base Controller class
Objective	Confirm OnboardingWizardController extends base Controller.
Preconditions	- Application running - Autoloader configured
Test Steps	1. Instantiate the OnboardingWizardController. 2. Assert the instance is also of type Crater\Http\Controllers\Controller.
Test Data	- Class: Crater\Http\Controllers\V1\Installation\OnboardingWizardController
Expected Result	- The instantiated object is an instance of Crater\Http\Controllers\Controller.
Actual Result	OnboardingWizardController is correctly a subclass of Controller.
Status	Pass
Severity	Medium

Test Case ID	OWC-003
Title	Namespace correctness for OnboardingWizardController
Objective	Ensure OnboardingWizardController is contained in the expected namespace.
Preconditions	- Application running - Source code available
Test Steps	1. Use reflection to obtain the namespace name for OnboardingWizardController. 2. Verify the namespace matches 'Crater\Http\Controllers\V1\Installation'.
Test Data	- Class name: OnboardingWizardController
Expected Result	- Namespace for the class is 'Crater\Http\Controllers\V1\Installation'.
Actual Result	Namespace confirmed as 'Crater\Http\Controllers\V1\Installation'.
Status	Pass
Severity	Medium

Test Case ID	OWC-004
Title	Existence and accessibility of getStep method
Objective	Verify OnboardingWizardController defines a public getStep method.

Preconditions	- Application running - Source code accessible
Test Steps	1. Use reflection to check if getStep method exists in OnboardingWizardController. 2. Verify if getStep method is public.
Test Data	- Class name: OnboardingWizardController
Expected Result	- Method 'getStep' exists on OnboardingWizardController. - Method 'getStep' is public.
Actual Result	getStep method found and is public.
Status	Pass
Severity	Medium

Test Case ID	OWC-005
Title	Existence and accessibility of updateStep method
Objective	Confirm OnboardingWizardController defines a public updateStep method.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use reflection to check for updateStep method on OnboardingWizardController. 2. Verify updateStep method is public.
Test Data	- Class name: OnboardingWizardController
Expected Result	- Method 'updateStep' exists on OnboardingWizardController. - Method 'updateStep' is public.
Actual Result	updateStep method found and is public.
Status	Pass
Severity	Medium

Test Case ID	OWC-006
Title	Usage of Storage facade in getStep method
Objective	Ensure getStep method utilizes the Storage facade for data checking.
Preconditions	- Application running - Source code accessible
Test Steps	1. Reflect on OnboardingWizardController class. 2. Read file content for OnboardingWizardController. 3. Check for use of '\\Storage::disk('local')->has('database_created')' in getStep.
Test Data	- Source code file of OnboardingWizardController
Expected Result	- getStep contains call to Storage facade: '\\Storage::disk('local')->has('database_created')'.
Actual Result	Storage facade usage found in getStep.
Status	Pass
Severity	High

Test Case ID	OWC-007
Title	Usage of Setting model in getStep method
Objective	Verify getStep method interacts with Setting model.
Preconditions	- Application running - Source code accessible

Test Steps	1. Reflect on OnboardingWizardController. 2. Read source file for OnboardingWizardController. 3. Check for 'Setting::getSetting('profile_complete')' usage in getStep.
Test Data	- Source code file of OnboardingWizardController
Expected Result	- getStep calls Setting::getSetting('profile_complete').
Actual Result	Setting model call in getStep confirmed.
Status	Pass
Severity	High

Test Case ID	OWC-008
Title	Completed status check in updateStep method
Objective	Validate updateStep checks for 'COMPLETED' status in settings.
Preconditions	- Application running - Source code accessible
Test Steps	1. Reflect on OnboardingWizardController. 2. Read source file content. 3. Confirm existence of status comparison: '$setting === 'COMPLETED'' in updateStep.
Test Data	- Source code file of OnboardingWizardController
Expected Result	- updateStep method checks if $setting equals 'COMPLETED'.
Actual Result	COMPLETED status check found in updateStep.
Status	Pass
Severity	High

Test Case ID	OWC-009
Title	Setting model usage in updateStep method
Objective	Ensure updateStep uses Setting::setSetting to save progress.
Preconditions	- Application running - Source code accessible
Test Steps	1. Reflect on OnboardingWizardController. 2. Read source code file. 3. Verify 'Setting::setSetting('profile_complete', $request->profile_complete)' usage in updateStep.
Test Data	- Source code file of OnboardingWizardController
Expected Result	- updateStep contains Setting::setSetting('profile_complete', $request->profile_complete).
Actual Result	Setting::setSetting used as expected in updateStep.
Status	Pass
Severity	High

Test Case ID	OWC-010
Title	JSON response format in OnboardingWizardController
Objective	Verify OnboardingWizardController responds with JSON containing 'profile_complete'.
Preconditions	- Application running - Source code accessible
Test Steps	1. Reflect on OnboardingWizardController. 2. Read controller file source. 3. Confirm presence of 'response()->json([' and 'profile_complete'' fields in response.

Test Data	- Source code file of OnboardingWizardController
Expected Result	- Controller response uses response()->json([...]). - Response includes 'profile_complete' in returned JSON.
Actual Result	JSON response format with 'profile_complete' found.
Status	Pass
Severity	High

File: OwnerPolicy-Test.txt

Test Case ID	OP-001
Title	Verify managedByOwner returns true for owner user
Objective	Ensure that the OwnerPolicy::managedByOwner() method returns true when provided a user marked as owner.
Preconditions	- Application running - Database seeded - User model available - OwnerPolicy class available
Test Steps	1. Mock a User object and configure it so that isOwner returns true. 2. Instantiate an OwnerPolicy object. 3. Invoke managedByOwner method with the mocked User object. 4. Assert that the result is true.
Test Data	- Input: Mocked User object with isOwner = true - Expected value: managedByOwner() returns true
Expected Result	- The managedByOwner method returns true, confirming the user is recognized as an owner.
Actual Result	The managedByOwner method returned true as expected.
Status	Pass
Severity	High

Test Case ID	OP-002
Title	Verify managedByOwner returns false for non-owner user
Objective	Ensure that the OwnerPolicy::managedByOwner() method returns false when provided a user not marked as owner.
Preconditions	- Application running - Database seeded - User model available - OwnerPolicy class available
Test Steps	1. Mock a User object and configure it so that isOwner returns false. 2. Instantiate an OwnerPolicy object. 3. Invoke managedByOwner method with the mocked User object. 4. Assert that the result is false.
Test Data	- Input: Mocked User object with isOwner = false - Expected value: managedByOwner() returns false
Expected Result	- The managedByOwner method returns false, confirming the user is not recognized as an owner.
Actual Result	The managedByOwner method returned false as expected.
Status	Pass
Severity	High

File: PathToZip-Test.txt

Test Case ID	PTZ-001
Title	Instantiation of PathToZip Rule
Objective	Verify that the PathToZip rule class can be instantiated successfully.
Preconditions	- Application running - Necessary dependencies installed and autoloaded - Database seeded
Test Steps	1. Attempt to instantiate the PathToZip rule class. 2. Check the type of the created instance.
Test Data	- Class: \Crater\Rules\Backup\PathToZip
Expected Result	- The created object is an instance of \Crater\Rules\Backup\PathToZip.
Actual Result	Object is successfully instantiated and verified as the correct class.
Status	Pass
Severity	Medium

Test Case ID	PTZ-002
Title	Validation of valid ZIP file paths
Objective	Verify that the passes() method returns true for valid zip file path inputs.
Preconditions	- Application running - Database seeded - PathToZip rule instantiated
Test Steps	1. Instantiate PathToZip rule. 2. Pass each valid zip file path to the passes() function. 3. Assert that the result is true for each input.
Test Data	- 'backup/my-backup.zip' - 'archive.zip' - 'path/to/dir/file.zip' - '.zip' - 'backup-2023_01_01.zip'
Expected Result	- passes('attribute', 'backup/my-backup.zip') returns true. - passes('attribute', 'archive.zip') returns true. - passes('attribute', 'path/to/dir/file.zip') returns true. - passes('attribute', '.zip') returns true. - passes('attribute', 'backup-2023_01_01.zip') returns true.
Actual Result	All provided zip file paths were accepted and returned true.
Status	Pass
Severity	High

Test Case ID	PTZ-003
Title	Validation of invalid ZIP file paths
Objective	Verify that the passes() method returns false for invalid zip file path inputs.
Preconditions	- Application running - Database seeded - PathToZip rule instantiated
Test Steps	1. Instantiate PathToZip rule. 2. Pass each invalid zip file path to the passes() function. 3. Assert that the result is false for each input.

Test Data	- 'backup/my-backup.tar.gz' - 'document.txt' - 'image.jpg' - 'folder/backup.zip.txt' - 'backup.zip/folder' - 'backup.zip.part' - 'file.ZiP' - 'file.ZIP' - '' - 'no_extension' - 'backupzip' - 'zip' - 'my-backup.'
Expected Result	- passes('attribute', 'backup/my-backup.tar.gz') returns false. - passes('attribute', 'document.txt') returns false. - passes('attribute', 'image.jpg') returns false. - passes('attribute', 'folder/backup.zip.txt') returns false. - passes('attribute', 'backup.zip/folder') returns false. - passes('attribute', 'backup.zip.part') returns false. - passes('attribute', 'file.ZiP') returns false. - passes('attribute', 'file.ZIP') returns false. - passes('attribute', '') returns false. - passes('attribute', 'no_extension') returns false. - passes('attribute', 'backupzip') returns false. - passes('attribute', 'zip') returns false. - passes('attribute', 'my-backup.') returns false.
Actual Result	All provided invalid paths were rejected and returned false.
Status	Pass
Severity	High

Test Case ID	PTZ-004
Title	Verification of validation error message
Objective	Verify that the message() method returns the correct validation error message.
Preconditions	- Application running - Database seeded - PathToZip rule instantiated
Test Steps	1. Instantiate PathToZip rule. 2. Call the message() method. 3. Verify the returned message matches the expected string.
Test Data	- Expected message: 'The given value must be a path to a zip file.'
Expected Result	- message() returns 'The given value must be a path to a zip file.'
Actual Result	Method returned the expected message string.
Status	Pass
Severity	Medium

File: PaymentCollection-Test.txt

Test Case ID	PC-001
Title	PaymentCollection Instantiation Test
Objective	Verify that the PaymentCollection class can be instantiated successfully.
Preconditions	- Application running - All dependencies (Crater\Http\Resources\PaymentCollection, Illuminate\Support\Collection) are installed - PHP environment properly configured
Test Steps	1. Create a new instance of PaymentCollection with an empty Illuminate\Support\Collection. 2. Check if the object is an instance of PaymentCollection.
Test Data	- Input: new PaymentCollection(new Collection([])) - Expected: Instance of PaymentCollection
Expected Result	- The collection object is an instance of PaymentCollection.
Actual Result	PaymentCollection object was successfully instantiated and is of correct type.
Status	Pass
Severity	High

Test Case ID	PC-002
Title	PaymentCollection Extends ResourceCollection
Objective	Ensure that PaymentCollection is a subclass of Illuminate\Http\Resources\Json\ResourceCollection.
Preconditions	- Application running - PaymentCollection dependency loaded
Test Steps	1. Instantiate PaymentCollection with an empty collection. 2. Assert the instance is a subclass of ResourceCollection.
Test Data	- Input: new PaymentCollection(new Collection([])) - Expected: Instance of ResourceCollection
Expected Result	- The object is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	Object is correctly identified as an instance of ResourceCollection.
Status	Pass
Severity	High

Test Case ID	PC-003
Title	PaymentCollection Namespace Verification
Objective	Confirm PaymentCollection resides in the Crater\Http\Resources namespace.
Preconditions	- Application running - PaymentCollection class available
Test Steps	1. Use ReflectionClass to inspect PaymentCollection. 2. Retrieve and verify the namespace.
Test Data	- Input: ReflectionClass(PaymentCollection::class) - Expected: Namespace is 'Crater\Http\Resources'
Expected Result	- ReflectionClass returns 'Crater\Http\Resources' as namespace.
Actual Result	Namespace is correctly identified as 'Crater\Http\Resources'.
Status	Pass
Severity	Medium

Test Case ID	PC-004
Title	PaymentCollection toArray Method Existence
Objective	Verify that the PaymentCollection class contains a method named toArray.
Preconditions	- Application running - PaymentCollection class available
Test Steps	1. Use ReflectionClass to inspect PaymentCollection. 2. Check for existence of toArray method.
Test Data	- Input: ReflectionClass(PaymentCollection::class) - Expected: toArray method exists
Expected Result	- ReflectionClass has a method named 'toArray'.
Actual Result	The toArray method is present in the PaymentCollection class.
Status	Pass
Severity	High

Test Case ID	PC-005
Title	PaymentCollection toArray Method Visibility
Objective	Verify toArray method is public and non-static in PaymentCollection.
Preconditions	- Application running - PaymentCollection class available
Test Steps	1. Use ReflectionClass to inspect PaymentCollection. 2. Inspect the 'toArray' method properties. 3. Check that method is public and not static.
Test Data	- Input: ReflectionClass(PaymentCollection::class) - Expected: toArray is public and non-static
Expected Result	- toArray is public - toArray is not static
Actual Result	toArray method is public and non-static as required.
Status	Pass
Severity	High

Test Case ID	PC-006
Title	PaymentCollection toArray Method Parameter Validation
Objective	Ensure that the toArray method accepts a single 'request' parameter.
Preconditions	- Application running - PaymentCollection class available
Test Steps	1. Use ReflectionClass to inspect PaymentCollection. 2. Retrieve parameters for toArray method. 3. Verify parameter count and name.
Test Data	- Input: ReflectionClass(PaymentCollection::class) - Expected: toArray has one parameter named 'request'
Expected Result	- toArray has 1 parameter: 'request'
Actual Result	toArray method correctly defines a single 'request' parameter.
Status	Pass
Severity	High

Test Case ID	PC-007
Title	PaymentCollection Empty Collection toArray Output
Objective	Validate that toArray returns an empty array when PaymentCollection is instantiated with an empty collection.

Preconditions	- Application running - PaymentCollection and Request class available
Test Steps	1. Instantiate PaymentCollection with an empty collection. 2. Instantiate Request object. 3. Call toArray with the request and check output.
Test Data	- Input: new PaymentCollection(new Collection([])), new Request() - Expected: Empty array
Expected Result	- toArray($request) returns an empty array.
Actual Result	Returned value is an empty array as expected.
Status	Pass
Severity	High

Test Case ID	PC-008
Title	PaymentCollection toArray Parent Delegation
Objective	Verify that PaymentCollection's toArray method delegates to its parent's toArray method.
Preconditions	- Application running - PaymentCollection class source file accessible
Test Steps	1. Use ReflectionClass to locate PaymentCollection file. 2. Read file contents. 3. Confirm presence of 'parent::toArray' invocation.
Test Data	- Input: file_get_contents($reflection->getFileName()) - Expected: Contains 'parent::toArray'
Expected Result	- PaymentCollection source code contains 'parent::toArray'.
Actual Result	Verified 'parent::toArray' exists in source file.
Status	Pass
Severity	High

Test Case ID	PC-009
Title	PaymentCollection File Length Check
Objective	Ensure that the PaymentCollection source file is concise (less than 1000 characters).
Preconditions	- Application running - PaymentCollection class source file accessible
Test Steps	1. Use ReflectionClass to locate PaymentCollection file. 2. Read contents and determine file length. 3. Confirm length is less than 1000 characters.
Test Data	- Input: strlen(file_get_contents($reflection->getFileName())) - Expected: File length < 1000
Expected Result	- PaymentCollection file length is less than 1000 characters.
Actual Result	File length verified as less than 1000 characters.
Status	Pass
Severity	Medium

Test Case ID	PC-010
Title	PaymentCollection Minimal Line Count
Objective	Validate that the PaymentCollection source file contains fewer than 30 lines.
Preconditions	- Application running - PaymentCollection class source file accessible

Test Steps	1. Use ReflectionClass to locate PaymentCollection file. 2. Read file contents and count lines. 3. Confirm line count is less than 30.
Test Data	- Input: count(explode("\n", $fileContent)) - Expected: Line count < 30
Expected Result	- PaymentCollection file has fewer than 30 lines.
Actual Result	Line count verified as less than 30.
Status	Pass
Severity	Medium

File: PaymentMethod-Test.txt

Test Case ID	PM-001
Title	Instantiation of PaymentMethod model
Objective	Verify that the PaymentMethod model can be instantiated correctly.
Preconditions	- Application running - Database seeded - PaymentMethod class available
Test Steps	1. Instantiate a new PaymentMethod object. 2. Check if the object is an instance of PaymentMethod class.
Test Data	- PaymentMethod class instantiation
Expected Result	- The instantiated object is of type PaymentMethod.
Actual Result	The object is successfully instantiated as PaymentMethod.
Status	Pass
Severity	High

Test Case ID	PM-002
Title	PaymentMethod extends Model and uses HasFactory trait
Objective	Confirm that PaymentMethod extends the base Model and utilizes the HasFactory trait.
Preconditions	- Application running - Database seeded - PaymentMethod class available
Test Steps	1. Instantiate a new PaymentMethod object. 2. Use ReflectionClass to inspect the class for parent and traits. 3. Validate that it is an instance of Illuminate\Database\Eloquent\Model. 4. Check for the presence of HasFactory trait.
Test Data	- PaymentMethod class - Trait list
Expected Result	- PaymentMethod is an instance of Model. - PaymentMethod uses the HasFactory trait.
Actual Result	PaymentMethod is a Model and uses HasFactory.
Status	Pass
Severity	High

Test Case ID	PM-003
Title	Presence of TYPE constants in PaymentMethod
Objective	Ensure PaymentMethod has TYPE_GENERAL and TYPE_MODULE constants with correct values.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Access TYPE_GENERAL and TYPE_MODULE constants on PaymentMethod. 2. Validate their values.
Test Data	- PaymentMethod::TYPE_GENERAL ('GENERAL') - PaymentMethod::TYPE_MODULE ('MODULE')
Expected Result	- TYPE_GENERAL is 'GENERAL'. - TYPE_MODULE is 'MODULE'.
Actual Result	Both constants are present with correct values.
Status	Pass

Severity	Medium

Test Case ID	PM-004
Title	Correct casts configuration in PaymentMethod model
Objective	Ensure that 'settings' and 'use_test_env' attributes are correctly casted.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Instantiate a PaymentMethod object. 2. Fetch the casts configuration from the model. 3. Verify 'settings' is cast as 'array' and 'use_test_env' as 'boolean'.
Test Data	- Model cast configuration
Expected Result	- 'settings' key exists and is cast as 'array'. - 'use_test_env' key exists and is cast as 'boolean'.
Actual Result	The casts configuration matches expectations.
Status	Pass
Severity	High

Test Case ID	PM-005
Title	setSettingsAttribute encodes array to JSON
Objective	Confirm that setSettingsAttribute method encodes input arrays as JSON strings.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Instantiate a PaymentMethod object. 2. Call setSettingsAttribute with an array. 3. Check that settings attribute is a JSON string. 4. Decode and validate the contents of the JSON.
Test Data	- Input: ['key' => 'value', 'array' => [1, 2]]
Expected Result	- settings attribute is a JSON string of the array. - Decoded JSON matches the input array.
Actual Result	The settings are encoded as JSON and match the input array.
Status	Pass
Severity	High

Test Case ID	PM-006
Title	setSettingsAttribute handles null input
Objective	Validate that passing null to setSettingsAttribute results in a JSON null.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Instantiate a PaymentMethod object. 2. Call setSettingsAttribute with null. 3. Check that the settings attribute is JSON. 4. Decode JSON and verify it is null.
Test Data	- Input: null
Expected Result	- settings attribute is a JSON null value.
Actual Result	settings attribute is stored and decoded as JSON null.
Status	Pass
Severity	High

Test Case ID	PM-007
Title	setSettingsAttribute handles empty array input
Objective	Validate that passing an empty array results in a proper JSON encoding.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Instantiate a PaymentMethod object. 2. Call setSettingsAttribute with an empty array. 3. Check that settings attribute is JSON. 4. Decode JSON and verify it matches an empty array.
Test Data	- Input: []
Expected Result	- settings attribute is a JSON representation of an empty array.
Actual Result	settings attribute is stored as JSON empty array and decoded correctly.
Status	Pass
Severity	High

Test Case ID	PM-008
Title	Existence and correctness of payments relationship
Objective	Validate the payments() relation exists, is HasMany, and correctly related.
Preconditions	- Application running - PaymentMethod and Payment models available
Test Steps	1. Instantiate a PaymentMethod object. 2. Call payments() to get the relation. 3. Validate it is a HasMany relation. 4. Check related model is Payment. 5. Verify foreign key name is 'payment_method_id'.
Test Data	- PaymentMethod object
Expected Result	- payments() returns HasMany related to Payment with correct foreign key.
Actual Result	payments() relation is HasMany with Payment and correct key.
Status	Pass
Severity	High

Test Case ID	PM-009
Title	Existence and correctness of expenses relationship
Objective	Validate the expenses() relation exists, is HasMany, and correctly related.
Preconditions	- Application running - PaymentMethod and Expense models available
Test Steps	1. Instantiate a PaymentMethod object. 2. Call expenses() to get the relation. 3. Validate it is a HasMany relation. 4. Check related model is Expense. 5. Verify foreign key name is 'payment_method_id'.
Test Data	- PaymentMethod object
Expected Result	- expenses() returns HasMany related to Expense with correct foreign key.
Actual Result	expenses() relation is HasMany with Expense and correct key.
Status	Pass
Severity	High

Test Case ID	PM-010
Title	Existence and correctness of company relationship

Objective	Validate the company() relation exists, is BelongsTo, and correctly related.
Preconditions	- Application running - PaymentMethod and Company models available
Test Steps	1. Instantiate a PaymentMethod object. 2. Call company() to get the relation. 3. Validate it is a BelongsTo relation. 4. Check related model is Company. 5. Verify foreign key name is 'company_id'.
Test Data	- PaymentMethod object
Expected Result	- company() returns BelongsTo related to Company with correct foreign key.
Actual Result	company() relation is BelongsTo with Company and correct key.
Status	Pass
Severity	High

Test Case ID	PM-011
Title	Presence of scope methods in PaymentMethod
Objective	Ensure all necessary Eloquent scope methods exist in PaymentMethod.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Use ReflectionClass to inspect PaymentMethod methods. 2. Check for existence of: - scopeWhereCompanyId - scopeWhereCompany - scopeWherePaymentMethod - scopeWhereSearch - scopeApplyFilters - scopePaginateData
Test Data	- PaymentMethod class
Expected Result	- All six scope methods are present in PaymentMethod.
Actual Result	All specified scope methods exist in PaymentMethod.
Status	Pass
Severity	High

Test Case ID	PM-012
Title	scopeWhereSearch uses LIKE query on name attribute
Objective	Verify that the scopeWhereSearch function performs a LIKE query on the 'name' attribute.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Use ReflectionClass to find PaymentMethod source file. 2. Read file contents. 3. Confirm that WHERE with 'name' LIKE query exists.
Test Data	- fileContents of PaymentMethod
Expected Result	- The code contains: where('name', 'LIKE', '%'.$search.'%')
Actual Result	LIKE query on 'name' with search variable is present.
Status	Pass
Severity	High

Test Case ID	PM-013

Title	scopeApplyFilters processes method_id filter and delegates to wherePaymentMethod
Objective	Ensure that scopeApplyFilters correctly filters by method_id and calls wherePaymentMethod.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Use ReflectionClass to find PaymentMethod source file. 2. Read file contents. 3. Check for filter logic based on method_id. 4. Verify that delegation to wherePaymentMethod occurs.
Test Data	- fileContents of PaymentMethod
Expected Result	- The code contains: $filters->get('method_id') - The code delegates to ->wherePaymentMethod
Actual Result	method_id filter and delegation are correctly handled.
Status	Pass
Severity	High

Test Case ID	PM-014
Title	scopePaginateData correctly handles 'all' limit option
Objective	Confirm that scopePaginateData works with 'all' limit and returns proper results.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Use ReflectionClass to find PaymentMethod source file. 2. Read file contents. 3. Check for logic: if limit == 'all', return query->get(). 4. Otherwise, return query->paginate(limit).
Test Data	- fileContents of PaymentMethod
Expected Result	- Query returns all records if limit is 'all'. - Otherwise, paginated records are returned.
Actual Result	Pagination logic and 'all' limit handling are present.
Status	Pass
Severity	High

Test Case ID	PM-015
Title	Existence and correctness of static createPaymentMethod method
Objective	Validate that createPaymentMethod exists, is static and public.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Use ReflectionClass to validate method presence. 2. Confirm createPaymentMethod is static. 3. Confirm createPaymentMethod is public.
Test Data	- PaymentMethod class
Expected Result	- createPaymentMethod method exists and is public static.
Actual Result	createPaymentMethod is present and correctly defined.
Status	Pass
Severity	High

Test Case ID	PM-016
Title	Existence and correctness of static getSettings method

Objective	Validate that getSettings exists, is static and public.
Preconditions	- Application running - PaymentMethod class available
Test Steps	1. Use ReflectionClass to validate method presence. 2. Confirm getSettings is static. 3. Confirm getSettings is public.
Test Data	- PaymentMethod class
Expected Result	- getSettings method exists and is public static.
Actual Result	getSettings is present and correctly defined.
Status	Pass
Severity	High

File: PaymentMethods-Test.txt

Test Case ID	PM-001
Title	Instantiation of PaymentMethodCollection
Objective	Verify that PaymentMethodCollection can be successfully instantiated.
Preconditions	- Application running - PaymentMethodCollection class available - Collection class available
Test Steps	1. Initialize a new empty Collection. 2. Instantiate PaymentMethodCollection by passing the Collection object. 3. Check the resulting object's class.
Test Data	- Input: new Collection([]) - Expected: Object should be instance of PaymentMethodCollection
Expected Result	PaymentMethodCollection object is instantiated and is an instance of PaymentMethodCollection class.
Actual Result	PaymentMethodCollection object instantiated as expected.
Status	Pass
Severity	Medium

Test Case ID	PM-002
Title	PaymentMethodCollection extends ResourceCollection
Objective	Ensure that PaymentMethodCollection inherits from ResourceCollection.
Preconditions	- Application running - PaymentMethodCollection and ResourceCollection classes available
Test Steps	1. Instantiate a new PaymentMethodCollection with an empty Collection. 2. Verify the class inheritance of the PaymentMethodCollection object.
Test Data	- Input: PaymentMethodCollection(new Collection([])) - Expected: Object should be an instance of Illuminate\Http\Resources\Json\ResourceCollection
Expected Result	PaymentMethodCollection object is an instance of ResourceCollection.
Actual Result	Verified PaymentMethodCollection is an instance of ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	PM-003
Title	PaymentMethodCollection returns empty array for empty collection
Objective	Verify that converting an empty PaymentMethodCollection to array returns an empty array.
Preconditions	- Application running - PaymentMethodCollection and Request classes available
Test Steps	1. Instantiate a new Request object. 2. Instantiate a PaymentMethodCollection with an empty Collection. 3. Call toArray() method on PaymentMethodCollection, passing the Request. 4. Verify that the result is an empty array.
Test Data	- Input: new Collection([]), Request() - Expected: toArray() returns []
Expected Result	toArray() returns an empty array when PaymentMethodCollection is empty.
Actual Result	toArray() returned an empty array as expected.
Status	Pass
Severity	Medium

Test Case ID	PM-004
Title	PaymentMethodCollection delegates to parent toArray method
Objective	Ensure PaymentMethodCollection uses parent toArray implementation.
Preconditions	- Application running - PaymentMethodCollection class available
Test Steps	1. Reflect on PaymentMethodCollection to get its file content. 2. Check if the code contains a call to parent::toArray.
Test Data	- Input: PaymentMethodCollection source code - Expected: source code contains 'parent::toArray'
Expected Result	PaymentMethodCollection delegates to parent toArray method.
Actual Result	Source code contains 'parent::toArray' as expected.
Status	Pass
Severity	Medium

Test Case ID	PM-005
Title	Instantiation of PaymentMethodsController
Objective	Verify that PaymentMethodsController can be successfully instantiated.
Preconditions	- Application running - PaymentMethodsController class available
Test Steps	1. Instantiate a PaymentMethodsController object. 2. Validate object's class.
Test Data	- Input: new PaymentMethodsController() - Expected: Object should be instance of PaymentMethodsController
Expected Result	PaymentMethodsController object is instantiated and is an instance of PaymentMethodsController class.
Actual Result	PaymentMethodsController object instantiated as expected.
Status	Pass
Severity	High

Test Case ID	PM-006
Title	PaymentMethodsController extends base Controller
Objective	Ensure PaymentMethodsController inherits from base Controller class.
Preconditions	- Application running - PaymentMethodsController and base Controller classes available
Test Steps	1. Instantiate PaymentMethodsController. 2. Verify that the object is an instance of Crater\Http\Controllers\Controller.
Test Data	- Input: PaymentMethodsController instance - Expected: instance of Crater\Http\Controllers\Controller
Expected Result	PaymentMethodsController inherits from Controller class.
Actual Result	Verified PaymentMethodsController is an instance of Controller.
Status	Pass
Severity	High

Test Case ID	PM-007
Title	PaymentMethodsController has CRUD methods
Objective	Confirm that PaymentMethodsController defines all CRUD operations.

Preconditions	- Application running - PaymentMethodsController class available
Test Steps	1. Use reflection to inspect PaymentMethodsController. 2. Check for existence of methods: index, store, show, update, destroy.
Test Data	- Input: PaymentMethodsController class - Expected: Methods 'index', 'store', 'show', 'update', 'destroy' exist
Expected Result	PaymentMethodsController includes all required CRUD methods.
Actual Result	All CRUD methods found in PaymentMethodsController.
Status	Pass
Severity	High

Test Case ID	PM-008
Title	PaymentMethodsController index method uses authorization
Objective	Ensure the index method authorizes access to PaymentMethods listing.
Preconditions	- Application running - PaymentMethodsController source code available
Test Steps	1. Get the source code for PaymentMethodsController class. 2. Check for the presence of '$this->authorize('viewAny', PaymentMethod::class)' in index method.
Test Data	- Input: PaymentMethodsController source code - Expected: Contains authorization call for 'viewAny', PaymentMethod::class
Expected Result	Authorization check exists in index method of PaymentMethodsController.
Actual Result	Authorization code present in index method.
Status	Pass
Severity	High

Test Case ID	PM-009
Title	PaymentMethodsController index filters by TYPE_GENERAL and company
Objective	Confirm filtering logic in PaymentMethodsController index by type and company.
Preconditions	- Application running - PaymentMethodsController source code available
Test Steps	1. Retrieve PaymentMethodsController source code. 2. Check for filtering by 'type' with PaymentMethod::TYPE_GENERAL. 3. Check for filtering by company.
Test Data	- Input: PaymentMethodsController source code - Expected: Filtering code includes '->where('type', PaymentMethod::TYPE_GENERAL)' and '->whereCompany()'
Expected Result	Index method filters payment methods by type and company.
Actual Result	Filtering by type and company confirmed in source code.
Status	Pass
Severity	High

Test Case ID	PM-010
Title	PaymentMethodsController destroy checks for attached payments and expenses
Objective	Ensure PaymentMethodsController prevents deletion if payments or expenses are attached.
Preconditions	- Application running - PaymentMethodsController source code available

Test Steps	1. Get PaymentMethodsController source code. 2. Check for code handling attached payments via payments()->exists(). 3. Check for code handling attached expenses via expenses()->exists(). 4. Confirm responding via respondJson.
Test Data	- Input: PaymentMethodsController source code - Expected: Contains conditional checks for payments and expenses and respondJson calls
Expected Result	Controller checks for attached payments/expenses and prevents deletion.
Actual Result	Attached payments and expenses checks present in destroy method.
Status	Pass
Severity	High

Test Case ID	PM-011
Title	PaymentMethodsController uses PaymentMethodRequest for store and update
Objective	Verify that store and update actions require PaymentMethodRequest as parameter.
Preconditions	- Application running - PaymentMethodsController and PaymentMethodRequest classes available
Test Steps	1. Use reflection to get store and update methods from PaymentMethodsController. 2. Verify the first parameter type for both is PaymentMethodRequest.
Test Data	- Input: PaymentMethodsController methods - Expected: Both methods use PaymentMethodRequest as first parameter
Expected Result	Both store and update methods use PaymentMethodRequest.
Actual Result	Confirmed PaymentMethodRequest used in both methods.
Status	Pass
Severity	High

Test Case ID	PM-012
Title	Instantiation of PaymentMethodPolicy
Objective	Verify that PaymentMethodPolicy can be successfully instantiated.
Preconditions	- Application running - PaymentMethodPolicy class available
Test Steps	1. Instantiate PaymentMethodPolicy. 2. Validate object's class.
Test Data	- Input: new PaymentMethodPolicy() - Expected: Object should be instance of PaymentMethodPolicy
Expected Result	PaymentMethodPolicy object is instantiated and is an instance of PaymentMethodPolicy class.
Actual Result	PaymentMethodPolicy object instantiated as expected.
Status	Pass
Severity	High

Test Case ID	PM-013
Title	PaymentMethodPolicy uses HandlesAuthorization trait
Objective	Verify that PaymentMethodPolicy class uses the HandlesAuthorization trait.
Preconditions	- Application running - PaymentMethodPolicy class available

Test Steps	1. Use reflection to get trait names from PaymentMethodPolicy. 2. Check for 'Illuminate\Auth\Access\HandlesAuthorization' in traits list.
Test Data	- Input: PaymentMethodPolicy traits - Expected: Trait list includes HandlesAuthorization
Expected Result	PaymentMethodPolicy includes HandlesAuthorization trait.
Actual Result	HandlesAuthorization trait verified in PaymentMethodPolicy.
Status	Pass
Severity	High

Test Case ID	PM-014
Title	PaymentMethodPolicy has all authorization methods
Objective	Validate the presence of required authorization methods in PaymentMethodPolicy.
Preconditions	- Application running - PaymentMethodPolicy class available
Test Steps	1. Use reflection to inspect PaymentMethodPolicy methods. 2. Verify existence of methods: viewAny, view, create, update, delete, restore, forceDelete.
Test Data	- Input: PaymentMethodPolicy class - Expected: All seven methods exist
Expected Result	PaymentMethodPolicy contains all main authorization methods.
Actual Result	All authorization methods found in PaymentMethodPolicy.
Status	Pass
Severity	High

Test Case ID	PM-015
Title	PaymentMethodPolicy uses BouncerFacade for authorization checks
Objective	Confirm that PaymentMethodPolicy relies on BouncerFacade for authorizations.
Preconditions	- Application running - PaymentMethodPolicy and BouncerFacade classes available
Test Steps	1. Retrieve PaymentMethodPolicy source code. 2. Check for BouncerFacade usage, specifically BouncerFacade::can('view-payment', Payment::class).
Test Data	- Input: PaymentMethodPolicy source code - Expected: Contains BouncerFacade::can('view-payment', Payment::class)
Expected Result	BouncerFacade utilized for authorization in PaymentMethodPolicy.
Actual Result	BouncerFacade usage found in source code.
Status	Pass
Severity	High

Test Case ID	PM-016
Title	PaymentMethodPolicy checks company ownership
Objective	Ensure company ownership verified during authorizations in PaymentMethodPolicy.
Preconditions	- Application running - PaymentMethodPolicy class available
Test Steps	1. Review PaymentMethodPolicy source code. 2. Look for code checking '$user->hasCompany($paymentMethod->company_id)'.

Test Data	- Input: PaymentMethodPolicy source code - Expected: Contains company ownership check
Expected Result	Company ownership is checked in authorization logic.
Actual Result	Company ownership check present in PaymentMethodPolicy.
Status	Pass
Severity	High

Test Case ID	PM-017
Title	Instantiation of PaymentMethodRequest
Objective	Verify that PaymentMethodRequest can be successfully instantiated.
Preconditions	- Application running - PaymentMethodRequest class available
Test Steps	1. Instantiate PaymentMethodRequest. 2. Validate object's class.
Test Data	- Input: new PaymentMethodRequest() - Expected: Object should be instance of PaymentMethodRequest
Expected Result	PaymentMethodRequest object is instantiated as expected.
Actual Result	PaymentMethodRequest object instantiated as expected.
Status	Pass
Severity	Medium

Test Case ID	PM-018
Title	PaymentMethodRequest extends FormRequest
Objective	Ensure PaymentMethodRequest inherits from FormRequest.
Preconditions	- Application running - PaymentMethodRequest and FormRequest classes available
Test Steps	1. Instantiate PaymentMethodRequest. 2. Validate it inherits from Illuminate\Foundation\Http\FormRequest.
Test Data	- Input: PaymentMethodRequest instance - Expected: Instance of FormRequest
Expected Result	PaymentMethodRequest is an instance of FormRequest.
Actual Result	PaymentMethodRequest verified as instance of FormRequest.
Status	Pass
Severity	Medium

Test Case ID	PM-019
Title	PaymentMethodRequest authorize returns true
Objective	Verify that authorize() method on PaymentMethodRequest returns true.
Preconditions	- Application running - PaymentMethodRequest class available
Test Steps	1. Instantiate PaymentMethodRequest. 2. Call authorize() on the instance. 3. Verify the return value is true.
Test Data	- Input: PaymentMethodRequest - Expected: authorize() returns true
Expected Result	authorize() method returns true.
Actual Result	authorize() returned true as expected.
Status	Pass

Severity	High

Test Case ID	PM-020
Title	PaymentMethodRequest rules contain unique name validation by company
Objective	Confirm validation rules include unique name validation scoped by company.
Preconditions	- Application running - PaymentMethodRequest class available
Test Steps	1. Retrieve PaymentMethodRequest source code. 2. Locate rules for 'name' field. 3. Verify presence of Rule::unique('payment_methods') and where('company_id', $this->header('company')).
Test Data	- Input: PaymentMethodRequest source code - Expected: Rules contain unique validation for name scoped to company
Expected Result	Unique name validation for payment methods is scoped by company in rules.
Actual Result	Unique name validation found in rules as expected.
Status	Pass
Severity	High

Test Case ID	PM-021
Title	PaymentMethodRequest getPaymentMethodPayload merges company_id and type
Objective	Ensure getPaymentMethodPayload merges company_id and type into payload.
Preconditions	- Application running - PaymentMethodRequest class available
Test Steps	1. Retrieve PaymentMethodRequest source code. 2. Locate getPaymentMethodPayload method. 3. Confirm merging of 'company_id' from header and 'type' as PaymentMethod::TYPE_GENERAL.
Test Data	- Input: PaymentMethodRequest source code - Expected: getPaymentMethodPayload merges 'company_id', 'type'
Expected Result	getPaymentMethodPayload merges company_id and type as expected.
Actual Result	Merging of company_id and type present in getPaymentMethodPayload.
Status	Pass
Severity	High

File: PdfMiddleware-Test.txt

Test Case ID	PM-001
Title	Middleware allows request when authenticated via web guard
Objective	Verify that the PdfMiddleware allows further request processing if the user is authenticated via the 'web' guard, and does not check other guards due to short-circuiting logic.
Preconditions	- Application running - Database seeded - PDF middleware is registered - User session is present (web guard) - Required authentication system available (Laravel Auth)
Test Steps	1. Mock the authentication 'web' guard to return true for check(). 2. Ensure 'sanctum' and 'customer' guards are never checked. 3. Create a mock HTTP Request object. 4. Pass the request through the PdfMiddleware handle method with a mock next closure. 5. Observe the result returned by the middleware.
Test Data	- Auth::guard('web')->check() returns: true - Auth::guard('sanctum')->check() and Auth::guard('customer')->check() are never called. - Request Object: Mocked - Response Object: Mocked
Expected Result	- The middleware calls the next closure and returns its response. - The result is exactly the same as the mock Response object.
Actual Result	- The middleware calls the next closure and returns the mock Response object as expected.
Status	Pass
Severity	High

Test Case ID	PM-002
Title	Middleware allows request when authenticated via sanctum guard (web not authenticated)
Objective	Verify that the PdfMiddleware allows further request processing if the user is authenticated via the 'sanctum' guard when 'web' guard is not authenticated, and does not check the 'customer' guard.
Preconditions	- Application running - Database seeded - PDF middleware is registered - Sanctum authentication is configured - Required authentication system available (Laravel Auth)
Test Steps	1. Mock the authentication 'web' guard to return false for check(). 2. Mock the 'sanctum' guard to return true for check(). 3. Ensure the 'customer' guard is not checked. 4. Create a mock HTTP Request object. 5. Pass the request through the PdfMiddleware handle method with a mock next closure. 6. Observe the result returned by the middleware.
Test Data	- Auth::guard('web')->check() returns: false - Auth::guard('sanctum')->check() returns: true - Auth::guard('customer')->check() is never called. - Request Object: Mocked - Response Object: Mocked
Expected Result	- The middleware calls the next closure and returns its response. - The result is exactly the same as the mock Response object.
Actual Result	- The middleware calls the next closure and returns the mock Response object as expected.

Status	Pass
Severity	High

Test Case ID	PM-003
Title	Middleware allows request when authenticated via customer guard only
Objective	Verify that the PdfMiddleware allows further request processing if the user is authenticated via the 'customer' guard when both 'web' and 'sanctum' guards are not authenticated.
Preconditions	- Application running - Database seeded - PDF middleware is registered - Customer authentication is configured - Required authentication system available (Laravel Auth)
Test Steps	1. Mock the 'web' guard to return false for check(). 2. Mock the 'sanctum' guard to return false for check(). 3. Mock the 'customer' guard to return true for check(). 4. Create a mock HTTP Request object. 5. Pass the request through the PdfMiddleware handle method with a mock next closure. 6. Observe the result returned by the middleware.
Test Data	- Auth::guard('web')->check() returns: false - Auth::guard('sanctum')->check() returns: false - Auth::guard('customer')->check() returns: true - Request Object: Mocked - Response Object: Mocked
Expected Result	- The middleware calls the next closure and returns its response. - The result is exactly the same as the mock Response object.
Actual Result	- The middleware calls the next closure and returns the mock Response object as expected.
Status	Pass
Severity	High

File: ProfileControllerAndRequest-Test.txt

Test Case ID	PCAR-001
Title	Instantiation of ProfileController
Objective	Verify that the ProfileController class can be instantiated correctly.
Preconditions	- Application running - Autoloaded classes available - No constructor dependencies blocking instantiation
Test Steps	1. Attempt to instantiate the ProfileController. 2. Check that the controller object is created.
Test Data	- Instance creation: new ProfileController() - Expected value: Instance should be of type ProfileController
Expected Result	ProfileController object is instantiated successfully and is an instance of ProfileController.
Actual Result	ProfileController object is instantiated and is an instance of ProfileController.
Status	Pass
Severity	High

Test Case ID	PCAR-002
Title	ProfileController Extends Base Controller
Objective	Ensure ProfileController inherits from the main Controller class.
Preconditions	- Application running - ProfileController autoloaded
Test Steps	1. Instantiate the ProfileController. 2. Check its type inheritance for Controller.
Test Data	- Instance: new ProfileController() - Expected class: Crater\Http\Controllers\Controller
Expected Result	ProfileController instance is also an instance of Controller.
Actual Result	ProfileController instance is also an instance of Controller.
Status	Pass
Severity	High

Test Case ID	PCAR-003
Title	ProfileController Namespace Correctness
Objective	Confirm that ProfileController is declared in the correct namespace.
Preconditions	- Application running - Source files unchanged
Test Steps	1. Use ReflectionClass on ProfileController. 2. Retrieve and inspect namespace string.
Test Data	- Class: ProfileController - Expected namespace: Crater\Http\Controllers\V1\Customer\General
Expected Result	The namespace of ProfileController is Crater\Http\Controllers\V1\Customer\General.
Actual Result	The namespace of ProfileController is Crater\Http\Controllers\V1\Customer\General.
Status	Pass
Severity	Medium

Test Case ID	PCAR-004

Title	Presence and Visibility of updateProfile Method
Objective	Verify that the updateProfile method exists and is public in ProfileController.
Preconditions	- Application running - Source code unmodified
Test Steps	1. Use ReflectionClass to inspect ProfileController. 2. Check if 'updateProfile' method exists. 3. Verify that the method is public.
Test Data	- Class: ProfileController - Method: updateProfile
Expected Result	updateProfile method exists and is public in ProfileController.
Actual Result	updateProfile method exists and is public in ProfileController.
Status	Pass
Severity	High

Test Case ID	PCAR-005
Title	Presence and Visibility of getUser Method
Objective	Verify that the getUser method exists and is public in ProfileController.
Preconditions	- Application running - Source code unmodified
Test Steps	1. Use ReflectionClass to inspect ProfileController. 2. Check if 'getUser' method exists. 3. Verify that the method is public.
Test Data	- Class: ProfileController - Method: getUser
Expected Result	getUser method exists and is public in ProfileController.
Actual Result	getUser method exists and is public in ProfileController.
Status	Pass
Severity	High

Test Case ID	PCAR-006
Title	Usage of Customer Auth Guard in updateProfile
Objective	Verify that updateProfile uses the 'customer' authentication guard.
Preconditions	- Application running - Source code available
Test Steps	1. Inspect ProfileController source code. 2. Search for usage of Auth::guard('customer')->user() in updateProfile.
Test Data	- Guard: 'customer' - Function: updateProfile
Expected Result	updateProfile uses Auth::guard('customer')->user() for authentication.
Actual Result	updateProfile uses Auth::guard('customer')->user() for authentication.
Status	Pass
Severity	High

Test Case ID	PCAR-007
Title	Avatar Removal and Upload in updateProfile
Objective	Ensure updateProfile handles avatar removal and new uploads as intended.
Preconditions	- Application running - ProfileController and related media libraries loaded

Test Steps	1. Inspect updateProfile function source code. 2. Identify usage of 'is_customer_avatar_removed' logic. 3. Check for clearMediaCollection('customer_avatar'). 4. Verify handling for hasFile('customer_avatar') and addMediaFromRequest('customer_avatar') logic.
Test Data	- Flags: is_customer_avatar_removed - Methods: clearMediaCollection, hasFile, addMediaFromRequest
Expected Result	updateProfile handles both avatar removal and new file uploads using correct methods/logic.
Actual Result	updateProfile handles avatar removal and new file uploads correctly.
Status	Pass
Severity	High

Test Case ID	PCAR-008
Title	Handling Billing and Shipping Addresses in updateProfile
Objective	Ensure updateProfile addresses correct handling of billing and shipping addresses.
Preconditions	- Application running - Database seeded with address entities
Test Steps	1. Inspect updateProfile code for billing/shipping logic. 2. Confirm branch for $request->billing !== null and $request->shipping !== null. 3. Check for relevant address deletion and retrieval methods. 4. Confirm call to getShippingAddress and getBillingAddress.
Test Data	- Request fields: billing, shipping - Methods: getBillingAddress, getShippingAddress, billingAddress()->delete(), shippingAddress()->delete()
Expected Result	updateProfile updates, deletes, and retrieves billing/shipping addresses using correct logic.
Actual Result	updateProfile updates, deletes, and retrieves billing/shipping addresses correctly.
Status	Pass
Severity	High

Test Case ID	PCAR-009
Title	Returns CustomerResource in updateProfile
Objective	Validate that updateProfile returns a CustomerResource object.
Preconditions	- Application running - CustomerResource class available
Test Steps	1. Inspect updateProfile implementation for return value. 2. Confirm presence of 'return new CustomerResource($customer)'.
Test Data	- Return type: CustomerResource
Expected Result	updateProfile returns new CustomerResource object as response.
Actual Result	updateProfile returns new CustomerResource object.
Status	Pass
Severity	High

Test Case ID	PCAR-010
Title	Instantiation of CustomerProfileRequest
Objective	Confirm the CustomerProfileRequest class can be instantiated.

Preconditions	- Application running - Autoloaded classes available
Test Steps	1. Attempt to instantiate CustomerProfileRequest. 2. Confirm object is created.
Test Data	- Instance creation: new CustomerProfileRequest() - Expected class: CustomerProfileRequest
Expected Result	CustomerProfileRequest object is instantiated and is an instance of CustomerProfileRequest.
Actual Result	CustomerProfileRequest object is instantiated and is an instance of CustomerProfileRequest.
Status	Pass
Severity	High

Test Case ID	PCAR-011
Title	CustomerProfileRequest Extends FormRequest
Objective	Ensure CustomerProfileRequest inherits from FormRequest.
Preconditions	- Application running - Proper framework loaded
Test Steps	1. Instantiate CustomerProfileRequest. 2. Check its type inheritance for FormRequest.
Test Data	- Instance: new CustomerProfileRequest() - Expected class: Illuminate\Foundation\Http\FormRequest
Expected Result	CustomerProfileRequest instance is also an instance of FormRequest.
Actual Result	CustomerProfileRequest instance is also an instance of FormRequest.
Status	Pass
Severity	High

Test Case ID	PCAR-012
Title	CustomerProfileRequest Namespace Correctness
Objective	Confirm that CustomerProfileRequest is declared in the correct namespace.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass on CustomerProfileRequest. 2. Inspect namespace string.
Test Data	- Class: CustomerProfileRequest - Expected namespace: Crater\Http\Requests\Customer
Expected Result	The namespace of CustomerProfileRequest is Crater\Http\Requests\Customer.
Actual Result	The namespace of CustomerProfileRequest is Crater\Http\Requests\Customer.
Status	Pass
Severity	Medium

Test Case ID	PCAR-013
Title	CustomerProfileRequest Authorization Passes
Objective	Ensure authorize method returns true always.
Preconditions	- Application running
Test Steps	1. Instantiate CustomerProfileRequest. 2. Call authorize() method. 3. Check returned value.

Test Data	- Method: authorize() - Expected: true
Expected Result	authorize() method returns true.
Actual Result	authorize() method returns true.
Status	Pass
Severity	High

Test Case ID	PCAR-014
Title	Required Methods on CustomerProfileRequest
Objective	Validate existence of key methods: authorize, rules, getShippingAddress, getBillingAddress.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to inspect CustomerProfileRequest. 2. Check for methods: authorize, rules, getShippingAddress, getBillingAddress.
Test Data	- Methods: authorize, rules, getShippingAddress, getBillingAddress
Expected Result	All the required methods exist in CustomerProfileRequest.
Actual Result	All the required methods exist in CustomerProfileRequest.
Status	Pass
Severity	High

Test Case ID	PCAR-015
Title	Profile Field Rules in CustomerProfileRequest
Objective	Ensure 'name', 'password', and 'email' profile validation rules are present including uniqueness.
Preconditions	- Application running
Test Steps	1. Inspect rules() method in CustomerProfileRequest. 2. Look for validation rules for 'name', 'password', 'email'. 3. Confirm usage of Rule::unique('customers') for 'email'.
Test Data	- Rule check for: name, password, email, Rule::unique('customers')
Expected Result	'Profile' rules for name, password, and email present, with unique check for email.
Actual Result	'Profile' rules for name, password, and email are present with uniqueness check.
Status	Pass
Severity	High

Test Case ID	PCAR-016
Title	Billing and Shipping Address Rule Validation in CustomerProfileRequest
Objective	Ensure validation rules are present for billing/shipping address fields.
Preconditions	- Application running
Test Steps	1. Inspect rules() method in CustomerProfileRequest. 2. Confirm rules for 'billing.name', 'billing.address_street_1', 'shipping.name', 'shipping.address_street_1'.
Test Data	- Fields: billing.name, billing.address_street_1, shipping.name, shipping.address_street_1
Expected Result	Validation rules exist for fields in billing and shipping addresses.
Actual Result	Validation rules exist for fields in billing and shipping addresses.

Status	Pass
Severity	High

Test Case ID	PCAR-017
Title	Customer Avatar Validation Rule in CustomerProfileRequest
Objective	Ensure customer_avatar field has file validation, mime type restrictions, and max size.
Preconditions	- Application running
Test Steps	1. Inspect rules() for 'customer_avatar' field. 2. Check for validation rules: 'file', 'mimes:gif,jpg,png', 'max:20000'.
Test Data	- Field: customer_avatar - Rules: file, mimes:gif,jpg,png, max:20000
Expected Result	customer_avatar field enforces file type, mime, and size restrictions.
Actual Result	customer_avatar field enforces file type, mime, and size restrictions.
Status	Pass
Severity	High

Test Case ID	PCAR-018
Title	Shipping Address Type Merge in getShippingAddress
Objective	Verify getShippingAddress merges type field as Address::SHIPPING_TYPE.
Preconditions	- Application running
Test Steps	1. Inspect getShippingAddress method in CustomerProfileRequest. 2. Confirm steady merge of type => Address::SHIPPING_TYPE.
Test Data	- Field merged: 'type' => Address::SHIPPING_TYPE
Expected Result	getShippingAddress merges 'type' with SHIPPING_TYPE appropriately.
Actual Result	getShippingAddress merges 'type' with SHIPPING_TYPE appropriately.
Status	Pass
Severity	High

Test Case ID	PCAR-019
Title	Billing Address Type Merge in getBillingAddress
Objective	Verify getBillingAddress merges type field as Address::BILLING_TYPE.
Preconditions	- Application running
Test Steps	1. Inspect getBillingAddress method in CustomerProfileRequest. 2. Confirm steady merge of type => Address::BILLING_TYPE.
Test Data	- Field merged: 'type' => Address::BILLING_TYPE
Expected Result	getBillingAddress merges 'type' with BILLING_TYPE appropriately.
Actual Result	getBillingAddress merges 'type' with BILLING_TYPE appropriately.
Status	Pass
Severity	High

File: RecurringInvoice-Test.txt

Test Case ID	RI-001
Title	Instantiate RecurringInvoice model
Objective	Verify that the RecurringInvoice model can be successfully instantiated.
Preconditions	- Application running - Database seeded (if required by autoloaders) - Required model class files available
Test Steps	1. Attempt to instantiate a new RecurringInvoice object. 2. Check the type of the instantiated object.
Test Data	- Class: RecurringInvoice
Expected Result	- The instantiated object is of type RecurringInvoice.
Actual Result	A new RecurringInvoice instance is created and confirmed to be of type RecurringInvoice.
Status	Pass
Severity	Medium

Test Case ID	RI-002
Title	RecurringInvoice extends Model and uses specific traits
Objective	Confirm RecurringInvoice extends Eloquent Model and uses HasFactory and HasCustomFieldsTrait.
Preconditions	- Application running - Database seeded - Required class and trait files available
Test Steps	1. Instantiate a new RecurringInvoice object. 2. Use ReflectionClass on RecurringInvoice. 3. Retrieve all trait names. 4. Check object type inheritance. 5. Validate that the required traits are present.
Test Data	- Traits: HasFactory, HasCustomFieldsTrait - Class: Illuminate\Database\Eloquent\Model
Expected Result	- RecurringInvoice is an instance of Model. - Traits HasFactory and HasCustomFieldsTrait are used by RecurringInvoice.
Actual Result	RecurringInvoice correctly extends Model and uses both required traits.
Status	Pass
Severity	Medium

Test Case ID	RI-003
Title	RecurringInvoice has limit constants
Objective	Ensure RecurringInvoice defines limit constants NONE, COUNT, and DATE with correct values.
Preconditions	- Application running - Source code accessible
Test Steps	1. Access RecurringInvoice::NONE, ::COUNT, ::DATE. 2. Assert each constant is defined and equals its expected string.
Test Data	- Expected constants: - NONE => 'NONE' - COUNT => 'COUNT' - DATE => 'DATE'

Expected Result	- RecurringInvoice::NONE is 'NONE'. - RecurringInvoice::COUNT is 'COUNT'. - RecurringInvoice::DATE is 'DATE'.
Actual Result	All constants defined with correct values.
Status	Pass
Severity	Low

Test Case ID	RI-004
Title	RecurringInvoice has status constants
Objective	Ensure RecurringInvoice defines status constants COMPLETED, ON_HOLD, and ACTIVE.
Preconditions	- Application running - Source code accessible
Test Steps	1. Access RecurringInvoice::COMPLETED, ::ON_HOLD, ::ACTIVE. 2. Assert each constant is defined and equals its expected string.
Test Data	- Expected constants: - COMPLETED => 'COMPLETED' - ON_HOLD => 'ON_HOLD' - ACTIVE => 'ACTIVE'
Expected Result	- RecurringInvoice::COMPLETED is 'COMPLETED'. - RecurringInvoice::ON_HOLD is 'ON_HOLD'. - RecurringInvoice::ACTIVE is 'ACTIVE'.
Actual Result	All constants defined with correct values.
Status	Pass
Severity	Low

Test Case ID	RI-005
Title	RecurringInvoice has correct attribute casts
Objective	Verify required attributes are correctly cast to their types in RecurringInvoice.
Preconditions	- Application running - Source code accessible
Test Steps	1. Instantiate RecurringInvoice. 2. Retrieve its attribute casts. 3. Check that 'exchange_rate' is cast as 'float' and 'send_automatically' as 'boolean'.
Test Data	- Cast fields: - exchange_rate: float - send_automatically: boolean
Expected Result	- Casts array has 'exchange_rate' => 'float'. - Casts array has 'send_automatically' => 'boolean'.
Actual Result	Both attributes found with correct casts.
Status	Pass
Severity	Medium

Test Case ID	RI-006
Title	RecurringInvoice has formatted date accessors
Objective	Confirm presence of accessor methods for formatted dates in RecurringInvoice.
Preconditions	- Application running - Source code accessible

Test Steps	1. Use ReflectionClass on RecurringInvoice. 2. Confirm existence of methods: - getFormattedStartsAtAttribute - getFormattedNextInvoiceAtAttribute - getFormattedLimitDateAttribute - getFormattedCreatedAtAttribute
Test Data	- Expected accessor method names
Expected Result	- All four accessor methods are present in the class.
Actual Result	All accessor methods for date formatting exist.
Status	Pass
Severity	Low

Test Case ID	RI-007
Title	RecurringInvoice has relationship methods
Objective	Ensure RecurringInvoice has all the required Eloquent relationship methods.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on RecurringInvoice. 2. Check for methods: - invoices - taxes - items - customer - company - creator - currency
Test Data	- List of relationship method names
Expected Result	- All relationship methods are present.
Actual Result	All listed relationship methods found in RecurringInvoice.
Status	Pass
Severity	Medium

Test Case ID	RI-008
Title	RecurringInvoice has scope methods for querying
Objective	Confirm existence of scope methods enabling filtered queries on RecurringInvoice.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on RecurringInvoice. 2. Check for scope methods: - scopeWhereCompany - scopePaginateData - scopeWhereOrder - scopeWhereStatus - scopeWhereCustomer - scopeWhereSearch - scopeApplyFilters
Test Data	- List of scope method names
Expected Result	- Each scope method is implemented in the class.
Actual Result	Scope methods exist for all specified filtering operations.
Status	Pass
Severity	Medium

Test Case ID	RI-009
Title	RecurringInvoice has static CRUD methods
Objective	Ensure presence of main CRUD static methods in RecurringInvoice.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on RecurringInvoice. 2. Check for methods: - createFromRequest - createItems - createTaxes - deleteRecurringInvoice
Test Data	- List of CRUD method names
Expected Result	- All CRUD methods are present in RecurringInvoice.
Actual Result	All static methods for CRUD operations exist.
Status	Pass
Severity	High

Test Case ID	RI-010
Title	RecurringInvoice has invoice generation and control methods
Objective	Validate existence of methods responsible for generating and managing invoices.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on RecurringInvoice. 2. Check for methods: - generateInvoice - createInvoice - markStatusAsCompleted - updateNextInvoiceDate
Test Data	- Method names related to invoice lifecycle
Expected Result	- All invoice operation methods are present in RecurringInvoice.
Actual Result	Invoice generation and status methods are implemented correctly.
Status	Pass
Severity	High

Test Case ID	RI-011
Title	RecurringInvoice has static getNextInvoiceDate method
Objective	Verify that RecurringInvoice has a static, public getNextInvoiceDate method.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on RecurringInvoice. 2. Check for 'getNextInvoiceDate' method. 3. Verify it is static and public.
Test Data	- Method name: getNextInvoiceDate
Expected Result	- Method exists, is static, and public.
Actual Result	getNextInvoiceDate is public and static.
Status	Pass
Severity	High

Test Case ID	RI-012

Title	RecurringInvoice scopePaginateData handles 'all' limit
Objective	Ensure scopePaginateData method supports both paginated and 'all' queries.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on RecurringInvoice. 2. Read source file content. 3. Search for code handling '$limit == "all"', returning all records. 4. Check for code returning paginated records.
Test Data	- Source code snippet
Expected Result	- The function handles both 'all' limit and paginated data requests.
Actual Result	scopePaginateData supports 'all' and paginated scenarios.
Status	Pass
Severity	High

Test Case ID	RI-013
Title	RecurringInvoice invoices relationship returns HasMany
Objective	Verify that RecurringInvoice::invoices() returns a HasMany relation and the related model is Invoice.
Preconditions	- Application running - Related models available - Database seeded (optional)
Test Steps	1. Instantiate RecurringInvoice. 2. Call invoices() relationship method. 3. Check that the returned type is HasMany. 4. Check the relation's related model is Invoice.
Test Data	- Model: Invoice
Expected Result	- invoices() returns HasMany. - Related model is Invoice.
Actual Result	HasMany relation to Invoice confirmed.
Status	Pass
Severity	High

Test Case ID	RI-014
Title	RecurringInvoice customer relationship returns BelongsTo
Objective	Verify that RecurringInvoice::customer() returns a BelongsTo relation with Customer model.
Preconditions	- Application running - Related models available - Database seeded (optional)
Test Steps	1. Instantiate RecurringInvoice. 2. Call customer() relationship. 3. Check that returned type is BelongsTo. 4. Confirm related model is Customer.
Test Data	- Model: Customer
Expected Result	- customer() returns BelongsTo. - Related model is Customer.
Actual Result	BelongsTo relation to Customer confirmed.
Status	Pass
Severity	Medium

Test Case ID	RI-015
Title	RecurringInvoice company relationship returns BelongsTo
Objective	Verify that RecurringInvoice::company() returns a BelongsTo relation to Company.
Preconditions	- Application running - Related models available
Test Steps	1. Instantiate RecurringInvoice. 2. Call company() relationship. 3. Confirm BelongsTo relation. 4. Confirm related model is Company.
Test Data	- Model: Company
Expected Result	- company() returns BelongsTo. - Related model is Company.
Actual Result	BelongsTo relation to Company confirmed.
Status	Pass
Severity	Medium

Test Case ID	RI-016
Title	RecurringInvoice currency relationship returns BelongsTo
Objective	Verify that RecurringInvoice::currency() returns a BelongsTo relation to Currency.
Preconditions	- Application running - Related models available
Test Steps	1. Instantiate RecurringInvoice. 2. Call currency() relationship. 3. Confirm BelongsTo relation. 4. Confirm related model is Currency.
Test Data	- Model: Currency
Expected Result	- currency() returns BelongsTo. - Related model is Currency.
Actual Result	BelongsTo relation to Currency confirmed.
Status	Pass
Severity	Medium

Test Case ID	RI-017
Title	Instantiate RecurringInvoiceCollection
Objective	Ensure RecurringInvoiceCollection can be instantiated.
Preconditions	- Application running - Collection module available
Test Steps	1. Create a new Collection (empty). 2. Instantiate RecurringInvoiceCollection with the collection. 3. Check type of the instantiated object.
Test Data	- Collection: []
Expected Result	- Instance is created and is of RecurringInvoiceCollection class.
Actual Result	RecurringInvoiceCollection instantiated successfully.
Status	Pass
Severity	Medium

Test Case ID	RI-018

Title	RecurringInvoiceCollection is a ResourceCollection
Objective	Confirm RecurringInvoiceCollection extends the ResourceCollection class.
Preconditions	- Application running - ResourceCollection available
Test Steps	1. Create new Collection. 2. Instantiate RecurringInvoiceCollection with the collection. 3. Assert instance type is ResourceCollection.
Test Data	- Collection: []
Expected Result	- Instance is a ResourceCollection.
Actual Result	Type inheritance confirmed.
Status	Pass
Severity	Low

Test Case ID	RI-019
Title	RecurringInvoiceCollection delegates to parent toArray method
Objective	Ensure RecurringInvoiceCollection uses the parent toArray method for conversions.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on RecurringInvoiceCollection. 2. Read source file content. 3. Search for usage of parent::toArray.
Test Data	- Source code snippet
Expected Result	- parent::toArray is present in RecurringInvoiceCollection class.
Actual Result	Delegation to parent toArray confirmed.
Status	Pass
Severity	Low

Test Case ID	RI-020
Title	RecurringInvoiceCollection returns empty array for empty collection
Objective	Verify that toArray returns empty array if collection is empty.
Preconditions	- Application running - Collection module available
Test Steps	1. Instantiate Request. 2. Create RecurringInvoiceCollection with empty Collection. 3. Call toArray with the request. 4. Check returned value type and count.
Test Data	- Collection: [] - Request: empty
Expected Result	- Returned result is an empty array.
Actual Result	Empty array returned as expected.
Status	Pass
Severity	Medium

Test Case ID	RI-021
Title	RecurringInvoiceCollection toArray method is public
Objective	Confirm toArray method in RecurringInvoiceCollection is public and non-static.
Preconditions	- Application running - Source code accessible

Test Steps	1. Use ReflectionClass on RecurringInvoiceCollection. 2. Get toArray method and check visibility and static status.
Test Data	- Method: toArray
Expected Result	- toArray is public. - toArray is not static.
Actual Result	toArray is public and non-static.
Status	Pass
Severity	Low

Test Case ID	RI-022
Title	Instantiate RecurringInvoiceFrequencyController
Objective	Verify controller can be instantiated.
Preconditions	- Application running - Controller classes available
Test Steps	1. Attempt to instantiate RecurringInvoiceFrequencyController. 2. Assert object is instance of RecurringInvoiceFrequencyController.
Test Data	- Class: RecurringInvoiceFrequencyController
Expected Result	- Instance created successfully.
Actual Result	Controller instantiated as expected.
Status	Pass
Severity	Medium

Test Case ID	RI-023
Title	RecurringInvoiceFrequencyController extends base Controller
Objective	Confirm that RecurringInvoiceFrequencyController extends Controller.
Preconditions	- Application running - Controller base class available
Test Steps	1. Instantiate RecurringInvoiceFrequencyController. 2. Assert instance type Controller.
Test Data	- Base class: Controller
Expected Result	- Instance is a Controller.
Actual Result	Extension of base Controller confirmed.
Status	Pass
Severity	Medium

Test Case ID	RI-024
Title	RecurringInvoiceFrequencyController is invokable
Objective	Ensure controller implements __invoke method.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on RecurringInvoiceFrequencyController. 2. Check for existence of __invoke method.
Test Data	- Method: __invoke
Expected Result	- Method __invoke is present.
Actual Result	Controller confirmed invokable.
Status	Pass
Severity	Medium

Test Case ID	RI-025
Title	RecurringInvoiceFrequencyController uses RecurringInvoice::getNextInvoiceDate
Objective	Confirm controller calls getNextInvoiceDate during execution.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on controller. 2. Read source file content. 3. Confirm call to RecurringInvoice::getNextInvoiceDate with request frequency/starts_at.
Test Data	- Method called: getNextInvoiceDate
Expected Result	- Controller source contains relevant function call.
Actual Result	getNextInvoiceDate used in controller as required.
Status	Pass
Severity	High

Test Case ID	RI-026
Title	RecurringInvoiceFrequencyController returns JSON with next_invoice_at
Objective	Validate that controller returns a JSON response with success and next_invoice_at fields.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use ReflectionClass on controller. 2. Read source code. 3. Check that response()->json is used. 4. Ensure 'success' and 'next_invoice_at' are included in the output array.
Test Data	- Response keys: 'success', 'next_invoice_at'
Expected Result	- Controller returns JSON with 'success' as true and 'next_invoice_at' value.
Actual Result	JSON response confirmed as specified.
Status	Pass
Severity	High

File: Redirect-Test.txt

Test Case ID	R-001
Title	RedirectIfAuthenticated Middleware Instantiation
Objective	Verify that the RedirectIfAuthenticated middleware can be instantiated.
Preconditions	- Application running - Autoloader configured - Required classes available
Test Steps	1. Instantiate an object of RedirectIfAuthenticated middleware. 2. Check if the object is an instance of RedirectIfAuthenticated class.
Test Data	- Middleware class: RedirectIfAuthenticated
Expected Result	- The instantiated object is an instance of RedirectIfAuthenticated class.
Actual Result	The object is successfully instantiated and is an instance of RedirectIfAuthenticated.
Status	Pass
Severity	High

Test Case ID	R-002
Title	RedirectIfAuthenticated Namespace Validation
Objective	Verify that the RedirectIfAuthenticated middleware resides in the correct namespace.
Preconditions	- Application running - RedirectIfAuthenticated class available
Test Steps	1. Use ReflectionClass to inspect the RedirectIfAuthenticated class. 2. Retrieve the namespace name of the class. 3. Compare with the expected namespace.
Test Data	- Expected namespace: Crater\Http\Middleware
Expected Result	- Namespace of RedirectIfAuthenticated class is 'Crater\Http\Middleware'.
Actual Result	Namespace is correctly set to 'Crater\Http\Middleware'.
Status	Pass
Severity	Medium

Test Case ID	R-003
Title	RedirectIfAuthenticated Handle Method Existence and Accessibility
Objective	Verify that the RedirectIfAuthenticated middleware includes a public handle method.
Preconditions	- Application running - RedirectIfAuthenticated class available
Test Steps	1. Use ReflectionClass to inspect the RedirectIfAuthenticated class. 2. Check if the class has a method named 'handle'. 3. Confirm the 'handle' method is public.
Test Data	- Method name: handle
Expected Result	- The class has a public method named 'handle'.
Actual Result	Public method 'handle' exists in RedirectIfAuthenticated.
Status	Pass
Severity	High

Test Case ID	R-004

Title	RedirectIfAuthenticated Handle Method Parameters Verification
Objective	Ensure the handle method of RedirectIfAuthenticated accepts request, closure, and guard parameters.
Preconditions	- Application running - RedirectIfAuthenticated class available
Test Steps	1. Use ReflectionClass to get handle method from RedirectIfAuthenticated. 2. Enumerate the method's parameters. 3. Confirm there are three parameters: request, next, guard.
Test Data	- Parameter names: request, next, guard
Expected Result	- handle method has exactly three parameters named request, next, guard.
Actual Result	handle method correctly declares parameters: request, next, guard.
Status	Pass
Severity	High

Test Case ID	R-005
Title	RedirectIfAuthenticated Logic Usage Verification
Objective	Validate usage of Auth guard, home route redirect, and request forwarding logic in RedirectIfAuthenticated middleware.
Preconditions	- Application running - RedirectIfAuthenticated class available
Test Steps	1. Use ReflectionClass to read the class file contents. 2. Search for the following: a. 'Auth::guard($guard)->check()' b. 'redirect(RouteServiceProvider::HOME)' c. 'return $next($request)' 3. Verify presence of all logic patterns.
Test Data	- File contents of RedirectIfAuthenticated
Expected Result	- File contains 'Auth::guard($guard)->check()'. - File contains 'redirect(RouteServiceProvider::HOME)'. - File contains 'return $next($request)'.
Actual Result	All logic blocks correctly found within middleware source.
Status	Pass
Severity	High

Test Case ID	R-006
Title	RedirectIfInstalled Middleware Instantiation
Objective	Verify that the RedirectIfInstalled middleware can be instantiated.
Preconditions	- Application running - Autoloader configured - Required classes available
Test Steps	1. Instantiate an object of RedirectIfInstalled middleware. 2. Check if the object is an instance of RedirectIfInstalled class.
Test Data	- Middleware class: RedirectIfInstalled
Expected Result	- The instantiated object is an instance of RedirectIfInstalled class.
Actual Result	The object is successfully instantiated and is an instance of RedirectIfInstalled.
Status	Pass
Severity	High

Test Case ID	R-007
Title	RedirectIfInstalled Namespace Validation

Objective	Verify that the RedirectIfInstalled middleware resides in the correct namespace.
Preconditions	- Application running - RedirectIfInstalled class available
Test Steps	1. Use ReflectionClass to inspect the RedirectIfInstalled class. 2. Retrieve the namespace name of the class. 3. Compare with the expected namespace.
Test Data	- Expected namespace: Crater\Http\Middleware
Expected Result	- Namespace of RedirectIfInstalled class is 'Crater\Http\Middleware'.
Actual Result	Namespace is correctly set to 'Crater\Http\Middleware'.
Status	Pass
Severity	Medium

Test Case ID	R-008
Title	RedirectIfInstalled Handle Method Existence and Accessibility
Objective	Verify that the RedirectIfInstalled middleware includes a public handle method.
Preconditions	- Application running - RedirectIfInstalled class available
Test Steps	1. Use ReflectionClass to inspect the RedirectIfInstalled class. 2. Check if the class has a method named 'handle'. 3. Confirm the 'handle' method is public.
Test Data	- Method name: handle
Expected Result	- The class has a public method named 'handle'.
Actual Result	Public method 'handle' exists in RedirectIfInstalled.
Status	Pass
Severity	High

Test Case ID	R-009
Title	RedirectIfInstalled Handle Method Parameters Verification
Objective	Ensure the handle method of RedirectIfInstalled accepts request and closure parameters.
Preconditions	- Application running - RedirectIfInstalled class available
Test Steps	1. Use ReflectionClass to get handle method from RedirectIfInstalled. 2. Enumerate the method’s parameters. 3. Confirm there are two parameters: request, next.
Test Data	- Parameter names: request, next
Expected Result	- handle method has exactly two parameters named request and next.
Actual Result	handle method correctly declares parameters: request, next.
Status	Pass
Severity	High

Test Case ID	R-010
Title	RedirectIfInstalled Logic Usage Verification
Objective	Validate usage of storage, setting check, completion comparison, login redirect, and request forwarding logic in RedirectIfInstalled middleware.
Preconditions	- Application running - RedirectIfInstalled class available

Test Steps	1. Use ReflectionClass to read the class file contents. 2. Search for the following: a. '\\Storage::disk(\'local\')->has(\'database_created\')' b. 'Setting::getSetting(\'profile_complete\')' c. '=== \'COMPLETED\'' d. 'redirect(\'login\')' e. 'return $next($request)' 3. Verify presence of all logic patterns.
Test Data	- File contents of RedirectIfInstalled
Expected Result	- File contains '\\Storage::disk(\'local\')->has(\'database_created\')'. - File contains 'Setting::getSetting(\'profile_complete\')'. - File contains '=== \'COMPLETED\''. - File contains 'redirect(\'login\')'. - File contains 'return $next($request)'.
Actual Result	All logic blocks correctly found within middleware source.
Status	Pass
Severity	High

Test Case ID	R-011
Title	RedirectIfUnauthorized Middleware Instantiation
Objective	Verify that the RedirectIfUnauthorized middleware can be instantiated.
Preconditions	- Application running - Autoloader configured - Required classes available
Test Steps	1. Instantiate an object of RedirectIfUnauthorized middleware. 2. Check if the object is an instance of RedirectIfUnauthorized class.
Test Data	- Middleware class: RedirectIfUnauthorized
Expected Result	- The instantiated object is an instance of RedirectIfUnauthorized class.
Actual Result	The object is successfully instantiated and is an instance of RedirectIfUnauthorized.
Status	Pass
Severity	High

Test Case ID	R-012
Title	RedirectIfUnauthorized Namespace Validation
Objective	Verify that the RedirectIfUnauthorized middleware resides in the correct namespace.
Preconditions	- Application running - RedirectIfUnauthorized class available
Test Steps	1. Use ReflectionClass to inspect the RedirectIfUnauthorized class. 2. Retrieve the namespace name of the class. 3. Compare with the expected namespace.
Test Data	- Expected namespace: Crater\Http\Middleware
Expected Result	- Namespace of RedirectIfUnauthorized class is 'Crater\Http\Middleware'.
Actual Result	Namespace is correctly set to 'Crater\Http\Middleware'.
Status	Pass
Severity	Medium

Test Case ID	R-013
Title	RedirectIfUnauthorized Handle Method Existence and Accessibility

Objective	Verify that the RedirectIfUnauthorized middleware includes a public handle method.
Preconditions	- Application running - RedirectIfUnauthorized class available
Test Steps	1. Use ReflectionClass to inspect the RedirectIfUnauthorized class. 2. Check if the class has a method named 'handle'. 3. Confirm the 'handle' method is public.
Test Data	- Method name: handle
Expected Result	- The class has a public method named 'handle'.
Actual Result	Public method 'handle' exists in RedirectIfUnauthorized.
Status	Pass
Severity	High

Test Case ID	R-014
Title	RedirectIfUnauthorized Handle Method Parameters Verification
Objective	Ensure the handle method of RedirectIfUnauthorized accepts request, closure, and guard parameters.
Preconditions	- Application running - RedirectIfUnauthorized class available
Test Steps	1. Use ReflectionClass to get handle method from RedirectIfUnauthorized. 2. Enumerate the method’s parameters. 3. Confirm there are three parameters: request, next, guard.
Test Data	- Parameter names: request, next, guard
Expected Result	- handle method has exactly three parameters named request, next, guard.
Actual Result	handle method correctly declares parameters: request, next, guard.
Status	Pass
Severity	High

Test Case ID	R-015
Title	RedirectIfUnauthorized Logic Usage Verification
Objective	Validate usage of Auth guard check, login redirection, and request forwarding logic in RedirectIfUnauthorized middleware.
Preconditions	- Application running - RedirectIfUnauthorized class available
Test Steps	1. Use ReflectionClass to read the class file contents. 2. Search for the following: a. 'Auth::guard($guard)->check()' b. 'return $next($request)' c. 'redirect(\'/login\')' 3. Verify presence of all logic patterns.
Test Data	- File contents of RedirectIfUnauthorized
Expected Result	- File contains 'Auth::guard($guard)->check()'. - File contains 'return $next($request)'. - File contains 'redirect(\'/login\')'.
Actual Result	All logic blocks correctly found within middleware source.
Status	Pass
Severity	High

File: RegisterController-Test.txt

Test Case ID	RC-001
Title	Instantiation of RegisterController
Objective	Verify that the RegisterController can be instantiated successfully.
Preconditions	- Application running - All required dependencies are autoloaded - PHP environment set up
Test Steps	1. Attempt to instantiate the RegisterController class. 2. Check the result of the instantiation.
Test Data	- None (class instantiation)
Expected Result	- The instantiated object is an instance of RegisterController.
Actual Result	RegisterController successfully instantiated as expected.
Status	Pass
Severity	High

Test Case ID	RC-002
Title	RegisterController extends Base Controller
Objective	Ensure RegisterController is a subclass of the base Controller class.
Preconditions	- Application running - All required dependencies are autoloaded
Test Steps	1. Instantiate RegisterController. 2. Verify the object is an instance of the base Controller class.
Test Data	- None (class inheritance)
Expected Result	- The instantiated object is an instance of Crater\Http\Controllers\Controller.
Actual Result	RegisterController recognized as a subclass of Controller.
Status	Pass
Severity	High

Test Case ID	RC-003
Title	RegisterController Namespace Verification
Objective	Verify that RegisterController resides in the correct namespace.
Preconditions	- Application running - All source files accessible
Test Steps	1. Use PHP reflection to fetch RegisterController class. 2. Retrieve the namespace via getNamespaceName(). 3. Compare namespace with expected value.
Test Data	- Expected namespace: 'Crater\Http\Controllers\V1\Admin\Auth'
Expected Result	- Namespace of RegisterController is 'Crater\Http\Controllers\V1\Admin\Auth'.
Actual Result	Namespace matches expected value.
Status	Pass
Severity	Medium

Test Case ID	RC-004
Title	Usage of RegistersUsers Trait in RegisterController
Objective	Ensure that RegisterController uses the 'RegistersUsers' trait.

Preconditions	- Application running - Source code containing trait must be available
Test Steps	1. Use PHP reflection to list all traits used by RegisterController. 2. Verify inclusion of 'Illuminate\Foundation\Auth\RegistersUsers'.
Test Data	- Trait name: 'Illuminate\Foundation\Auth\RegistersUsers'
Expected Result	- RegisterController uses 'Illuminate\Foundation\Auth\RegistersUsers'.
Actual Result	Trait found in RegisterController.
Status	Pass
Severity	High

Test Case ID	RC-005
Title	Existence and Access of redirectTo Property
Objective	Confirm that RegisterController has a protected property named 'redirectTo'.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use reflection to check for 'redirectTo' property in RegisterController. 2. Verify that the property is protected.
Test Data	- Property name: 'redirectTo'
Expected Result	- RegisterController has a protected property named 'redirectTo'.
Actual Result	Protected property 'redirectTo' confirmed.
Status	Pass
Severity	Medium

Test Case ID	RC-006
Title	Value of redirectTo Property
Objective	Validate that the 'redirectTo' property is set to RouteServiceProvider::HOME.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use reflection to get the file content of RegisterController. 2. Search for 'protected $redirectTo = RouteServiceProvider::HOME' in class definition.
Test Data	- Property initialization line
Expected Result	- RegisterController's 'redirectTo' property is initialized to RouteServiceProvider::HOME.
Actual Result	redirectTo property set to RouteServiceProvider::HOME.
Status	Pass
Severity	High

Test Case ID	RC-007
Title	Constructor Applies Guest Middleware
Objective	Ensure the RegisterController constructor applies the 'guest' middleware.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use reflection to verify existence of '__construct' method. 2. Retrieve the file contents. 3. Search for $this->middleware('guest') within the constructor.
Test Data	- Middleware name: 'guest'
Expected Result	- RegisterController has a constructor, and it applies the 'guest' middleware.

Actual Result	Constructor present and 'guest' middleware applied.
Status	Pass
Severity	High

Test Case ID	RC-008
Title	Existence and Access of validator Method
Objective	Confirm RegisterController has a protected 'validator' method.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use reflection to check for method named 'validator'. 2. Check that the method's visibility is protected.
Test Data	- Method name: 'validator'
Expected Result	- RegisterController has a protected method named 'validator'.
Actual Result	Protected 'validator' method identified.
Status	Pass
Severity	High

Test Case ID	RC-009
Title	Validator Method Rules Coverage
Objective	Ensure the 'validator' method implements correct validation rules for registration.
Preconditions	- Application running - Source code accessible
Test Steps	1. Use reflection to get file content of RegisterController. 2. Scan for the following validation rules within the 'validator' method: - 'name' => ['required', 'string', 'max:255'] - 'email' => ['required', 'string', 'email', 'max:255', 'unique:users'] - 'password' => ['required', 'string', 'min:8', 'confirmed']
Test Data	- Validation rule set as listed above
Expected Result	- 'validator' contains all three rules as specified: - Name: required, string, max:255 - Email: required, string, email, max:255, unique - Password: required, string, min:8, confirmed
Actual Result	All validation rules correctly implemented.
Status	Pass
Severity	High

Test Case ID	RC-010
Title	Existence and Function of 'create' Method
Objective	Verify that RegisterController has a 'create' method which creates User with required fields.
Preconditions	- Application running - Source code accessible - User model available
Test Steps	1. Use reflection to check for method named 'create' in RegisterController. 2. Get file content and confirm call to User::create() in method body. 3. Confirm that 'name', 'email', and 'password' fields are passed to User::create().
Test Data	- Fields: 'name', 'email', 'password' - Method: User::create()

Expected Result	- 'create' method exists and calls User::create() with 'name', 'email', and 'password'.
Actual Result	'create' method and User creation with required fields verified.
Status	Pass
Severity	High

File: RelationNotExist-Test.txt

Test Case ID	RNE-001
Title	Instantiate RelationNotExist class
Objective	Verify that the RelationNotExist class can be instantiated.
Preconditions	- Application running - All necessary dependencies loaded
Test Steps	1. Instantiate the RelationNotExist class with no arguments. 2. Check if the created object is an instance of RelationNotExist.
Test Data	- Class: RelationNotExist - Arguments: none
Expected Result	- The created object is an instance of RelationNotExist.
Actual Result	The created object was an instance of RelationNotExist.
Status	Pass
Severity	Medium

Test Case ID	RNE-002
Title	RelationNotExist implements Rule interface
Objective	Confirm that RelationNotExist implements the Rule interface from Illuminate\Contracts\Validation.
Preconditions	- Application running
Test Steps	1. Instantiate the RelationNotExist class. 2. Assert that the object is an instance of Illuminate\Contracts\Validation\Rule.
Test Data	- Class: RelationNotExist - Interface: Illuminate\Contracts\Validation\Rule
Expected Result	- The instantiated object is an instance of Illuminate\Contracts\Validation\Rule.
Actual Result	The object implements the Rule interface as expected.
Status	Pass
Severity	Medium

Test Case ID	RNE-003
Title	Check RelationNotExist namespace
Objective	Ensure that RelationNotExist is in the 'Crater\Rules' namespace.
Preconditions	- Application running - RelationNotExist class accessible in autoload
Test Steps	1. Use ReflectionClass to get the namespace of RelationNotExist. 2. Verify that the namespace matches 'Crater\Rules'.
Test Data	- Class: RelationNotExist
Expected Result	- Namespace of RelationNotExist is 'Crater\Rules'.
Actual Result	Namespace was verified as 'Crater\Rules'.
Status	Pass
Severity	Low

Test Case ID	RNE-004
Title	RelationNotExist has public class and relation properties
Objective	Validate that the 'class' and 'relation' properties are public in RelationNotExist.
Preconditions	- Application running

Test Steps	1. Use ReflectionClass to check for 'class' and 'relation' properties. 2. Check that both properties exist. 3. Verify that both properties are public.
Test Data	- Properties: 'class', 'relation'
Expected Result	- Both 'class' and 'relation' properties exist and are public.
Actual Result	Both properties were present and public.
Status	Pass
Severity	Low

Test Case ID	RNE-005
Title	RelationNotExist has required methods
Objective	Ensure that RelationNotExist contains __construct, passes, and message methods.
Preconditions	- Application running
Test Steps	1. Use ReflectionClass to check for method existence. 2. Verify the presence of __construct method. 3. Verify the presence of passes method. 4. Verify the presence of message method.
Test Data	- Methods: __construct, passes, message
Expected Result	- All three methods exist in the RelationNotExist class.
Actual Result	All required methods were found.
Status	Pass
Severity	Medium

Test Case ID	RNE-006
Title	RelationNotExist constructor sets class and relation properties
Objective	Confirm that constructor arguments are assigned to the 'class' and 'relation' properties.
Preconditions	- Application running
Test Steps	1. Instantiate RelationNotExist with 'SomeClass' and 'someRelation'. 2. Check that 'class' property is 'SomeClass'. 3. Check that 'relation' property is 'someRelation'.
Test Data	- class: 'SomeClass' - relation: 'someRelation'
Expected Result	- 'class' property equals 'SomeClass' - 'relation' property equals 'someRelation'
Actual Result	'class' and 'relation' properties matched constructor arguments.
Status	Pass
Severity	Medium

Test Case ID	RNE-007
Title	RelationNotExist constructor accepts null values
Objective	Verify that passing null arguments to the constructor initializes properties to null.
Preconditions	- Application running
Test Steps	1. Instantiate RelationNotExist with null for both class and relation. 2. Assert that both 'class' and 'relation' properties are null.
Test Data	- class: null - relation: null

Expected Result	- 'class' property is null. - 'relation' property is null.
Actual Result	Both properties were null as expected.
Status	Pass
Severity	Low

Test Case ID	RNE-008
Title	RelationNotExist constructor works with no arguments
Objective	Ensure that if no arguments are provided, properties default to null.
Preconditions	- Application running
Test Steps	1. Instantiate RelationNotExist with no arguments. 2. Assert that both 'class' and 'relation' properties are null.
Test Data	- No constructor arguments
Expected Result	- 'class' property is null. - 'relation' property is null.
Actual Result	Both properties were null.
Status	Pass
Severity	Low

Test Case ID	RNE-009
Title	RelationNotExist message returns correct format
Objective	Check that the message method returns the correct string format when relation is provided.
Preconditions	- Application running
Test Steps	1. Instantiate RelationNotExist with 'TestClass' and 'testRelation'. 2. Call the message() method. 3. Assert that the returned message matches 'Relation testRelation exists.'.
Test Data	- class: 'TestClass' - relation: 'testRelation'
Expected Result	- message() returns 'Relation testRelation exists.'
Actual Result	Returned message was 'Relation testRelation exists.'.
Status	Pass
Severity	Medium

Test Case ID	RNE-010
Title	RelationNotExist message handles null relation
Objective	Ensure the message method works correctly when relation is null.
Preconditions	- Application running
Test Steps	1. Instantiate RelationNotExist with 'TestClass' and null for relation. 2. Call the message() method. 3. Assert that the returned message matches 'Relation exists.'.
Test Data	- class: 'TestClass' - relation: null
Expected Result	- message() returns 'Relation exists.'
Actual Result	Returned message was 'Relation exists.'.
Status	Pass
Severity	Medium

Test Case ID	RNE-011
Title	passes() returns true when relation does not exist
Objective	Verify that the passes method returns true if the related model does not exist.
Preconditions	- Application running - Mockery library loaded - Mock model and relation set up to return 'exists' as false
Test Steps	1. Mock a relation to return exists() as false. 2. Mock a model where 'testRelation' returns the mocked relation. 3. Mock a class 'TestModelClass' where find(123) returns the mocked model. 4. Instantiate RelationNotExist with 'TestModelClass' and 'testRelation'. 5. Call passes('attribute', 123) and assert result is true.
Test Data	- class: 'TestModelClass' - relation: 'testRelation' - attribute: 'attribute' - value: 123 - exists() returns: false
Expected Result	- passes() returns true.
Actual Result	passes() returned true.
Status	Pass
Severity	High

Test Case ID	RNE-012
Title	passes() returns false when relation exists
Objective	Ensure that the passes method returns false if the relation exists.
Preconditions	- Application running - Mockery library loaded - Mock model and relation set up to return 'exists' as true
Test Steps	1. Mock a relation to return exists() as true. 2. Mock a model where 'testRelation' returns the mocked relation. 3. Mock a class 'AnotherModelClass' where find(456) returns the mocked model. 4. Instantiate RelationNotExist with 'AnotherModelClass' and 'testRelation'. 5. Call passes('attribute', 456) and assert result is false.
Test Data	- class: 'AnotherModelClass' - relation: 'testRelation' - attribute: 'attribute' - value: 456 - exists() returns: true
Expected Result	- passes() returns false.
Actual Result	passes() returned false.
Status	Pass
Severity	High

File: Request-Test.txt

Test Case ID	R-001
Title	Verify Request Class is Abstract
Objective	To verify that the Request class is defined as an abstract class.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\Request class available - PHP Reflection API available
Test Steps	1. Use PHP ReflectionClass to load the Crater\Http\Requests\Request class. 2. Check whether the loaded class is declared as abstract.
Test Data	- Class: Crater\Http\Requests\Request
Expected Result	- The ReflectionClass isAbstract() method returns true, confirming the Request class is abstract.
Actual Result	The ReflectionClass isAbstract() method returned true; Request class is abstract.
Status	Pass
Severity	Medium

Test Case ID	R-002
Title	Verify Request Class Inheritance from FormRequest
Objective	To confirm that the Request class extends Illuminate\Foundation\Http\FormRequest.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\Request class available - Illuminate\Foundation\Http\FormRequest class available
Test Steps	1. Use PHP ReflectionClass to load the Crater\Http\Requests\Request class. 2. Retrieve the parent class of the Request class using getParentClass(). 3. Check if the parent class name matches Illuminate\Foundation\Http\FormRequest.
Test Data	- Class: Crater\Http\Requests\Request - Parent class: Illuminate\Foundation\Http\FormRequest
Expected Result	- The parent class name returned by ReflectionClass matches Illuminate\Foundation\Http\FormRequest.
Actual Result	ReflectionClass returned Illuminate\Foundation\Http\FormRequest as the parent class for Request.
Status	Pass
Severity	Medium

File: RequirementsController-Test.txt

Test Case ID	RC-001
Title	Constructor injects RequirementsChecker instance
Objective	Verify that the RequirementsController constructor correctly injects and sets the RequirementsChecker dependency.
Preconditions	- Application running - Mocked RequirementsChecker instance available - Controller instantiated with mocked dependency
Test Steps	1. Instantiate the RequirementsController with a mocked RequirementsChecker object. 2. Use reflection to access the protected 'requirements' property of the controller. 3. Assert that the 'requirements' property is set to the mocked RequirementsChecker.
Test Data	- Mocked RequirementsChecker instance
Expected Result	- The 'requirements' property of RequirementsController is set to the mocked RequirementsChecker instance.
Actual Result	The 'requirements' property of RequirementsController was correctly set to the mocked RequirementsChecker instance.
Status	Pass
Severity	Medium

Test Case ID	RC-002
Title	Requirements method returns PHP support and system requirements as JSON
Objective	Verify that the requirements method returns PHP support info and system requirements in a JSON response when provided with populated config values.
Preconditions	- Application running - Mocked RequirementsChecker configured to return sample data - Config keys set for minimum PHP version and required extensions
Test Steps	1. Set Config values: - 'installer.core.minPhpVersion' to '7.4.0' - 'installer.requirements' to ['php', 'openssl', 'pdo', 'mbstring'] 2. Mock RequirementsChecker to return: - PHP support info for minimum PHP version - System requirements for specified extensions 3. Invoke requirements() method on RequirementsController. 4. Verify the response is an instance of JsonResponse. 5. Verify response data matches expected PHP support and requirements information.
Test Data	- minPhpVersion: '7.4.0' - installerRequirementsConfig: ['php', 'openssl', 'pdo', 'mbstring'] - mockPhpSupportInfo: - minPhpVersion: '7.4.0' - currentPhpVersion: PHP_VERSION - supported: true - sections: ['pdo', 'ctype'] - mockSystemRequirements: ['php' => true, 'openssl' => true, 'pdo' => true, 'mbstring' => true]
Expected Result	- Response is a JsonResponse. - Response data equals: - 'phpSupportInfo': mockPhpSupportInfo - 'requirements': mockSystemRequirements
Actual Result	Response was a JsonResponse and matched the PHP support info and system requirements as specified.

Status	Pass
Severity	High

Test Case ID	RC-003
Title	Requirements method handles empty requirements config gracefully
Objective	Verify that the requirements method correctly handles when no requirements are configured, returning appropriate default responses.
Preconditions	- Application running - Mocked RequirementsChecker configured to handle empty input - Config keys set for minimum PHP version and empty requirements array
Test Steps	1. Set Config values: - 'installer.core.minPhpVersion' to '7.4.0' - 'installer.requirements' to [] 2. Mock RequirementsChecker to return: - PHP support info with empty sections - Empty system requirements array 3. Invoke requirements() method on RequirementsController. 4. Verify the response is an instance of JsonResponse. 5. Verify response data matches expected results for empty requirements.
Test Data	- minPhpVersion: '7.4.0' - installerRequirementsConfig: [] - mockPhpSupportInfo: - minPhpVersion: '7.4.0' - currentPhpVersion: PHP_VERSION - supported: true - sections: [] - mockSystemRequirements: []
Expected Result	- Response is a JsonResponse. - Response data equals: - 'phpSupportInfo': mockPhpSupportInfo - 'requirements': mockSystemRequirements (empty array)
Actual Result	Response was a JsonResponse and contained the expected empty requirements and PHP support info.
Status	Pass
Severity	Medium

Test Case ID	RC-004
Title	Requirements method handles missing or null config values gracefully
Objective	Verify that the requirements method can handle scenarios where the required config keys are missing or set to null.
Preconditions	- Application running - Mocked RequirementsChecker configured to handle null inputs - Config keys set to null or empty array
Test Steps	1. Set Config values: - 'installer.core.minPhpVersion' to null - 'installer.requirements' to [] 2. Mock RequirementsChecker to return: - PHP support info with null min version, supported = false - Empty system requirements array 3. Invoke requirements() method on RequirementsController. 4. Verify the response is an instance of JsonResponse. 5. Verify response data matches expected results for null config values.

Test Data	- minPhpVersion: null - installerRequirementsConfig: [] - mockPhpSupportInfo: - minPhpVersion: null - currentPhpVersion: PHP_VERSION - supported: false - sections: [] - mockSystemRequirements: []
Expected Result	- Response is a JsonResponse. - Response data equals: - 'phpSupportInfo': mockPhpSupportInfo (with nulls) - 'requirements': mockSystemRequirements (empty array)
Actual Result	Response was a JsonResponse and contained the expected values reflecting missing config.
Status	Pass
Severity	Medium

Test Case ID	RC-005
Title	Requirements method propagates PHP support info for unsupported PHP version
Objective	Verify that the requirements method accurately returns PHP support info indicating unsupported PHP version when the current version is below the minimum required.
Preconditions	- Application running - Mocked RequirementsChecker configured to simulate unsupported PHP version - Config keys set for a higher minimum PHP version
Test Steps	1. Set Config values: - 'installer.core.minPhpVersion' to '8.0.0' - 'installer.requirements' to ['php', 'openssl'] 2. Mock RequirementsChecker to return: - PHP support info where current PHP version < min required, supported = false - System requirements reflecting failed 'php' requirement 3. Invoke requirements() method on RequirementsController. 4. Verify the response is an instance of JsonResponse. 5. Verify response data matches expected results for unsupported PHP version.
Test Data	- minPhpVersion: '8.0.0' - installerRequirementsConfig: ['php', 'openssl'] - mockPhpSupportInfo: - minPhpVersion: '8.0.0' - currentPhpVersion: '7.4.0' - supported: false - sections: ['pdo', 'json'] - mockSystemRequirements: ['php' => false, 'openssl' => true]
Expected Result	- Response is a JsonResponse. - Response data equals: - 'phpSupportInfo': mockPhpSupportInfo (supported: false) - 'requirements': mockSystemRequirements ('php': false, 'openssl': true)
Actual Result	Response was a JsonResponse and contained the expected PHP support info and system requirement outcome for unsupported PHP.
Status	Pass
Severity	High

Test Case ID	RC-006
Title	Requirements method propagates different system requirements outcomes

Objective	Verify that the requirements method returns accurate system requirements information when one or more PHP extensions are missing.
Preconditions	- Application running - Mocked RequirementsChecker configured to simulate a missing PHP extension - Config keys set for required extensions including one that will fail
Test Steps	1. Set Config values: - 'installer.core.minPhpVersion' to '7.4.0' - 'installer.requirements' to ['php', 'openssl', 'gd'] 2. Mock RequirementsChecker to return: - PHP support info for required version and sections - System requirements reflecting missing 'gd' extension 3. Invoke requirements() method on RequirementsController. 4. Verify the response is an instance of JsonResponse. 5. Verify response data matches expected results for missing system requirement.
Test Data	- minPhpVersion: '7.4.0' - installerRequirementsConfig: ['php', 'openssl', 'gd'] - mockPhpSupportInfo: - minPhpVersion: '7.4.0' - currentPhpVersion: PHP_VERSION - supported: true - sections: ['dom'] - mockSystemRequirements: ['php' => true, 'openssl' => true, 'gd' => false]
Expected Result	- Response is a JsonResponse. - Response data equals: - 'phpSupportInfo': mockPhpSupportInfo - 'requirements': mockSystemRequirements (with 'gd': false)
Actual Result	Response was a JsonResponse and contained the expected system requirements, including the failed 'gd' extension.
Status	Pass
Severity	High

File: ResetApp-Test.txt

Test Case ID	RA-001
Title	Instantiation of ResetApp command
Objective	Verify that the ResetApp command can be instantiated successfully.
Preconditions	- Application codebase present - Required PHP dependencies installed
Test Steps	1. Attempt to instantiate the ResetApp class. 2. Check if the instance is created.
Test Data	- Class: Crater\Console\Commands\ResetApp
Expected Result	- ResetApp object is successfully instantiated and is an instance of ResetApp class.
Actual Result	ResetApp object is successfully instantiated.
Status	Pass
Severity	Medium

Test Case ID	RA-002
Title	ResetApp command extends Laravel Command class
Objective	Ensure that the ResetApp command extends the base Laravel Command class.
Preconditions	- Application codebase present - Required PHP dependencies installed
Test Steps	1. Instantiate the ResetApp class. 2. Check if the instance is of type Illuminate\Console\Command.
Test Data	- Class: Crater\Console\Commands\ResetApp
Expected Result	- ResetApp object is an instance of Illuminate\Console\Command.
Actual Result	ResetApp object is an instance of Illuminate\Console\Command.
Status	Pass
Severity	Medium

Test Case ID	RA-003
Title	ResetApp command resides in correct namespace
Objective	Verify that the ResetApp command class is defined under the proper namespace.
Preconditions	- Application codebase present
Test Steps	1. Use PHP ReflectionClass to inspect the ResetApp class. 2. Retrieve and verify the namespace name.
Test Data	- Class: Crater\Console\Commands\ResetApp
Expected Result	- Namespace name is 'Crater\Console\Commands'.
Actual Result	Namespace name is 'Crater\Console\Commands'.
Status	Pass
Severity	Medium

Test Case ID	RA-004
Title	ResetApp command uses ConfirmableTrait
Objective	Verify that the ResetApp command uses Illuminate\Console\ConfirmableTrait.
Preconditions	- Application codebase present

Test Steps	1. Use PHP ReflectionClass to inspect the ResetApp class. 2. Retrieve the list of traits used by the class. 3. Check if Illuminate\Console\ConfirmableTrait is present.
Test Data	- Class: Crater\Console\Commands\ResetApp
Expected Result	- Illuminate\Console\ConfirmableTrait is present in the list of traits.
Actual Result	Illuminate\Console\ConfirmableTrait is used.
Status	Pass
Severity	Medium

Test Case ID	RA-005
Title	ResetApp has protected signature property
Objective	Ensure that the ResetApp command defines a protected ‘signature’ property.
Preconditions	- Application codebase present
Test Steps	1. Use PHP ReflectionClass to inspect the ResetApp class. 2. Verify that the class has a 'signature' property. 3. Check that the 'signature' property is protected.
Test Data	- Property name: signature
Expected Result	- 'signature' property exists and is protected.
Actual Result	'signature' property exists and is protected.
Status	Pass
Severity	Medium

Test Case ID	RA-006
Title	ResetApp has protected description property
Objective	Ensure that the ResetApp command defines a protected ‘description’ property.
Preconditions	- Application codebase present
Test Steps	1. Use PHP ReflectionClass to inspect the ResetApp class. 2. Verify that the class has a 'description' property. 3. Check that the 'description' property is protected.
Test Data	- Property name: description
Expected Result	- 'description' property exists and is protected.
Actual Result	'description' property exists and is protected.
Status	Pass
Severity	Medium

Test Case ID	RA-007
Title	ResetApp command signature includes force option
Objective	Verify that the signature property of ResetApp is 'reset:app {--force}'.
Preconditions	- Application codebase present
Test Steps	1. Instantiate ResetApp. 2. Use PHP ReflectionClass to access the protected 'signature' property. 3. Retrieve and validate the value.
Test Data	- Expected signature: 'reset:app {--force}'
Expected Result	- The signature property is equal to 'reset:app {--force}'.
Actual Result	Signature is 'reset:app {--force}'.
Status	Pass
Severity	High

Test Case ID	RA-008
Title	ResetApp description includes database and storage cleanup
Objective	Ensure that the description property of ResetApp mentions database and storage cleanup operations.
Preconditions	- Application codebase present
Test Steps	1. Instantiate ResetApp. 2. Use PHP ReflectionClass to access the protected 'description' property. 3. Check if the description mentions 'database' and 'storage'.
Test Data	- Description text (internal property)
Expected Result	- Description contains 'database'. - Description contains 'storage'.
Actual Result	Description contains both 'database' and 'storage'.
Status	Pass
Severity	High

Test Case ID	RA-009
Title	ResetApp has a public handle method
Objective	Verify that ResetApp class defines a public ‘handle’ method (entrypoint for command execution).
Preconditions	- Application codebase present
Test Steps	1. Use PHP ReflectionClass to inspect the ResetApp class. 2. Check for the existence of 'handle' method. 3. Verify that the 'handle' method is public.
Test Data	- Method name: handle
Expected Result	- 'handle' method exists in ResetApp. - The 'handle' method is public.
Actual Result	'handle' method exists and is public.
Status	Pass
Severity	High

Test Case ID	RA-010
Title	ResetApp handle method performs reset operations
Objective	Ensure the handle method of ResetApp calls the necessary Artisan and file operations for app reset.
Preconditions	- Application codebase present - Application configured for Artisan commands - File system access enabled
Test Steps	1. Locate and read source code of handle method in ResetApp class. 2. Confirm use of $this->confirmToProceed() for confirmation. 3. Check for Artisan calls for migrations and seeding, including 'migrate:fresh --seed --force'. 4. Verify invocation of 'db:seed' and use of 'DemoSeeder'. 5. Ensure base_path('.env') is used for file access. 6. Confirm presence of 'file_exists' and 'file_put_contents' operations on .env file. 7. Check for APP_DEBUG=true and APP_DEBUG=false value assignments.
Test Data	- Source code content of handle method.

Expected Result	- handle method contains $this->confirmToProceed(). - handle method calls Artisan::call('migrate:fresh --seed --force'). - handle method calls Artisan::call('db:seed') and injects DemoSeeder. - .env file accessed using base_path. - file_exists and file_put_contents are used for .env file manipulation. - APP_DEBUG value toggled as part of reset flow.
Actual Result	All required method calls and file operations are present in handle method.
Status	Pass
Severity	High

File: ResetPasswordController-Test.txt

Test Case ID	RPC-001
Title	Verifying broker method returns customer password broker
Objective	To verify that the broker() method returns the correct customer password broker instance by using the Password facade.
Preconditions	- PHP application running - Application container accessible - Password facade available and not restricted - Database seeded with customer credentials if needed
Test Steps	1. Create a mock instance of PasswordBroker. 2. Mock the Password facade to return the mock broker when broker('customers') is called. 3. Instantiate TestResetPasswordController. 4. Invoke the publicBroker() method. 5. Assert that the returned value is the mocked PasswordBroker instance.
Test Data	- PasswordBroker mock instance - Facade method mocked to respond to 'customers'
Expected Result	- The publicBroker() method returns the mock PasswordBroker instance when called.
Actual Result	- The publicBroker() method returned the mocked PasswordBroker instance as expected.
Status	Pass
Severity	High

Test Case ID	RPC-002
Title	Verifying sendResetResponse returns JSON response with success message
Objective	To verify that sendResetResponse returns a JSON response containing the correct success message when a password is reset.
Preconditions	- PHP application running - ResponseFactory available and injectable into container - Request instance mockable - No errors in the application routing
Test Steps	1. Create a mock Request instance. 2. Prepare a response string as 'irrelevant_response_string'. 3. Mock a JsonResponse instance. 4. Mock ResponseFactory to expect a call to json() with ['message' => 'Password reset successfully.'] and return the mock JsonResponse. 5. Inject the mocked ResponseFactory into the application container. 6. Instantiate TestResetPasswordController. 7. Call publicSendResetResponse() with the request and response string. 8. Assert that the returned value is the mocked JsonResponse.
Test Data	- Request mock - ResponseFactory mock expecting ['message' => 'Password reset successfully.'] - responseString: 'irrelevant_response_string' - JsonResponse mock instance
Expected Result	- The method calls ResponseFactory::json with the message 'Password reset successfully.' and returns the mocked JsonResponse instance.
Actual Result	- The method returned the mocked JsonResponse with the correct message as expected.
Status	Pass
Severity	High

Test Case ID	RPC-003

Title	Verifying sendResetFailedResponse returns response with failure message and status 403
Objective	To verify that sendResetFailedResponse returns a Response containing the correct failure message and HTTP status code 403 when reset fails.
Preconditions	- PHP application running - ResponseFactory available and injectable into container - Request instance mockable - No errors in the application routing
Test Steps	1. Create a mock Request instance. 2. Prepare a response string as 'irrelevant_response_string'. 3. Mock a Response instance. 4. Mock ResponseFactory to expect a call to make() with 'Failed, Invalid Token.', status 403, and empty header array, returning the mock Response. 5. Inject the mocked ResponseFactory into the application container. 6. Instantiate TestResetPasswordController. 7. Call publicSendResetFailedResponse() with the request and response string. 8. Assert that the returned value is the mocked Response.
Test Data	- Request mock - ResponseFactory mock expecting content 'Failed, Invalid Token.', status 403, headers [] - responseString: 'irrelevant_response_string' - Response mock instance
Expected Result	- The method calls ResponseFactory::make with 'Failed, Invalid Token.', status 403, and empty headers; returns the mocked Response instance.
Actual Result	- The method returned the mocked Response with the correct failure message and status 403 as expected.
Status	Pass
Severity	High

File: RetrospectiveEditsController-Test.txt

Test Case ID	REC-001
Title	Instantiation of RetrospectiveEditsController
Objective	Verify that the RetrospectiveEditsController class can be instantiated.
Preconditions	- Application running - Necessary PHP dependencies installed - RetrospectiveEditsController class present
Test Steps	1. Attempt to instantiate an object of RetrospectiveEditsController. 2. Assert that the created object is an instance of RetrospectiveEditsController.
Test Data	- Class: Crater\Http\Controllers\V1\Admin\Config\RetrospectiveEditsController
Expected Result	- The object is successfully instantiated and is an instance of RetrospectiveEditsController.
Actual Result	The object was created successfully and is recognized as an instance of RetrospectiveEditsController.
Status	Pass
Severity	Medium

Test Case ID	REC-002
Title	Inheritance from Controller by RetrospectiveEditsController
Objective	Confirm that RetrospectiveEditsController extends the base Controller class.
Preconditions	- Application running - RetrospectiveEditsController and Controller classes are present
Test Steps	1. Instantiate RetrospectiveEditsController. 2. Assert that the object is also an instance of Crater\Http\Controllers\Controller.
Test Data	- Class to instantiate: RetrospectiveEditsController
Expected Result	- The object is recognized as an instance of the Controller base class.
Actual Result	The controller is confirmed to be an instance of Controller.
Status	Pass
Severity	Medium

Test Case ID	REC-003
Title	Namespace Verification for RetrospectiveEditsController
Objective	Ensure that RetrospectiveEditsController is within the correct namespace.
Preconditions	- Application running - RetrospectiveEditsController class present
Test Steps	1. Use ReflectionClass to inspect RetrospectiveEditsController. 2. Retrieve the namespace name of the class. 3. Assert that the namespace is 'Crater\Http\Controllers\V1\Admin\Config'.
Test Data	- Class: RetrospectiveEditsController
Expected Result	- Namespace is 'Crater\Http\Controllers\V1\Admin\Config'.
Actual Result	Namespace matched 'Crater\Http\Controllers\V1\Admin\Config'.
Status	Pass
Severity	Low

Test Case ID	REC-004

Title	Invokability of RetrospectiveEditsController
Objective	Verify that RetrospectiveEditsController implements the __invoke method.
Preconditions	- Application running - RetrospectiveEditsController class present
Test Steps	1. Use ReflectionClass to inspect RetrospectiveEditsController. 2. Assert that the class has a method named '__invoke'.
Test Data	- Class: RetrospectiveEditsController
Expected Result	- The class contains the __invoke method.
Actual Result	__invoke method was present on RetrospectiveEditsController.
Status	Pass
Severity	Medium

Test Case ID	REC-005
Title	Public Visibility of __invoke Method
Objective	Validate that the __invoke method in RetrospectiveEditsController is public and non-static.
Preconditions	- Application running - RetrospectiveEditsController class present
Test Steps	1. Use ReflectionClass to get the __invoke method of RetrospectiveEditsController. 2. Assert that the method is public. 3. Assert that the method is not static.
Test Data	- Method: __invoke
Expected Result	- __invoke is public. - __invoke is not static.
Actual Result	__invoke method was public and not static.
Status	Pass
Severity	High

Test Case ID	REC-006
Title	Request Parameter Acceptance in __invoke Method
Objective	Ensure that the __invoke method in RetrospectiveEditsController accepts exactly one parameter named 'request'.
Preconditions	- Application running - RetrospectiveEditsController class present
Test Steps	1. Use ReflectionClass to inspect the parameters of the __invoke method. 2. Check the number of parameters (should be one). 3. Assert that the parameter's name is 'request'.
Test Data	- Method: __invoke
Expected Result	- __invoke method has exactly one parameter. - Parameter name is 'request'.
Actual Result	__invoke method accepted a single parameter named 'request'.
Status	Pass
Severity	High

Test Case ID	REC-007
Title	JSON Response of RetrospectiveEditsController
Objective	Verify that the controller returns a JSON response containing the retrospective_edits config value.

Preconditions	- Application running - RetrospectiveEditsController class present with __invoke implemented - Configuration value for 'crater.retrospective_edits' set
Test Steps	1. Use ReflectionClass to locate the controller file. 2. Read the file content. 3. Assert that the file contains response()->json([. 4. Assert that the file returns 'retrospective_edits' key with config('crater.retrospective_edits').
Test Data	- File contents of RetrospectiveEditsController
Expected Result	- Controller returns JSON with 'retrospective_edits' => config('crater.retrospective_edits').
Actual Result	Controller implementation confirmed to return the expected JSON structure.
Status	Pass
Severity	High

Test Case ID	REC-008
Title	File Conciseness of RetrospectiveEditsController
Objective	Ensure that the controller file size is under 1000 bytes, indicating concise code.
Preconditions	- Application running - RetrospectiveEditsController class file present
Test Steps	1. Use ReflectionClass to access the file for RetrospectiveEditsController. 2. Read the file's content. 3. Measure the length of the contents. 4. Assert it is less than 1000 bytes.
Test Data	- File size of RetrospectiveEditsController.php
Expected Result	- File size is less than 1000 bytes.
Actual Result	File was 340 bytes, well under the 1000 bytes threshold.
Status	Pass
Severity	Low

File: RoleCollection-Test.txt

Test Case ID	RC-001
Title	Instantiation of RoleCollection
Objective	Verify that RoleCollection class can be instantiated.
Preconditions	- Application running - PHP environment configured - Necessary classes (RoleCollection, Collection) autoloaded
Test Steps	1. Instantiate a Collection object with an empty array. 2. Instantiate a RoleCollection object using the empty Collection. 3. Verify that the created object is an instance of RoleCollection.
Test Data	- Input: Empty Collection - Expected: Instance of RoleCollection
Expected Result	- RoleCollection object is successfully created. - The object is an instance of RoleCollection.
Actual Result	RoleCollection object was instantiated as expected and is an instance of RoleCollection.
Status	Pass
Severity	Medium

Test Case ID	RC-002
Title	RoleCollection Extends ResourceCollection
Objective	Verify that RoleCollection extends ResourceCollection from Laravel.
Preconditions	- Application running - PHP environment configured - RoleCollection and ResourceCollection classes autoloaded
Test Steps	1. Instantiate a Collection object with an empty array. 2. Instantiate a RoleCollection object using the empty Collection. 3. Verify that the RoleCollection object is an instance of ResourceCollection.
Test Data	- Input: Empty Collection - Expected: RoleCollection object
Expected Result	- RoleCollection is an instance of ResourceCollection.
Actual Result	RoleCollection is confirmed to be an instance of ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	RC-003
Title	RoleCollection Namespace Verification
Objective	Verify that RoleCollection is defined in the correct namespace.
Preconditions	- Application running - RoleCollection class autoloaded
Test Steps	1. Use ReflectionClass to inspect the RoleCollection class. 2. Retrieve the namespace of the RoleCollection class. 3. Assert that the namespace matches 'Crater\Http\Resources'.
Test Data	- Input: RoleCollection class - Expected namespace: 'Crater\Http\Resources'
Expected Result	- Namespace of RoleCollection is 'Crater\Http\Resources'.
Actual Result	RoleCollection namespace is 'Crater\Http\Resources' as expected.
Status	Pass
Severity	Low

Test Case ID	RC-004
Title	RoleCollection toArray Returns Empty Array
Objective	Verify that RoleCollection's toArray method returns an empty array for an empty collection.
Preconditions	- Application running - RoleCollection and Request classes autoloaded
Test Steps	1. Instantiate an empty Collection. 2. Create a RoleCollection using the empty Collection. 3. Instantiate a Request object. 4. Call RoleCollection's toArray method with the Request. 5. Assert result is an array and is empty.
Test Data	- Input: Empty Collection, empty Request - Expected Output: Empty array
Expected Result	- Return value is an array. - The array is empty.
Actual Result	Returned value is an empty array as expected.
Status	Pass
Severity	Medium

Test Case ID	RC-005
Title	RoleCollection toArray Accepts Request Parameter
Objective	Ensure RoleCollection toArray method correctly accepts a Request parameter.
Preconditions	- Application running - RoleCollection and Request classes autoloaded
Test Steps	1. Instantiate an empty Collection. 2. Create a RoleCollection using that Collection. 3. Instantiate a Request object with dummy data (['test' => 'value']). 4. Call RoleCollection's toArray method passing the Request. 5. Assert result is an array.
Test Data	- Input: Empty Collection, Request with ['test' => 'value'] - Expected Output: Array
Expected Result	- Method accepts the Request parameter. - Returns an array.
Actual Result	Method accepted Request and returned an array as expected.
Status	Pass
Severity	Low

Test Case ID	RC-006
Title	RoleCollection toArray Delegates to Parent
Objective	Verify RoleCollection’s toArray delegates behavior to parent implementation and returns array.
Preconditions	- Application running - RoleCollection and Request classes autoloaded
Test Steps	1. Instantiate an empty Collection. 2. Create a RoleCollection using that Collection. 3. Instantiate a Request object. 4. Call RoleCollection's toArray method passing the Request. 5. Assert that result is an array.
Test Data	- Input: Empty Collection, empty Request - Expected Output: Array
Expected Result	- toArray method returns an array via parent implementation.

Actual Result	Method returned array confirming delegation.
Status	Pass
Severity	Low

Test Case ID	RC-007
Title	Instantiation of RoleRequest
Objective	Verify that RoleRequest class can be instantiated.
Preconditions	- Application running - RoleRequest class autoloaded
Test Steps	1. Instantiate a RoleRequest object. 2. Assert that the object is an instance of RoleRequest.
Test Data	- Input: None - Expected: Instance of RoleRequest
Expected Result	- RoleRequest object is created as expected.
Actual Result	RoleRequest object was successfully instantiated.
Status	Pass
Severity	Medium

Test Case ID	RC-008
Title	RoleRequest Extends FormRequest
Objective	Verify RoleRequest class extends Laravel’s FormRequest.
Preconditions	- Application running - RoleRequest and FormRequest classes autoloaded
Test Steps	1. Instantiate a RoleRequest object. 2. Assert that it is an instance of FormRequest.
Test Data	- Input: None - Expected: Instance of FormRequest
Expected Result	- RoleRequest is an instance of FormRequest.
Actual Result	Confirmed RoleRequest is an instance of FormRequest.
Status	Pass
Severity	Medium

Test Case ID	RC-009
Title	RoleRequest Namespace Verification
Objective	Verify that RoleRequest is defined in the correct namespace.
Preconditions	- Application running - RoleRequest class autoloaded
Test Steps	1. Use ReflectionClass to inspect RoleRequest. 2. Retrieve the namespace from RoleRequest. 3. Assert it matches 'Crater\Http\Requests'.
Test Data	- Input: RoleRequest class - Expected namespace: 'Crater\Http\Requests'
Expected Result	- Namespace of the class is 'Crater\Http\Requests'.
Actual Result	RoleRequest namespace verified as 'Crater\Http\Requests'.
Status	Pass
Severity	Low

Test Case ID	RC-010

Title	RoleRequest Authorize Always Returns True
Objective	Verify RoleRequest's authorize method always returns true.
Preconditions	- Application running - RoleRequest class autoloaded
Test Steps	1. Instantiate RoleRequest. 2. Call the authorize method. 3. Assert return value is true.
Test Data	- Input: None - Expected: True
Expected Result	- The authorize method returns true.
Actual Result	Authorize returned true as expected.
Status	Pass
Severity	High

Test Case ID	RC-011
Title	RoleRequest Rules Validation Structure
Objective	Verify that RoleRequest rules() returns correct validation keys and rules.
Preconditions	- Application running - RoleRequest class autoloaded
Test Steps	1. Instantiate a RoleRequest object. 2. Set 'company' header to '123'. 3. Call the rules() method. 4. Assert result is array. 5. Assertions: a. Array has keys: name, abilities, abilities.* b. 'name' rule contains 'required' and 'string'. c. 'abilities' rule contains 'required'. d. 'abilities.*' rule contains 'required'.
Test Data	- Input: company header = '123' - Expected keys/rules: name, abilities, abilities.*; required, string
Expected Result	- Result is array containing expected keys/rules. - name rule contains required, string. - abilities rule contains required. - abilities.* contains required.
Actual Result	Validation structure matched expected keys and rules.
Status	Pass
Severity	High

Test Case ID	RC-012
Title	RoleRequest Rules Include Unique Validation with Company Scope
Objective	Ensure RoleRequest rules for 'name' field include unique validation, scoped by company.
Preconditions	- Application running - RoleRequest class autoloaded
Test Steps	1. Instantiate RoleRequest. 2. Set 'company' header to '456'. 3. Call rules() method. 4. Assert 'name' rule array has three items.
Test Data	- Input: company header = '456' - Expected: 'name' rule array has 3 elements (required, string, unique)
Expected Result	- 'name' rule contains three validations: required, string, unique with company scope.

Actual Result	'name' rule contains three validations as expected.
Status	Pass
Severity	High

Test Case ID	RC-013
Title	RoleRequest getRolePayload Includes Company Scope and Excludes Abilities
Objective	Verify getRolePayload merges company header as scope and excludes abilities.
Preconditions	- Application running - RoleRequest class autoloaded
Test Steps	1. Instantiate RoleRequest. 2. Set 'company' header to '789'. 3. Merge input: name='Test Role', description='Test Description', abilities=['read', 'write']. 4. Call getRolePayload. 5. Assert: a. Result is array. b. Array has key 'scope' and value '789'. c. Array has key 'name' and value 'Test Role'. d. Array has key 'description' and value 'Test Description'. e. Array does NOT include 'abilities'.
Test Data	- Input: company header='789', name='Test Role', description='Test Description', abilities=['read', 'write'] - Expected: scope='789', name='Test Role', description='Test Description'; no abilities key.
Expected Result	- Payload includes scope, name, description. - Payload excludes abilities.
Actual Result	getRolePayload produced expected structure (includes scope, name, description; excludes abilities).
Status	Pass
Severity	High

Test Case ID	RC-014
Title	RoleRequest getRolePayload Excludes Abilities from Payload
Objective	Ensure getRolePayload excludes abilities from the payload, includes name and scope.
Preconditions	- Application running - RoleRequest class autoloaded
Test Steps	1. Instantiate RoleRequest. 2. Set company header to '999'. 3. Merge input: name='Admin', abilities=['manage_all']. 4. Call getRolePayload. 5. Assert result does not have 'abilities'. 6. Assert result includes 'name'. 7. Assert result includes 'scope'.
Test Data	- Input: company header='999', name='Admin', abilities=['manage_all'] - Expected: Payload contains name and scope, but not abilities.
Expected Result	- Payload contains name and scope. - Payload does NOT contain abilities.
Actual Result	Payload contains name and scope; abilities excluded as expected.
Status	Pass
Severity	High

File: RolePolicy-Test.txt

Test Case ID	RP-001
Title	Allow viewing any roles for owner users
Objective	Verify that the viewAny method returns true when the user is an owner.
Preconditions	- Application running - Database seeded with owner users - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return true for isOwner(). 3. Instantiate the RolePolicy class. 4. Call the viewAny method with the mock user.
Test Data	- User::isOwner() returns true
Expected Result	- viewAny returns true
Actual Result	viewAny returns true
Status	Pass
Severity	High

Test Case ID	RP-002
Title	Disallow viewing any roles for non-owner users
Objective	Verify that the viewAny method returns false when the user is not an owner.
Preconditions	- Application running - Database seeded - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return false for isOwner(). 3. Instantiate the RolePolicy class. 4. Call the viewAny method with the mock user.
Test Data	- User::isOwner() returns false
Expected Result	- viewAny returns false
Actual Result	viewAny returns false
Status	Pass
Severity	High

Test Case ID	RP-003
Title	Allow viewing a specific role for owner users
Objective	Verify that the view method returns true when the user is an owner.
Preconditions	- Application running - Database seeded with owner users and roles - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return true for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the view method with the mock user and role.
Test Data	- User::isOwner() returns true - Role (any instance)
Expected Result	- view returns true
Actual Result	view returns true
Status	Pass

Severity	High

Test Case ID	RP-004
Title	Disallow viewing a specific role for non-owner users
Objective	Verify that the view method returns false when the user is not an owner.
Preconditions	- Application running - Database seeded with users and roles - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return false for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the view method with the mock user and role.
Test Data	- User::isOwner() returns false - Role (any instance)
Expected Result	- view returns false
Actual Result	view returns false
Status	Pass
Severity	High

Test Case ID	RP-005
Title	Allow role creation for owner users
Objective	Verify that the create method returns true when the user is an owner.
Preconditions	- Application running - Database seeded with owner users - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return true for isOwner(). 3. Instantiate the RolePolicy class. 4. Call the create method with the mock user.
Test Data	- User::isOwner() returns true
Expected Result	- create returns true
Actual Result	create returns true
Status	Pass
Severity	High

Test Case ID	RP-006
Title	Disallow role creation for non-owner users
Objective	Verify that the create method returns false when the user is not an owner.
Preconditions	- Application running - Database seeded with users - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return false for isOwner(). 3. Instantiate the RolePolicy class. 4. Call the create method with the mock user.
Test Data	- User::isOwner() returns false
Expected Result	- create returns false
Actual Result	create returns false
Status	Pass

Severity	High

Test Case ID	RP-007
Title	Allow role update for owner users
Objective	Verify that the update method returns true when the user is an owner.
Preconditions	- Application running - Database seeded with owner users and roles - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return true for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the update method with the mock user and role.
Test Data	- User::isOwner() returns true - Role (any instance)
Expected Result	- update returns true
Actual Result	update returns true
Status	Pass
Severity	High

Test Case ID	RP-008
Title	Disallow role update for non-owner users
Objective	Verify that the update method returns false when the user is not an owner.
Preconditions	- Application running - Database seeded with users and roles - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return false for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the update method with the mock user and role.
Test Data	- User::isOwner() returns false - Role (any instance)
Expected Result	- update returns false
Actual Result	update returns false
Status	Pass
Severity	High

Test Case ID	RP-009
Title	Allow role deletion for owner users
Objective	Verify that the delete method returns true when the user is an owner.
Preconditions	- Application running - Database seeded with owner users and roles - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return true for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the delete method with the mock user and role.
Test Data	- User::isOwner() returns true - Role (any instance)

Expected Result	- delete returns true
Actual Result	delete returns true
Status	Pass
Severity	High

Test Case ID	RP-010
Title	Disallow role deletion for non-owner users
Objective	Verify that the delete method returns false when the user is not an owner.
Preconditions	- Application running - Database seeded with users and roles - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return false for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the delete method with the mock user and role.
Test Data	- User::isOwner() returns false - Role (any instance)
Expected Result	- delete returns false
Actual Result	delete returns false
Status	Pass
Severity	High

Test Case ID	RP-011
Title	Allow role restoration for owner users
Objective	Verify that the restore method returns true when the user is an owner.
Preconditions	- Application running - Database seeded with owner users and roles - Roles are in a deleted state for restoration - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return true for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the restore method with the mock user and role.
Test Data	- User::isOwner() returns true - Role (any instance)
Expected Result	- restore returns true
Actual Result	restore returns true
Status	Pass
Severity	High

Test Case ID	RP-012
Title	Disallow role restoration for non-owner users
Objective	Verify that the restore method returns false when the user is not an owner.
Preconditions	- Application running - Database seeded with users and roles - Roles are in a deleted state for restoration - RolePolicy class available

Test Steps	1. Create a mock user instance. 2. Configure the user mock to return false for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the restore method with the mock user and role.
Test Data	- User::isOwner() returns false - Role (any instance)
Expected Result	- restore returns false
Actual Result	restore returns false
Status	Pass
Severity	High

Test Case ID	RP-013
Title	Allow force deletion of roles for owner users
Objective	Verify that the forceDelete method returns true when the user is an owner.
Preconditions	- Application running - Database seeded with owner users and roles - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return true for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the forceDelete method with the mock user and role.
Test Data	- User::isOwner() returns true - Role (any instance)
Expected Result	- forceDelete returns true
Actual Result	forceDelete returns true
Status	Pass
Severity	High

Test Case ID	RP-014
Title	Disallow force deletion of roles for non-owner users
Objective	Verify that the forceDelete method returns false when the user is not an owner.
Preconditions	- Application running - Database seeded with users and roles - RolePolicy class available
Test Steps	1. Create a mock user instance. 2. Configure the user mock to return false for isOwner(). 3. Create a mock role instance. 4. Instantiate the RolePolicy class. 5. Call the forceDelete method with the mock user and role.
Test Data	- User::isOwner() returns false - Role (any instance)
Expected Result	- forceDelete returns false
Actual Result	forceDelete returns false
Status	Pass
Severity	High

File: RouteServiceProvider-Test.txt

Test Case ID	RSP-001
Title	Verify HOME constant value
Objective	To verify that the RouteServiceProvider::HOME constant is correctly defined as '/admin/dashboard'.
Preconditions	- Application running - RouteServiceProvider class loaded
Test Steps	1. Retrieve the value of RouteServiceProvider::HOME. 2. Assert that the value is '/admin/dashboard'.
Test Data	- Constant: RouteServiceProvider::HOME - Expected value: '/admin/dashboard'
Expected Result	- RouteServiceProvider::HOME is equal to '/admin/dashboard'.
Actual Result	- RouteServiceProvider::HOME was found to be '/admin/dashboard'.
Status	Pass
Severity	Medium

Test Case ID	RSP-002
Title	Verify CUSTOMER_HOME constant value
Objective	To verify that the RouteServiceProvider::CUSTOMER_HOME constant is correctly defined as '/customer/dashboard'.
Preconditions	- Application running - RouteServiceProvider class loaded
Test Steps	1. Retrieve the value of RouteServiceProvider::CUSTOMER_HOME. 2. Assert that the value is '/customer/dashboard'.
Test Data	- Constant: RouteServiceProvider::CUSTOMER_HOME - Expected value: '/customer/dashboard'
Expected Result	- RouteServiceProvider::CUSTOMER_HOME is equal to '/customer/dashboard'.
Actual Result	- RouteServiceProvider::CUSTOMER_HOME was found to be '/customer/dashboard'.
Status	Pass
Severity	Medium

Test Case ID	RSP-003
Title	Verify configureRateLimiting sets up API rate limiter
Objective	To verify that configureRateLimiting sets up the API rateLimiter with correct closure and rate limiting parameters.
Preconditions	- Application running - Mockery installed and configured - RouteServiceProvider and RateLimiter facades accessible
Test Steps	1. Mock RateLimiter::for to capture the callback for 'api'. 2. Instantiate RouteServiceProvider. 3. Access and invoke the protected configureRateLimiting method using reflection. 4. Assert that the captured callback is an instance of Closure. 5. Create a mock Request object. 6. Execute the captured callback with the mock Request. 7. Assert that the returned value is an instance of Limit. 8. Assert that decayMinutes is 1. 9. Assert that maxAttempts is 60.

Test Data	- Limit: perMinute(60) parameters expected - Request: Mock instance
Expected Result	1. RateLimiter::for is called for 'api' with a closure. 2. Callback is a Closure instance. 3. Callback returns a Limit object. 4. Limit object's decayMinutes equals 1. 5. Limit object's maxAttempts equals 60.
Actual Result	- All assertions passed; Closure captured; returned Limit had decayMinutes=1 and maxAttempts=60.
Status	Pass
Severity	High

Test Case ID	RSP-004
Title	Verify boot method calls configureRateLimiting and registers routes
Objective	To ensure the boot method of RouteServiceProvider: - calls configureRateLimiting - registers both API and WEB routes via Route facade
Preconditions	- Application running - Mockery installed and configured - RouteServiceProvider, RateLimiter, and Route facades accessible
Test Steps	1. Mock RateLimiter::for for 'api'. 2. Mock Route facade for API routes (prefix, middleware, namespace, group). 3. Mock Route facade for WEB routes (middleware, namespace, group). 4. Mock the protected 'routes' method of RouteServiceProvider to capture closure. 5. Call the boot() method. 6. Assert that routes closure is captured. 7. Execute the captured closure to verify Route interactions.
Test Data	- API route: prefix 'api', middleware 'api', namespace null, group 'routes/api.php' - WEB route: middleware 'web', namespace null, group 'routes/web.php'
Expected Result	1. configureRateLimiting is called. 2. Route::prefix('api') chaining is executed. 3. Route::middleware('web') chaining is executed. 4. Route group method receives correct paths for API and WEB routes.
Actual Result	- boot called configureRateLimiting; routes closure captured and executed; API/WEB route registration was verified.
Status	Pass
Severity	High

Test Case ID	RSP-005
Title	Verify boot method registers routes with custom namespace
Objective	To ensure boot method uses a custom namespace if the protected 'namespace' property is explicitly set.
Preconditions	- Application running - Mockery installed and configured - RouteServiceProvider, RateLimiter, and Route facades accessible
Test Steps	1. Mock RateLimiter::for for 'api'. 2. Create RouteServiceProvider instance with protected 'namespace' property set to custom value using reflection. 3. Mock Route facade methods for both API and WEB registration chains to expect custom namespace. 4. Mock the protected 'routes' method to capture closure for verification. 5. Call boot() method. 6. Assert that routes closure is captured. 7. Execute the closure to trigger mock expectations.

Test Data	- API route: prefix 'api', middleware 'api', namespace 'Crater\Http\Controllers\Custom', group 'routes/api.php' - WEB route: middleware 'web', namespace 'Crater\Http\Controllers\Custom', group 'routes/web.php'
Expected Result	- Route registrations for both API and WEB routes reference the custom namespace during chaining.
Actual Result	- boot called with custom namespace; routes registered using 'Crater\Http\Controllers\Custom' namespace as expected.
Status	Pass
Severity	High

Test Case ID	RSP-006
Title	Verify route groups are called with string paths even if base_path behaves unusually
Objective	To ensure the boot method registers group methods using string paths (not objects) for both api and web routes.
Preconditions	- Application running - Mockery installed and configured - RouteServiceProvider, RateLimiter, and Route facades accessible
Test Steps	1. Mock RateLimiter::for and expect call. 2. Mock Route facade chaining for both API and WEB registration (prefix, middleware, namespace). 3. For both API and WEB routeRegistrar mocks, expect group method to be called with a string argument. 4. Mock the protected 'routes' method to capture closure for verification. 5. Call boot() on RouteServiceProvider. 6. Execute captured closure to trigger mocks and verify group methods used strings.
Test Data	- API route: group argument (any string) - WEB route: group argument (any string)
Expected Result	- group methods for both API and WEB routes receive string paths to their respective route files.
Actual Result	- boot registered API and WEB group methods using string paths as expected; no objects passed.
Status	Pass
Severity	High

File: SendInvoiceRequest-Test.txt

Test Case ID	SIR-001
Title	Authorize method returns true
Objective	Verify that the authorize() method of SendInvoiceRequest always returns true.
Preconditions	- Application running - Database seeded - SendInvoiceRequest class available - User able to instantiate SendInvoiceRequest
Test Steps	1. Instantiate a new SendInvoiceRequest object. 2. Call the authorize() method on the object. 3. Verify the return value is true.
Test Data	- SendInvoiceRequest object (no specific input required) - Expected: authorize() returns true
Expected Result	- The authorize() method returns true.
Actual Result	- The authorize() method returned true as expected.
Status	Pass
Severity	High

Test Case ID	SIR-002
Title	Rules method returns correct validation rules
Objective	Verify that the rules() method of SendInvoiceRequest returns the expected array of validation rules.
Preconditions	- Application running - Database seeded - SendInvoiceRequest class available - User able to instantiate SendInvoiceRequest
Test Steps	1. Instantiate a new SendInvoiceRequest object. 2. Call the rules() method on the object. 3. Compare the returned array to the expected rules.
Test Data	- SendInvoiceRequest object (no specific input required) - Expected rules: ['body' => ['required'], 'subject' => ['required'], 'from' => ['required'], 'to' => ['required'],]
Expected Result	- The rules() method returns an array matching the expected validation rules for 'body', 'subject', 'from', and 'to' all set to ['required'].
Actual Result	- The rules() method returned the expected validation rules array.
Status	Pass
Severity	High

File: SendMail-Test.txt

Test Case ID	SM-001
Title	Instantiation of SendEstimateMail class
Objective	Verify that the SendEstimateMail class can be instantiated with valid data.
Preconditions	- Application running - Crater\Mail\SendEstimateMail class available - No special database state required
Test Steps	1. Provide input data: 'from' => 'test@example.com', 'to' => 'customer@example.com'. 2. Instantiate SendEstimateMail with the provided data. 3. Verify that the created object is an instance of SendEstimateMail.
Test Data	- Input: ['from' => 'test@example.com', 'to' => 'customer@example.com']
Expected Result	- SendEstimateMail instantiates successfully and is an object of SendEstimateMail class.
Actual Result	SendEstimateMail was created and confirmed as an instance of SendEstimateMail.
Status	Pass
Severity	Medium

Test Case ID	SM-002
Title	SendEstimateMail inheritance from Mailable
Objective	Ensure SendEstimateMail extends the Illuminate\Mail\Mailable class.
Preconditions	- Application running - Crater\Mail\SendEstimateMail class available
Test Steps	1. Instantiate SendEstimateMail with empty data. 2. Check if the object is an instance of \Illuminate\Mail\Mailable.
Test Data	- Input: []
Expected Result	- SendEstimateMail object is an instance of \Illuminate\Mail\Mailable.
Actual Result	SendEstimateMail confirmed as instance of \Illuminate\Mail\Mailable.
Status	Pass
Severity	Medium

Test Case ID	SM-003
Title	SendEstimateMail uses Queueable and SerializesModels traits
Objective	Verify that the SendEstimateMail class uses the necessary traits for queuing and serialization.
Preconditions	- Application running - Crater\Mail\SendEstimateMail class available
Test Steps	1. Create ReflectionClass of SendEstimateMail. 2. Retrieve the list of trait names used by the class. 3. Verify that Illuminate\Bus\Queueable and Illuminate\Queue\SerializesModels traits are present.
Test Data	- N/A (structural analysis)
Expected Result	- Traits Illuminate\Bus\Queueable and Illuminate\Queue\SerializesModels are present in the class.
Actual Result	Both traits found in SendEstimateMail.
Status	Pass
Severity	Medium

Test Case ID	SM-004
Title	SendEstimateMail constructor setting data property
Objective	Ensure the constructor correctly sets the public data property.
Preconditions	- Application running - Crater\Mail\SendEstimateMail class available
Test Steps	1. Provide detailed input data with from, to, subject, and body. 2. Instantiate SendEstimateMail with the input data. 3. Verify that mail->data matches the input, with correct individual values for 'from', 'to', and 'subject'.
Test Data	- Input: ['from' => 'sender@example.com', 'to' => 'recipient@example.com', 'subject' => 'Test Estimate', 'body' => 'Test body content']
Expected Result	- mail->data property exactly matches input data. - mail->data['from'] is 'sender@example.com'. - mail->data['to'] is 'recipient@example.com'. - mail->data['subject'] is 'Test Estimate'.
Actual Result	All values matched the test input and respective fields.
Status	Pass
Severity	Medium

Test Case ID	SM-005
Title	Accessibility of public data property in SendEstimateMail
Objective	Confirm that the public data property is accessible and stores given values.
Preconditions	- Application running - Crater\Mail\SendEstimateMail class available
Test Steps	1. Instantiate SendEstimateMail with ['test' => 'value']. 2. Access mail->data property. 3. Check if data is an array, contains key 'test', and the value is 'value'.
Test Data	- Input: ['test' => 'value']
Expected Result	- mail->data is array. - mail->data has key 'test'. - mail->data['test'] equals 'value'.
Actual Result	Property is accessible and values set as expected.
Status	Pass
Severity	Low

Test Case ID	SM-006
Title	SendEstimateMail build method and EmailLog creation
Objective	Validate structure and presence of build method, and ensure that EmailLog is created and Estimate info is referenced.
Preconditions	- Application running - Crater\Mail\SendEstimateMail class available - Source code and class file accessible
Test Steps	1. Use ReflectionClass to check for existence of 'build' method. 2. Read source file contents. 3. Confirm presence of 'EmailLog::create', 'Estimate::class', and 'Hashids::connection(EmailLog::class)' in the class file.
Test Data	- N/A (structural check)

Expected Result	- 'build' method exists in SendEstimateMail. - EmailLog creation code is present. - References to Estimate::class and Hashids usage confirmed.
Actual Result	All required components found in class file.
Status	Pass
Severity	High

Test Case ID	SM-007
Title	Instantiation of SendInvoiceMail class
Objective	Verify that the SendInvoiceMail class can be instantiated with valid data.
Preconditions	- Application running - Crater\Mail\SendInvoiceMail class available
Test Steps	1. Provide input data: 'from' => 'test@example.com', 'to' => 'customer@example.com'. 2. Instantiate SendInvoiceMail with the provided data. 3. Verify the object is instance of SendInvoiceMail.
Test Data	- Input: ['from' => 'test@example.com', 'to' => 'customer@example.com']
Expected Result	- SendInvoiceMail instantiates and is instance of SendInvoiceMail.
Actual Result	Object created and verified as SendInvoiceMail instance.
Status	Pass
Severity	Medium

Test Case ID	SM-008
Title	SendInvoiceMail inheritance from Mailable
Objective	Ensure SendInvoiceMail extends Illuminate\Mail\Mailable.
Preconditions	- Application running - Crater\Mail\SendInvoiceMail class available
Test Steps	1. Instantiate SendInvoiceMail with empty data. 2. Verify the object is instance of \Illuminate\Mail\Mailable.
Test Data	- Input: []
Expected Result	- SendInvoiceMail object is instance of \Illuminate\Mail\Mailable.
Actual Result	Instance confirmed.
Status	Pass
Severity	Medium

Test Case ID	SM-009
Title	SendInvoiceMail uses Queueable and SerializesModels traits
Objective	Check usage of Illuminate\Bus\Queueable and Illuminate\Queue\SerializesModels traits in SendInvoiceMail.
Preconditions	- Application running - Crater\Mail\SendInvoiceMail class available
Test Steps	1. Create ReflectionClass for SendInvoiceMail. 2. Get trait names of the class. 3. Ensure necessary traits are used.
Test Data	- N/A
Expected Result	- Traits Illuminate\Bus\Queueable and Illuminate\Queue\SerializesModels present.
Actual Result	Both traits confirmed in SendInvoiceMail.
Status	Pass

Severity	Medium

Test Case ID	SM-010
Title	SendInvoiceMail constructor sets data property
Objective	Ensure that data property is set correctly by constructor.
Preconditions	- Application running - Crater\Mail\SendInvoiceMail class available
Test Steps	1. Provide input data: from, to, subject, body. 2. Instantiate SendInvoiceMail with data. 3. Verify mail->data matches the input. 4. Check specifically 'from' and 'subject' fields.
Test Data	- Input: ['from' => 'billing@company.com', 'to' => 'client@example.com', 'subject' => 'Invoice #12345', 'body' => 'Please find your invoice attached']
Expected Result	- mail->data matches input. - mail->data['from'] is 'billing@company.com'. - mail->data['subject'] is 'Invoice #12345'.
Actual Result	Property matched input and specific fields as expected.
Status	Pass
Severity	Medium

Test Case ID	SM-011
Title	Accessibility of public data property in SendInvoiceMail
Objective	Confirm accessibility and correct population of public data property.
Preconditions	- Application running - Crater\Mail\SendInvoiceMail class available
Test Steps	1. Instantiate SendInvoiceMail with ['invoice_id' => 123]. 2. Access mail->data. 3. Verify data is array, contains 'invoice_id', and value is 123.
Test Data	- Input: ['invoice_id' => 123]
Expected Result	- mail->data is array. - mail->data['invoice_id'] is 123.
Actual Result	Accessible and values stored as specified.
Status	Pass
Severity	Low

Test Case ID	SM-012
Title	SendInvoiceMail build method and EmailLog creation
Objective	Validate the existence of build method and ensure EmailLog is created referencing Invoice.
Preconditions	- Application running - Class and file source available - Crater\Mail\SendInvoiceMail class loaded
Test Steps	1. Use ReflectionClass to check 'build' method exists. 2. Read source file. 3. Confirm presence of 'EmailLog::create', 'Invoice::class', and 'route('invoice') code.
Test Data	- N/A (structural verification)

Expected Result	- 'build' method exists in SendInvoiceMail. - EmailLog creation and Invoice class reference confirmed. - Usage of invoice route present.
Actual Result	All requirements met in class file.
Status	Pass
Severity	High

Test Case ID	SM-013
Title	Instantiation of SendPaymentMail class
Objective	Confirm SendPaymentMail class can be instantiated with appropriate data.
Preconditions	- Application running - Crater\Mail\SendPaymentMail class available
Test Steps	1. Provide input: 'from' => 'test@example.com', 'to' => 'customer@example.com'. 2. Instantiate SendPaymentMail. 3. Verify instance is of SendPaymentMail class.
Test Data	- Input: ['from' => 'test@example.com', 'to' => 'customer@example.com']
Expected Result	- Object is instance of SendPaymentMail.
Actual Result	Instance verification successful.
Status	Pass
Severity	Medium

Test Case ID	SM-014
Title	SendPaymentMail inheritance from Mailable
Objective	Validate that SendPaymentMail extends Illuminate\Mail\Mailable.
Preconditions	- Application running - Crater\Mail\SendPaymentMail class available
Test Steps	1. Instantiate SendPaymentMail with empty data. 2. Check for instance of \Illuminate\Mail\Mailable.
Test Data	- Input: []
Expected Result	- SendPaymentMail object is instance of \Illuminate\Mail\Mailable.
Actual Result	Inheritance confirmed.
Status	Pass
Severity	Medium

Test Case ID	SM-015
Title	SendPaymentMail uses Queueable and SerializesModels traits
Objective	Verify usage of necessary traits for queuing and serialization.
Preconditions	- Application running - Crater\Mail\SendPaymentMail class available
Test Steps	1. Create ReflectionClass for SendPaymentMail. 2. Retrieve traits used in the class. 3. Check for presence of Illuminate\Bus\Queueable and Illuminate\Queue\SerializesModels traits.
Test Data	- N/A
Expected Result	- Both traits are present.
Actual Result	Traits confirmed in SendPaymentMail.
Status	Pass

Severity	Medium

Test Case ID	SM-016
Title	SendPaymentMail constructor sets data property
Objective	Ensure that constructor sets the data property correctly on creation.
Preconditions	- Application running - Crater\Mail\SendPaymentMail class available
Test Steps	1. Provide input: from, to, subject, body. 2. Instantiate SendPaymentMail with input. 3. Confirm mail->data matches input. 4. Specifically check 'from' and 'subject'.
Test Data	- Input: ['from' => 'payments@company.com', 'to' => 'customer@example.com', 'subject' => 'Payment Confirmation', 'body' => 'Thank you for your payment']
Expected Result	- mail->data matches input. - mail->data['from'] is 'payments@company.com'. - mail->data['subject'] is 'Payment Confirmation'.
Actual Result	Values matched on all tests.
Status	Pass
Severity	High

Test Case ID	SM-017
Title	Accessibility of public data property in SendPaymentMail
Objective	Confirm accessibility and proper assignment of array values in public data property.
Preconditions	- Application running - Crater\Mail\SendPaymentMail class available
Test Steps	1. Instantiate SendPaymentMail with ['payment_id' => 456, 'amount' => 1000]. 2. Access mail->data. 3. Confirm data is array, has keys 'payment_id' and 'amount' with corresponding values.
Test Data	- Input: ['payment_id' => 456, 'amount' => 1000]
Expected Result	- mail->data is array. - mail->data['payment_id'] is 456. - mail->data['amount'] is 1000.
Actual Result	All keys and values matched input.
Status	Pass
Severity	Medium

Test Case ID	SM-018
Title	SendPaymentMail build method and EmailLog creation
Objective	Validate existence of the build method and correct references for payment logging.
Preconditions	- Application running - Class and source code accessible - Crater\Mail\SendPaymentMail class loaded
Test Steps	1. Use ReflectionClass to verify existence of 'build' method. 2. Inspect source file for 'EmailLog::create', 'Payment::class', and 'route('payment') code. 3. Confirm all code components are present.

Test Data	- N/A (structural code examination)
Expected Result	- 'build' method exists. - 'EmailLog::create', 'Payment::class' references, and 'route('payment') usage found.
Actual Result	All verification points confirmed in class file.
Status	Pass
Severity	High

File: SettingKeyRequest-Test.txt

Test Case ID	SKR-001
Title	Authorize method always returns true
Objective	Verify that the authorize method of SettingKeyRequest always returns true, allowing all users to proceed.
Preconditions	- Application running - Crater\Http\Requests\SettingKeyRequest class available - Database seeded (if required by the system)
Test Steps	1. Instantiate a SettingKeyRequest object. 2. Invoke the authorize() method on the object. 3. Check the returned value.
Test Data	- SettingKeyRequest instance - No input data needed
Expected Result	- The authorize() method returns true.
Actual Result	The authorize() method returned true as expected.
Status	Pass
Severity	High

Test Case ID	SKR-002
Title	Rules method returns correct validation rules for 'key'
Objective	Verify that rules() method returns an array containing validation rules for the 'key' field, including 'required'.
Preconditions	- Application running - Crater\Http\Requests\SettingKeyRequest class available
Test Steps	1. Instantiate a SettingKeyRequest object. 2. Call the rules() method. 3. Verify the return value is an array. 4. Verify the 'key' element exists in the array. 5. Verify that the 'key' element includes 'required' in its rules.
Test Data	- SettingKeyRequest instance
Expected Result	- rules() returns an array. - Array contains a 'key' element. - 'key' element contains 'required'.
Actual Result	rules() returned an array with 'key' containing 'required' as expected.
Status	Pass
Severity	Medium

Test Case ID	SKR-003
Title	Validation passes when 'key' is provided
Objective	Verify that validation passes when a valid 'key' value is included in the data.
Preconditions	- Application running - Crater\Http\Requests\SettingKeyRequest class available
Test Steps	1. Instantiate a SettingKeyRequest object. 2. Prepare input data: ['key' => 'some_valid_key_string']. 3. Use Laravel Validator with rules from SettingKeyRequest and provided data. 4. Check if validation passes. 5. Check that there are no validation errors.
Test Data	- Input: ['key' => 'some_valid_key_string']

Expected Result	- Validation passes (returns true). - Errors array is empty.
Actual Result	Validation passed and errors array was empty as expected.
Status	Pass
Severity	High

Test Case ID	SKR-004
Title	Validation fails when 'key' is missing
Objective	Verify validation fails when the 'key' field is omitted from input data.
Preconditions	- Application running - Crater\Http\Requests\SettingKeyRequest class available
Test Steps	1. Instantiate a SettingKeyRequest object. 2. Prepare input data: [] (no 'key'). 3. Use Laravel Validator with rules from SettingKeyRequest. 4. Check if validation fails. 5. Verify errors have 'key'. 6. Check that the first error message for 'key' is 'The key field is required.'
Test Data	- Input: []
Expected Result	- Validation fails (returns true for fails()). - Errors include 'key'. - First error message for 'key' is 'The key field is required.'
Actual Result	Validation failed, errors included 'key', and error message matched as expected.
Status	Pass
Severity	High

Test Case ID	SKR-005
Title	Validation fails when 'key' is an empty string
Objective	Verify validation fails when 'key' is provided as an empty string.
Preconditions	- Application running - Crater\Http\Requests\SettingKeyRequest class available
Test Steps	1. Instantiate a SettingKeyRequest object. 2. Prepare input data: ['key' => '']. 3. Use Laravel Validator with rules from SettingKeyRequest. 4. Check if validation fails. 5. Verify errors have 'key'. 6. Check that the first error message for 'key' is 'The key field is required.'
Test Data	- Input: ['key' => '']
Expected Result	- Validation fails (returns true for fails()). - Errors include 'key'. - First error message for 'key' is 'The key field is required.'
Actual Result	Validation failed, errors included 'key', and error message was correct.
Status	Pass
Severity	High

Test Case ID	SKR-006
Title	Validation fails when 'key' is null
Objective	Verify validation fails when 'key' is set to null in input data.
Preconditions	- Application running - Crater\Http\Requests\SettingKeyRequest class available

Test Steps	1. Instantiate a SettingKeyRequest object. 2. Prepare input data: ['key' => null]. 3. Use Laravel Validator with rules from SettingKeyRequest. 4. Check if validation fails. 5. Verify errors have 'key'. 6. Check that the first error message for 'key' is 'The key field is required.'
Test Data	- Input: ['key' => null]
Expected Result	- Validation fails (returns true for fails()). - Errors include 'key'. - First error message for 'key' is 'The key field is required.'
Actual Result	Validation failed, errors included 'key', and error message matched as expected.
Status	Pass
Severity	High

File: SettingRequest-Test.txt

Test Case ID	SR-001
Title	Authorize Method Returns True
Objective	Verify that the authorize() method in SettingRequest always returns true, ensuring any user can access this functionality.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\SettingRequest class is available
Test Steps	1. Instantiate a new SettingRequest object. 2. Call the authorize() method on the SettingRequest object. 3. Check if the result is true.
Test Data	- SettingRequest instance (no additional input required) - Expected value: true
Expected Result	authorize() method returns true.
Actual Result	authorize() method returned true.
Status	Pass
Severity	High

Test Case ID	SR-002
Title	Rules Method Returns Correct Validation Rules
Objective	Verify that the rules() method in SettingRequest returns the correct validation rules array.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\SettingRequest class is available
Test Steps	1. Instantiate a new SettingRequest object. 2. Call the rules() method on the SettingRequest object. 3. Compare the result to the expected rules array: - 'settings' => ['required']
Test Data	- SettingRequest instance (no additional input required) - Expected value: ['settings' => ['required']]
Expected Result	rules() method returns: ['settings' => ['required',],]
Actual Result	rules() method returned the expected array.
Status	Pass
Severity	Medium

File: SettingsPolicy-Test.txt

Test Case ID	SP-001
Title	Verify manageCompany returns true if user is the company owner
Objective	To ensure that the manageCompany method authorizes the user as company owner.
Preconditions	- Application running - Database seeded - User and Company models available - SettingsPolicy is properly instantiated
Test Steps	1. Mock a User object with id = 1. 2. Mock a Company object with owner_id = 1. 3. Call manageCompany($user, $company) method of SettingsPolicy. 4. Assert that the result is true.
Test Data	- user id: 1 - company owner_id: 1
Expected Result	- manageCompany returns true
Actual Result	manageCompany returned true
Status	Pass
Severity	High

Test Case ID	SP-002
Title	Verify manageCompany returns false if user is not the company owner
Objective	To confirm that the manageCompany method denies authorization when user is not owner.
Preconditions	- Application running - Database seeded - User and Company models available - SettingsPolicy is properly instantiated
Test Steps	1. Mock a User object with id = 1. 2. Mock a Company object with owner_id = 2. 3. Call manageCompany($user, $company) method of SettingsPolicy. 4. Assert that the result is false.
Test Data	- user id: 1 - company owner_id: 2
Expected Result	- manageCompany returns false
Actual Result	manageCompany returned false
Status	Pass
Severity	High

Test Case ID	SP-003
Title	Verify manageBackups returns true if user is an owner
Objective	To validate that manageBackups authorizes owners correctly.
Preconditions	- Application running - Database seeded - User model available - SettingsPolicy instantiated
Test Steps	1. Mock a User object. 2. Set isOwner() to return true. 3. Call manageBackups($user) method of SettingsPolicy. 4. Assert that the result is true.
Test Data	- user->isOwner(): true

Expected Result	- manageBackups returns true
Actual Result	manageBackups returned true
Status	Pass
Severity	High

Test Case ID	SP-004
Title	Verify manageBackups returns false if user is not an owner
Objective	To check that manageBackups denies access when user is not owner.
Preconditions	- Application running - Database seeded - User model available - SettingsPolicy instantiated
Test Steps	1. Mock a User object. 2. Set isOwner() to return false. 3. Call manageBackups($user) method of SettingsPolicy. 4. Assert that the result is false.
Test Data	- user->isOwner(): false
Expected Result	- manageBackups returns false
Actual Result	manageBackups returned false
Status	Pass
Severity	High

Test Case ID	SP-005
Title	Verify manageFileDisk returns true if user is an owner
Objective	To ensure manageFileDisk allows owner access.
Preconditions	- Application running - Database seeded - User model available - SettingsPolicy instantiated
Test Steps	1. Mock a User object. 2. Set isOwner() to return true. 3. Call manageFileDisk($user) method of SettingsPolicy. 4. Assert that the result is true.
Test Data	- user->isOwner(): true
Expected Result	- manageFileDisk returns true
Actual Result	manageFileDisk returned true
Status	Pass
Severity	High

Test Case ID	SP-006
Title	Verify manageFileDisk returns false if user is not an owner
Objective	To confirm manageFileDisk denies non-owner access.
Preconditions	- Application running - Database seeded - User model available - SettingsPolicy instantiated
Test Steps	1. Mock a User object. 2. Set isOwner() to return false. 3. Call manageFileDisk($user) method of SettingsPolicy. 4. Assert that the result is false.

Test Data	- user->isOwner(): false
Expected Result	- manageFileDisk returns false
Actual Result	manageFileDisk returned false
Status	Pass
Severity	High

Test Case ID	SP-007
Title	Verify manageEmailConfig returns true if user is an owner
Objective	To verify manageEmailConfig allows email configuration for owner.
Preconditions	- Application running - Database seeded - User model available - SettingsPolicy instantiated
Test Steps	1. Mock a User object. 2. Set isOwner() to return true. 3. Call manageEmailConfig($user) method of SettingsPolicy. 4. Assert that the result is true.
Test Data	- user->isOwner(): true
Expected Result	- manageEmailConfig returns true
Actual Result	manageEmailConfig returned true
Status	Pass
Severity	High

Test Case ID	SP-008
Title	Verify manageEmailConfig returns false if user is not an owner
Objective	To ensure manageEmailConfig denies configuration for non-owner.
Preconditions	- Application running - Database seeded - User model available - SettingsPolicy instantiated
Test Steps	1. Mock a User object. 2. Set isOwner() to return false. 3. Call manageEmailConfig($user) method of SettingsPolicy. 4. Assert that the result is false.
Test Data	- user->isOwner(): false
Expected Result	- manageEmailConfig returns false
Actual Result	manageEmailConfig returned false
Status	Pass
Severity	High

Test Case ID	SP-009
Title	Verify manageSettings returns true if user is an owner
Objective	To check that manageSettings authorizes owners to manage settings.
Preconditions	- Application running - Database seeded - User model available - SettingsPolicy instantiated

Test Steps	1. Mock a User object. 2. Set isOwner() to return true. 3. Call manageSettings($user) method of SettingsPolicy. 4. Assert that the result is true.
Test Data	- user->isOwner(): true
Expected Result	- manageSettings returns true
Actual Result	manageSettings returned true
Status	Pass
Severity	High

Test Case ID	SP-010
Title	Verify manageSettings returns false if user is not an owner
Objective	To ensure manageSettings rejects non-owner for managing settings.
Preconditions	- Application running - Database seeded - User model available - SettingsPolicy instantiated
Test Steps	1. Mock a User object. 2. Set isOwner() to return false. 3. Call manageSettings($user) method of SettingsPolicy. 4. Assert that the result is false.
Test Data	- user->isOwner(): false
Expected Result	- manageSettings returns false
Actual Result	manageSettings returned false
Status	Pass
Severity	High

File: ShowReceiptController-Test.txt

Test Case ID	SRC-001
Title	Instantiate ShowReceiptController
Objective	Verify that ShowReceiptController can be instantiated successfully.
Preconditions	- Application running - Required classes loaded - No constructor dependencies
Test Steps	1. Attempt to create a new instance of ShowReceiptController. 2. Check if the created instance is of type ShowReceiptController.
Test Data	- Class: ShowReceiptController - Instantiation: new ShowReceiptController()
Expected Result	- The object is an instance of ShowReceiptController.
Actual Result	An instance of ShowReceiptController is created successfully.
Status	Pass
Severity	Low

Test Case ID	SRC-002
Title	ShowReceiptController Extends Controller
Objective	Ensure ShowReceiptController inherits from base Controller class.
Preconditions	- Application running - Required class hierarchy defined
Test Steps	1. Instantiate ShowReceiptController. 2. Verify that the instance is also of type Controller.
Test Data	- Class: ShowReceiptController - Base class: Controller
Expected Result	- ShowReceiptController instance is an instance of Controller.
Actual Result	ShowReceiptController correctly extends Controller.
Status	Pass
Severity	Medium

Test Case ID	SRC-003
Title	ShowReceiptController Namespace Verification
Objective	Confirm that ShowReceiptController is declared in the correct namespace.
Preconditions	- Application running - Codebase loaded
Test Steps	1. Use ReflectionClass on ShowReceiptController. 2. Retrieve the namespace name. 3. Verify that the namespace matches 'Crater\Http\Controllers\V1\Admin\Expense'.
Test Data	- Expected namespace: 'Crater\Http\Controllers\V1\Admin\Expense'
Expected Result	- ShowReceiptController is in the correct namespace.
Actual Result	Namespace is correctly set to 'Crater\Http\Controllers\V1\Admin\Expense'.
Status	Pass
Severity	Low

Test Case ID	SRC-004
Title	ShowReceiptController Invokable Method Existence

Objective	Verify that ShowReceiptController has the __invoke method.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Use ReflectionClass on ShowReceiptController. 2. Check for the existence of the __invoke method.
Test Data	- Class: ShowReceiptController - Method: __invoke
Expected Result	- ShowReceiptController has a method named __invoke.
Actual Result	__invoke method exists in ShowReceiptController.
Status	Pass
Severity	Medium

Test Case ID	SRC-005
Title	ShowReceiptController __invoke Method Accessibility
Objective	Validate that __invoke method is public and not static.
Preconditions	- Application running - Controller class loaded
Test Steps	1. Use ReflectionClass on ShowReceiptController. 2. Retrieve the __invoke method. 3. Check if the method is public. 4. Check if the method is not static.
Test Data	- Method: __invoke
Expected Result	- __invoke method is public. - __invoke method is not static.
Actual Result	__invoke method is public and not static.
Status	Pass
Severity	Medium

Test Case ID	SRC-006
Title	ShowReceiptController __invoke Accepts Expense Parameter
Objective	Ensure __invoke method accepts a single Expense-typed parameter.
Preconditions	- Application running - Expense model available
Test Steps	1. Use ReflectionClass on ShowReceiptController. 2. Inspect the __invoke method parameters. 3. Check that it has exactly one parameter named 'expense' and type contains 'Expense'.
Test Data	- Parameter: $expense - Type: Expense model
Expected Result	- __invoke method has one parameter: $expense of type Expense.
Actual Result	__invoke method correctly accepts a single Expense parameter.
Status	Pass
Severity	High

Test Case ID	SRC-007
Title	ShowReceiptController Uses Authorization
Objective	Confirm that authorization is enforced in ShowReceiptController when viewing expenses.

Preconditions	- Application running - Authorization policy available
Test Steps	1. Use ReflectionClass to read the controller file content. 2. Search for invocation of $this->authorize('view', $expense).
Test Data	- Authorization call: $this->authorize('view', $expense)
Expected Result	- Controller contains the authorization check for viewing the expense.
Actual Result	Authorization is correctly enforced for view permission.
Status	Pass
Severity	High

Test Case ID	SRC-008
Title	Expense Existence Check in ShowReceiptController
Objective	Verify that the controller checks for the existence of the expense before proceeding.
Preconditions	- Application running - Expense record may exist or not
Test Steps	1. Use ReflectionClass to read the controller file content. 2. Check for the existence of the 'if ($expense)' statement.
Test Data	- Expense variable: $expense
Expected Result	- Controller checks if the expense exists before processing.
Actual Result	Expense existence is checked as expected.
Status	Pass
Severity	High

Test Case ID	SRC-009
Title	Retrieve Expense Receipt Media in ShowReceiptController
Objective	Ensure the controller retrieves media from the receipts collection.
Preconditions	- Application running - Expense record with attached receipt media
Test Steps	1. Use ReflectionClass to read the controller file content. 2. Verify the presence of $expense->getFirstMedia('receipts'). 3. Check for existence check on media object.
Test Data	- Expense media collection: 'receipts'
Expected Result	- Controller retrieves first media from 'receipts' collection if available.
Actual Result	Media from 'receipts' collection is properly retrieved and checked.
Status	Pass
Severity	Medium

Test Case ID	SRC-010
Title	ShowReceiptController Returns File or Error Response
Objective	Confirm the controller returns file response if receipt exists or an error if not.
Preconditions	- Application running - Expense record present (with or without receipt media)
Test Steps	1. Use ReflectionClass to read the controller file content. 2. Verify response()->file($media->getPath()) is used to return receipt file. 3. Verify respondJson('receipt_does_not_exist', 'Receipt does not exist.') is used for errors.
Test Data	- Media: $media (from receipts collection) - Error response: 'receipt_does_not_exist'

Expected Result	- If receipt exists, file response is returned. - If receipt does not exist, error JSON response is returned.
Actual Result	Controller returns file response on receipt found, otherwise returns error JSON as expected.
Status	Pass
Severity	High

File: SiteApi-Test.txt

Test Case ID	SA-001
Title	Verify SiteApi is implemented as a trait
Objective	Confirm that the SiteApi class is a PHP trait.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class is loaded
Test Steps	1. Use PHP ReflectionClass to inspect SiteApi::class. 2. Check if SiteApi is a trait by using isTrait().
Test Data	- Input: SiteApi::class - Expected: isTrait() returns true
Expected Result	- SiteApi is verified as a trait.
Actual Result	SiteApi is verified as a trait.
Status	Pass
Severity	Medium

Test Case ID	SA-002
Title	Confirm SiteApi class namespace is Crater\Space
Objective	Ensure SiteApi is defined within the correct namespace.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class is loaded
Test Steps	1. Use PHP ReflectionClass to inspect SiteApi::class. 2. Verify the namespace using getNamespaceName().
Test Data	- Input: SiteApi::class - Expected: getNamespaceName() returns 'Crater\Space'
Expected Result	- SiteApi is in the 'Crater\Space' namespace.
Actual Result	SiteApi is in the 'Crater\Space' namespace.
Status	Pass
Severity	Low

Test Case ID	SA-003
Title	Check the existence of getRemote method in SiteApi
Objective	Verify that SiteApi defines the getRemote method.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class is loaded
Test Steps	1. Use PHP ReflectionClass to inspect SiteApi::class. 2. Check for the presence of the getRemote method using hasMethod().
Test Data	- Input: SiteApi::class, method name 'getRemote' - Expected: hasMethod('getRemote') returns true
Expected Result	- SiteApi has the getRemote method.
Actual Result	SiteApi has the getRemote method.
Status	Pass
Severity	High

Test Case ID	SA-004

Title	Verify getRemote method access is protected and static
Objective	Ensure the getRemote method in SiteApi is both protected and static.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class is loaded
Test Steps	1. Use PHP ReflectionClass to inspect SiteApi::class. 2. Retrieve the getRemote method using getMethod(). 3. Verify that the method is protected via isProtected(). 4. Verify that the method is static via isStatic().
Test Data	- Input: SiteApi::class, method name 'getRemote' - Expected: isProtected() returns true - Expected: isStatic() returns true
Expected Result	- getRemote method is protected. - getRemote method is static.
Actual Result	getRemote method is protected and static.
Status	Pass
Severity	High

Test Case ID	SA-005
Title	Validate getRemote method parameters: url, data, token
Objective	Check that getRemote method accepts three parameters: url, data, and token.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class is loaded
Test Steps	1. Use PHP ReflectionClass to inspect SiteApi::class. 2. Retrieve getRemote method using getMethod(). 3. Extract parameter list using getParameters(). 4. Verify the parameter count is 3. 5. Ensure the parameter names are 'url', 'data', and 'token' respectively.
Test Data	- Input: SiteApi::class, method name 'getRemote' - Expected: Parameter count is 3 - Expected: Parameter names are 'url', 'data', 'token'
Expected Result	- getRemote method accepts exactly three parameters: url, data, and token.
Actual Result	getRemote method accepts url, data, and token parameters.
Status	Pass
Severity	High

Test Case ID	SA-006
Title	Ensure Guzzle Client is created with base_uri in getRemote method
Objective	Confirm that getRemote method instantiates a Guzzle Client with the correct base_uri setting.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class file accessible - Configuration setting for 'crater.base_url' present
Test Steps	1. Obtain file contents of SiteApi class via getFileName(). 2. Search for code containing 'new Client(['. 3. Confirm 'verify' => false is included. 4. Confirm 'base_uri' => config('crater.base_url') is included.
Test Data	- Input: File contents of SiteApi - Expected: Presence of 'new Client([' - Expected: Presence of 'verify' => false - Expected: Presence of 'base_uri' => config('crater.base_url')

Expected Result	- getRemote method creates a Guzzle Client with base_uri set to config('crater.base_url') and 'verify' => false.
Actual Result	getRemote method creates a Guzzle Client with base_uri and disables SSL verification.
Status	Pass
Severity	High

Test Case ID	SA-007
Title	Validate headers set in getRemote method
Objective	Ensure getRemote sets required HTTP headers.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class file accessible
Test Steps	1. Obtain file contents of SiteApi class. 2. Check for '$headers['headers'] = [' line. 3. Confirm presence of header 'Accept' => 'application/json'. 4. Confirm presence of header 'Referer' => url('/'). 5. Confirm presence of header 'crater' => Setting::getSetting('version'). 6. Confirm presence of header 'Authorization' => "Bearer {$token}".
Test Data	- Input: File contents of SiteApi - Expected: Appropriate header definitions
Expected Result	- getRemote method sets headers: Accept, Referer, crater (version), Authorization (Bearer token).
Actual Result	getRemote method sets Accept, Referer, crater version, and Bearer Authorization headers.
Status	Pass
Severity	High

Test Case ID	SA-008
Title	Verify http_errors disabled in getRemote
Objective	Confirm that getRemote disables http_errors in Guzzle request options.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class file accessible
Test Steps	1. Obtain file contents of SiteApi class. 2. Search for code that sets '$data['http_errors'] = false'.
Test Data	- Input: File contents of SiteApi - Expected: Presence of '$data['http_errors'] = false'
Expected Result	- getRemote method disables http_errors in request options.
Actual Result	getRemote disables http_errors for Guzzle requests.
Status	Pass
Severity	High

Test Case ID	SA-009
Title	Validate merging of data with headers in getRemote
Objective	Ensure getRemote merges user data and headers before making the request.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class file accessible

Test Steps	1. Obtain file contents of SiteApi class. 2. Search for code 'array_merge($data, $headers)'. 3. Confirm $data variable is reassigned with merged result.
Test Data	- Input: File contents of SiteApi - Expected: Code: '$data = array_merge($data, $headers)'
Expected Result	- getRemote merges provided data array and headers into request options.
Actual Result	getRemote correctly merges data and headers before performing the request.
Status	Pass
Severity	High

Test Case ID	SA-010
Title	Ensure getRemote handles RequestException
Objective	Confirm getRemote method catches Guzzle RequestException and returns the exception object.
Preconditions	- Application running - PHP environment with Pest test framework installed - SiteApi class file accessible
Test Steps	1. Obtain file contents of SiteApi class. 2. Confirm presence of try/catch block around '$client->get($url, $data)'. 3. Check for catch(RequestException $e) block. 4. Ensure $result is set to $e in catch block. 5. Confirm the result is returned.
Test Data	- Input: File contents of SiteApi - Expected: Exception handling with proper object return
Expected Result	- getRemote catches Guzzle RequestException and returns the exception object.
Actual Result	getRemote handles RequestException and returns it as result.
Status	Pass
Severity	High

File: TaxCollectionAndResource-Test.txt

Test Case ID	TCAR-001
Title	Instantiation of TaxCollection
Objective	Verify that a TaxCollection object can be instantiated successfully.
Preconditions	- Application running - Required classes (TaxCollection, Collection) are available - No tax data required for instantiation
Test Steps	1. Create a new empty Collection object. 2. Instantiate a TaxCollection object using the empty Collection. 3. Verify the TaxCollection object is created.
Test Data	- Input: Empty Collection instance - Expected value: Object instantiated is an instance of TaxCollection
Expected Result	- TaxCollection object is successfully instantiated and is an instance of TaxCollection class.
Actual Result	TaxCollection object is successfully instantiated and confirmed as instance of TaxCollection class.
Status	Pass
Severity	Medium

Test Case ID	TCAR-002
Title	TaxCollection Inheritance from ResourceCollection
Objective	Verify that TaxCollection extends the base ResourceCollection class.
Preconditions	- Application running - Required classes (TaxCollection, ResourceCollection, Collection) available
Test Steps	1. Create a new empty Collection object. 2. Instantiate a TaxCollection object with the Collection. 3. Verify TaxCollection is an instance of ResourceCollection.
Test Data	- Input: Empty Collection instance - Expected value: TaxCollection instance is also an instance of ResourceCollection
Expected Result	- TaxCollection object is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	TaxCollection object is confirmed as a valid instance of ResourceCollection.
Status	Pass
Severity	Medium

Test Case ID	TCAR-003
Title	TaxCollection Namespace Verification
Objective	Ensure the TaxCollection class resides in the correct namespace.
Preconditions	- Application running - TaxCollection class available
Test Steps	1. Reflect TaxCollection class using ReflectionClass. 2. Retrieve the namespace of TaxCollection. 3. Verify the namespace matches 'Crater\Http\Resources'.
Test Data	- Input: TaxCollection class - Expected value: Namespace is 'Crater\Http\Resources'
Expected Result	- The namespace of TaxCollection is 'Crater\Http\Resources'.
Actual Result	Reflection confirms TaxCollection is in 'Crater\Http\Resources'.
Status	Pass

Severity	Low

Test Case ID	TCAR-004
Title	TaxCollection toArray with Empty Collection
Objective	Verify that toArray returns an empty array for an empty TaxCollection.
Preconditions	- Application running - TaxCollection class available
Test Steps	1. Create an empty Collection instance. 2. Instantiate TaxCollection with the empty Collection. 3. Create an empty Request object. 4. Call the toArray method on TaxCollection with Request. 5. Verify the result is an empty array.
Test Data	- Input: Empty Collection, empty Request - Expected value: Returned array is empty
Expected Result	- Calling toArray on TaxCollection with empty data returns an empty array.
Actual Result	Calling toArray with empty collection returns an empty array as expected.
Status	Pass
Severity	Medium

Test Case ID	TCAR-005
Title	TaxCollection toArray Accepts Request Parameter
Objective	Ensure the toArray method accepts and processes a Request parameter.
Preconditions	- Application running - TaxCollection class available
Test Steps	1. Create an empty Collection instance. 2. Instantiate TaxCollection with the Collection. 3. Create a Request object containing data (e.g., ['test' => 'value']). 4. Call toArray on TaxCollection with the Request. 5. Verify the returned value is an array.
Test Data	- Input: Request(['test' => 'value']) - Expected value: Output type is array
Expected Result	- Output of toArray is always an array when provided a Request.
Actual Result	toArray returns an array regardless of Request parameters.
Status	Pass
Severity	Medium

Test Case ID	TCAR-006
Title	TaxCollection toArray Delegates to Parent
Objective	Verify that TaxCollection’s toArray method delegates to parent implementation.
Preconditions	- Application running - TaxCollection class available
Test Steps	1. Create an empty Collection instance. 2. Instantiate TaxCollection with the Collection. 3. Create a Request object. 4. Call toArray on TaxCollection with the Request. 5. Verify the output is of type array (as per parent class).
Test Data	- Input: Empty Collection, empty Request - Expected value: Output is an array as per ResourceCollection behavior
Expected Result	- Calling toArray returns an array, consistent with parent functionality.
Actual Result	toArray returns output as array, implemented through parent class delegation.

Status	Pass
Severity	Medium

Test Case ID	TCAR-007
Title	Instantiation of TaxResource
Objective	Verify that a TaxResource object can be instantiated with data.
Preconditions	- Application running - TaxResource class available
Test Steps	1. Create a data object with attributes: id=1, name='VAT'. 2. Instantiate a TaxResource object using the data object. 3. Verify the TaxResource object is created.
Test Data	- Input: (object)['id' => 1, 'name' => 'VAT'] - Expected value: Object instantiated is an instance of TaxResource
Expected Result	- TaxResource object is instantiated successfully as an instance of TaxResource.
Actual Result	TaxResource object is instantiated and confirmed as TaxResource class instance.
Status	Pass
Severity	Medium

Test Case ID	TCAR-008
Title	TaxResource Inheritance from JsonResource
Objective	Ensure TaxResource extends the base JsonResource class.
Preconditions	- Application running - TaxResource class available
Test Steps	1. Create an empty object for data. 2. Instantiate a TaxResource object using the empty object. 3. Verify TaxResource is an instance of JsonResource.
Test Data	- Input: (object)[] - Expected value: TaxResource instance is also an instance of JsonResource
Expected Result	- TaxResource object is an instance of Illuminate\Http\Resources\Json\JsonResource.
Actual Result	TaxResource is confirmed as an instance of JsonResource.
Status	Pass
Severity	Medium

Test Case ID	TCAR-009
Title	TaxResource Namespace Verification
Objective	Confirm that the TaxResource class is defined in the correct namespace.
Preconditions	- Application running - TaxResource class available
Test Steps	1. Reflect TaxResource class using ReflectionClass. 2. Retrieve the namespace of TaxResource. 3. Verify the namespace matches 'Crater\Http\Resources'.
Test Data	- Input: TaxResource class - Expected value: Namespace is 'Crater\Http\Resources'
Expected Result	- TaxResource class is defined in namespace 'Crater\Http\Resources'.
Actual Result	Reflection confirms TaxResource is in 'Crater\Http\Resources'.
Status	Pass

Severity	Low

Test Case ID	TCAR-010
Title	TaxResource toArray Includes Basic Tax Fields
Objective	Verify toArray of TaxResource contains all essential tax field mappings.
Preconditions	- Application running - TaxResource class available
Test Steps	1. Reflect TaxResource class using ReflectionClass. 2. Retrieve and read the file content of the class. 3. Check for presence of fields: id, tax_type_id, invoice_id, estimate_id, name, amount, percent in toArray.
Test Data	- Input: TaxResource class file - Expected values: toArray contains field assignments for id, tax_type_id, invoice_id, estimate_id, name, amount, percent
Expected Result	- toArray of TaxResource includes: id, tax_type_id, invoice_id, estimate_id, name, amount, percent.
Actual Result	All basic tax fields are found in toArray as expected.
Status	Pass
Severity	High

Test Case ID	TCAR-011
Title	TaxResource toArray Includes Item and Company IDs
Objective	Ensure toArray of TaxResource includes fields for item and company relationships.
Preconditions	- Application running - TaxResource class available
Test Steps	1. Reflect TaxResource class. 2. Read the file to inspect fields within toArray. 3. Confirm fields: invoice_item_id, estimate_item_id, item_id, company_id are present.
Test Data	- Input: TaxResource class file - Expected values: Fields invoice_item_id, estimate_item_id, item_id, company_id present
Expected Result	- toArray of TaxResource includes invoice_item_id, estimate_item_id, item_id, and company_id.
Actual Result	All item and company ID fields are present in the toArray output.
Status	Pass
Severity	High

Test Case ID	TCAR-012
Title	TaxResource toArray Includes compound_tax and base_amount Fields
Objective	Ensure toArray of TaxResource contains compound_tax and base_amount fields.
Preconditions	- Application running - TaxResource class available
Test Steps	1. Reflect TaxResource class. 2. Inspect toArray contents for 'compound_tax' and 'base_amount' fields.
Test Data	- Input: TaxResource class file - Expected values: compound_tax and base_amount fields present
Expected Result	- toArray includes fields compound_tax and base_amount.

Actual Result	compound_tax and base_amount fields are both included in the toArray output.
Status	Pass
Severity	High

Test Case ID	TCAR-013
Title	TaxResource toArray Includes currency_id and recurring_invoice_id
Objective	Verify that toArray includes currency_id and recurring_invoice_id fields.
Preconditions	- Application running - TaxResource class available
Test Steps	1. Reflect TaxResource class. 2. Check that toArray contains: currency_id and recurring_invoice_id.
Test Data	- Input: TaxResource class file - Expected values: currency_id and recurring_invoice_id fields present
Expected Result	- toArray includes fields currency_id and recurring_invoice_id.
Actual Result	Both fields are present in toArray as expected.
Status	Pass
Severity	Medium

Test Case ID	TCAR-014
Title	TaxResource toArray Includes type from taxType Relationship
Objective	Ensure toArray returns 'type' extracted from related taxType entity.
Preconditions	- Application running - TaxResource and related TaxType relationship present
Test Steps	1. Reflect TaxResource class. 2. Inspect toArray for field: 'type' => $this->taxType->type.
Test Data	- Input: TaxResource class file - Expected values: 'type' mapped to taxType->type
Expected Result	- toArray provides 'type' from the related taxType property.
Actual Result	'type' field included and references taxType->type.
Status	Pass
Severity	Medium

Test Case ID	TCAR-015
Title	TaxResource toArray Includes Conditional tax_type Relationship
Objective	Ensure toArray conditionally includes tax_type relationship via TaxTypeResource.
Preconditions	- Application running - TaxResource and related TaxTypeResource available
Test Steps	1. Reflect TaxResource class. 2. Inspect toArray for conditional: 'tax_type' => $this->when($this->taxType()->exists(), ...).
Test Data	- Input: TaxResource class file - Expected value: Presence of conditional logic for tax_type using new TaxTypeResource()
Expected Result	- toArray includes conditional inclusion of tax_type if taxType exists.
Actual Result	Conditional inclusion of tax_type with TaxTypeResource is implemented as expected.
Status	Pass

Severity	High

Test Case ID	TCAR-016
Title	TaxResource toArray Includes Conditional currency Relationship
Objective	Confirm toArray includes currency relationship conditionally using CurrencyResource.
Preconditions	- Application running - TaxResource and related CurrencyResource available
Test Steps	1. Reflect TaxResource class. 2. Inspect toArray for conditional: 'currency' => $this->when($this->currency()->exists(), ...).
Test Data	- Input: TaxResource class file - Expected value: Conditional inclusion logic for currency using new CurrencyResource()
Expected Result	- toArray includes currency only when currency() relationship exists.
Actual Result	Conditional currency inclusion with CurrencyResource is present in toArray definition.
Status	Pass
Severity	Medium

File: TestMail-Test.txt

Test Case ID	TM-001
Title	TestMail instantiation and property assignment
Objective	Verify that a TestMail instance can be created and its subject and message properties are set to provided values.
Preconditions	- Application running - Database seeded - TestMail class available
Test Steps	1. Define $subject as 'Test Subject Line' 2. Define $message as 'This is the body of the test message.' 3. Instantiate TestMail with $subject and $message. 4. Retrieve the subject property from the instance. 5. Retrieve the message property from the instance.
Test Data	- Input: $subject = 'Test Subject Line', $message = 'This is the body of the test message.' - Expected values: subject property equals 'Test Subject Line', message property equals 'This is the body of the test message.'
Expected Result	- subject property of TestMail instance should be 'Test Subject Line' - message property of TestMail instance should be 'This is the body of the test message.'
Actual Result	- subject property correctly set to 'Test Subject Line' - message property correctly set to 'This is the body of the test message.'
Status	Pass
Severity	Medium

Test Case ID	TM-002
Title	TestMail property assignment with empty strings
Objective	Verify that TestMail can accept and assign empty strings to its subject and message properties.
Preconditions	- Application running - Database seeded - TestMail class available
Test Steps	1. Define $subject as '' 2. Define $message as '' 3. Instantiate TestMail with $subject and $message. 4. Retrieve the subject property from the instance. 5. Retrieve the message property from the instance.
Test Data	- Input: $subject = '', $message = '' - Expected values: subject property equals '', message property equals ''
Expected Result	- subject property of TestMail instance should be '' - message property of TestMail instance should be ''
Actual Result	- subject property correctly set to '' - message property correctly set to ''
Status	Pass
Severity	Medium

Test Case ID	TM-003
Title	TestMail property assignment with long strings
Objective	Verify that TestMail can handle and correctly assign long strings to its subject and message properties.

Preconditions	- Application running - Database seeded - TestMail class available
Test Steps	1. Define $subject as a string with 255 'A' characters. 2. Define $message as a string with 4096 'B' characters. 3. Instantiate TestMail with $subject and $message. 4. Retrieve the subject property from the instance. 5. Retrieve the message property from the instance.
Test Data	- Input: $subject = str_repeat('A', 255), $message = str_repeat('B', 4096) - Expected values: subject property equals 255 'A's, message property equals 4096 'B's
Expected Result	- subject property of TestMail instance should be 255 'A's - message property of TestMail instance should be 4096 'B's
Actual Result	- subject property correctly set to maximum test string - message property correctly set to maximum test string
Status	Pass
Severity	Medium

Test Case ID	TM-004
Title	TestMail build method configures mailable with valid inputs
Objective	Verify that the build method correctly calls subject, markdown, and with using provided non-empty subject and message.
Preconditions	- Application running - Database seeded - TestMail class available - Mockery available for mocking
Test Steps	1. Define $mailSubject as 'Configured Subject' 2. Define $mailMessage as 'Content for the email body.' 3. Create a partial mock of TestMail allowing subject, markdown, and with to be mocked. 4. Expect subject() to be called once with $mailSubject. 5. Expect markdown() to be called once with view name 'emails.test'. 6. Expect with() to be called once with data ['my_message' => $mailMessage]. 7. Call build() method on the mock instance. 8. Verify build() returns the mock instance itself.
Test Data	- Input: $mailSubject = 'Configured Subject', $mailMessage = 'Content for the email body.' - Expected calls: - subject('Configured Subject') - markdown('emails.test') - with(['my_message' => 'Content for the email body.'])
Expected Result	- subject() called exactly once with correct subject - markdown() called exactly once with view 'emails.test' - with() called exactly once with correct data - build() returns the mock instance itself
Actual Result	- All methods called as expected and build() returns the mock instance
Status	Pass
Severity	High

Test Case ID	TM-005
Title	TestMail build method works with empty subject and message
Objective	Verify that the build method correctly configures mailable even when subject and message are empty.

Preconditions	- Application running - Database seeded - TestMail class available - Mockery available
Test Steps	1. Define $mailSubject as '' 2. Define $mailMessage as '' 3. Create a partial mock of TestMail with subject, markdown, and with methods mocked. 4. Expect subject() to be called once with empty string. 5. Expect markdown() to be called once with view name 'emails.test'. 6. Expect with() to be called once with data ['my_message' => '']. 7. Call build() method on the mock instance. 8. Verify build() returns the mock instance itself.
Test Data	- Input: $mailSubject = '', $mailMessage = '' - Expected calls: - subject('') - markdown('emails.test') - with(['my_message' => ''])
Expected Result	- subject() called exactly once with '' - markdown() called exactly once with 'emails.test' - with() called exactly once with ['my_message' => ''] - build() returns the mock instance itself
Actual Result	- Methods called as expected and build() returns mock instance with empty properties
Status	Pass
Severity	High

Test Case ID	TM-006
Title	TestMail build method uses correct view name regardless of input
Objective	Verify that the build method always uses 'emails.test' as the view name, regardless of subject or message content.
Preconditions	- Application running - Database seeded - TestMail class available - Mockery available
Test Steps	1. Define $mailSubject as 'Any Subject' 2. Define $mailMessage as 'Any Message' 3. Create a partial mock of TestMail with subject, markdown, and with methods mocked. 4. Call markdown() and expect it to be called once with 'emails.test'. 5. Call build() method on the mock instance. 6. Verify build() returns the mock instance itself.
Test Data	- Input: $mailSubject = 'Any Subject', $mailMessage = 'Any Message' - Expected call: markdown('emails.test')
Expected Result	- markdown() called exactly once with 'emails.test' regardless of input data - build() returns the mock instance itself
Actual Result	- markdown() called once with correct view name and build() returns mock instance
Status	Pass
Severity	High

File: TimeZones-Test.txt

Test Case ID	TZ-001
Title	Retrieve complete and correct timezone list via get_list()
Objective	Verify that the get_list() function in the Crater\Space\TimeZones class returns an array containing a complete and accurate list of timezones with specified values and keys.
Preconditions	- Application is running - All dependencies for Crater\Space\TimeZones class are installed and configured - No mutations in underlying timezone data structure - Database seeded with default timezone data (if required) - User has permission to access timezone utilities
Test Steps	1. Initialize or import the Crater\Space\TimeZones class. 2. Call the get_list() method. 3. Retrieve its output array. 4. Compare the output array to the expected timezone list comprising value-key pairs for each timezone. 5. Validate that every expected timezone from the reference list is present, and the order matches. 6. Confirm each array element contains both 'value' (timezone identifier) and 'key' (timezone offset and descriptive location). 7. Assert the size of the returned array equals the size of the reference list.
Test Data	- Expected Timezone Array (e.g., [['value' => 'Pacific/Midway', 'key' => '(UTC-11:00) Midway'], ['value' => 'Pacific/Niue', 'key' => '(UTC-11:00) Niue'], ... [FULL LIST AS DEFINED IN THE TEST]]) - Actual timezone list returned by get_list()
Expected Result	- The timezone array returned from get_list() exactly matches the array of expected timezones in value and key pairs. - The array includes all world timezones specified by the business requirements and in the reference array. - No missing, duplicate, or malformed timezone entries.
Actual Result	Timezone array from get_list() matches the expected reference array exactly; all timezones are present, formatted correctly, and in the correct order.
Status	Pass
Severity	High

File: Transaction-Test.txt

Test Case ID	T-001
Title	Transaction Model instantiation
Objective	Verify that a Transaction model object can be created
Preconditions	- Application running - Database seeded - Crater\Models\Transaction class is available
Test Steps	1. Instantiate a new Transaction model object. 2. Verify that the object is an instance of Transaction.
Test Data	- Instantiation: new Transaction() - Expected instance: Transaction::class
Expected Result	- The created object is an instance of Transaction.
Actual Result	- The object is a valid instance of Transaction.
Status	Pass
Severity	High

Test Case ID	T-002
Title	Transaction extends Model and uses HasFactory trait
Objective	Ensure Transaction extends Eloquent Model and incorporates HasFactory trait
Preconditions	- Application running - Database seeded - Crater\Models\Transaction class is available
Test Steps	1. Instantiate a Transaction object. 2. Check if Transaction is an instance of Illuminate\Database\Eloquent\Model. 3. Use reflection to retrieve trait names for Transaction. 4. Check if HasFactory trait is present.
Test Data	- Class: Transaction - Trait: Illuminate\Database\Eloquent\Factories\HasFactory
Expected Result	- Transaction is an instance of Model. - Transaction uses HasFactory trait.
Actual Result	- Transaction extends Model and HasFactory trait is applied.
Status	Pass
Severity	High

Test Case ID	T-003
Title	Transaction has status constants defined
Objective	Verify Transaction defines status constants PENDING, FAILED, SUCCESS correctly
Preconditions	- Application running - Database seeded - Crater\Models\Transaction class is available
Test Steps	1. Access the constants Transaction::PENDING, Transaction::FAILED, Transaction::SUCCESS. 2. Confirm the values.
Test Data	- Constants: PENDING ('PENDING'), FAILED ('FAILED'), SUCCESS ('SUCCESS')
Expected Result	- Transaction::PENDING equals 'PENDING'. - Transaction::FAILED equals 'FAILED'. - Transaction::SUCCESS equals 'SUCCESS'.
Actual Result	- Status constants are defined as expected.

Status	Pass
Severity	High

Test Case ID	T-004
Title	Transaction defines guarded and dates properties
Objective	Ensure guarded and dates properties exist in the Transaction model
Preconditions	- Application running - Database seeded - Crater\Models\Transaction class is available
Test Steps	1. Use reflection to check if Transaction has the 'guarded' property. 2. Use reflection to check if Transaction has the 'dates' property.
Test Data	- Property names: guarded, dates
Expected Result	- The 'guarded' property exists. - The 'dates' property exists.
Actual Result	- Both 'guarded' and 'dates' properties found in Transaction model.
Status	Pass
Severity	High

Test Case ID	T-005
Title	Transaction model includes relationship and action methods
Objective	Confirm existence of key relationship and action methods in Transaction model
Preconditions	- Application running - Database seeded - Crater\Models\Transaction class is available
Test Steps	1. Use reflection to verify presence of methods: payments, invoice, company. 2. Verify presence of action methods: completeTransaction, failedTransaction, createTransaction, isExpired.
Test Data	- Method names: payments, invoice, company, completeTransaction, failedTransaction, createTransaction, isExpired
Expected Result	- Each listed method exists in Transaction model.
Actual Result	- All specified methods are present.
Status	Pass
Severity	High

Test Case ID	T-006
Title	Transaction payments returns HasMany relationship
Objective	Verify that payments() in Transaction returns a HasMany relation
Preconditions	- Application running - Database seeded - Crater\Models\Transaction and related Payment models available
Test Steps	1. Instantiate a new Transaction object. 2. Call payments() on Transaction object. 3. Check if the return value is a HasMany relation.
Test Data	- Transaction object - payments() relation
Expected Result	- payments() returns instance of Illuminate\Database\Eloquent\Relations\HasMany.
Actual Result	- payments() relationship verified as HasMany.
Status	Pass

Severity	High

Test Case ID	T-007
Title	Transaction invoice returns BelongsTo relationship
Objective	Ensure invoice() method on Transaction returns BelongsTo relation
Preconditions	- Application running - Database seeded - Crater\Models\Transaction and related Invoice models available
Test Steps	1. Instantiate a Transaction object. 2. Call invoice() on Transaction. 3. Assert that return value is BelongsTo relation.
Test Data	- Transaction object - invoice() relation
Expected Result	- invoice() returns an instance of Illuminate\Database\Eloquent\Relations\BelongsTo.
Actual Result	- invoice() relationship verified as BelongsTo.
Status	Pass
Severity	High

Test Case ID	T-008
Title	Transaction company returns BelongsTo relationship
Objective	Confirm company() method on Transaction gives BelongsTo relation
Preconditions	- Application running - Database seeded - Crater\Models\Transaction and related Company models available
Test Steps	1. Instantiate a Transaction object. 2. Call company() on Transaction. 3. Assert that return value is BelongsTo relation.
Test Data	- Transaction object - company() relation
Expected Result	- company() returns an instance of Illuminate\Database\Eloquent\Relations\BelongsTo.
Actual Result	- company() relationship verified as BelongsTo.
Status	Pass
Severity	High

Test Case ID	T-009
Title	CompleteTransaction method sets status to SUCCESS
Objective	Verify that calling completeTransaction sets Transaction status to SUCCESS
Preconditions	- Application running - Database seeded - Crater\Models\Transaction class file available
Test Steps	1. Use reflection to locate the Transaction class source file. 2. Inspect the file for assignment to $this->status = self::SUCCESS in completeTransaction. 3. Check that $this->save() is called after the assignment.
Test Data	- Method: completeTransaction - Status value: SUCCESS
Expected Result	- Method contains: $this->status = self::SUCCESS - Method contains: $this->save() call
Actual Result	- completeTransaction sets status to SUCCESS and saves the model.

Status	Pass
Severity	High

Test Case ID	T-010
Title	FailedTransaction method sets status to FAILED
Objective	Verify failedTransaction sets Transaction status to FAILED
Preconditions	- Application running - Database seeded - Crater\Models\Transaction class file available
Test Steps	1. Use reflection to locate the Transaction class source file. 2. Extract file contents. 3. Confirm the method assigns $this->status = self::FAILED.
Test Data	- Method: failedTransaction - Status value: FAILED
Expected Result	- Method contains: $this->status = self::FAILED
Actual Result	- failedTransaction sets status to FAILED.
Status	Pass
Severity	High

Test Case ID	T-011
Title	TransactionCollection instantiation
Objective	Verify TransactionCollection can be instantiated with an empty collection
Preconditions	- Application running - Crater\Http\Resources\TransactionCollection class available
Test Steps	1. Create an empty Illuminate\Support\Collection. 2. Instantiate TransactionCollection with the empty collection. 3. Confirm the object is an instance of TransactionCollection.
Test Data	- Collection: new Collection([]) - Resource: TransactionCollection
Expected Result	- TransactionCollection object is instantiated correctly.
Actual Result	- TransactionCollection successfully instantiated.
Status	Pass
Severity	Medium

Test Case ID	T-012
Title	TransactionCollection extends ResourceCollection
Objective	Ensure TransactionCollection extends the base ResourceCollection
Preconditions	- Application running - Crater\Http\Resources\TransactionCollection class available
Test Steps	1. Create an empty Collection. 2. Instantiate TransactionCollection. 3. Confirm the object is an instance of Illuminate\Http\Resources\Json\ResourceCollection.
Test Data	- Collection: new Collection([]) - Comparison class: ResourceCollection
Expected Result	- TransactionCollection extends ResourceCollection.
Actual Result	- TransactionCollection extends ResourceCollection as expected.
Status	Pass
Severity	Medium

Test Case ID	T-013
Title	TransactionCollection is in correct namespace
Objective	Confirm TransactionCollection is in 'Crater\Http\Resources' namespace
Preconditions	- Application running - Crater\Http\Resources\TransactionCollection class available
Test Steps	1. Use reflection to inspect TransactionCollection class. 2. Get namespace name. 3. Verify namespace matches 'Crater\Http\Resources'.
Test Data	- Class: TransactionCollection
Expected Result	- Namespace name is 'Crater\Http\Resources'.
Actual Result	- TransactionCollection found in correct namespace.
Status	Pass
Severity	Low

Test Case ID	T-014
Title	TransactionCollection toArray returns empty for empty collection
Objective	Validate that toArray() returns an empty array for an empty TransactionCollection
Preconditions	- Application running - Crater\Http\Resources\TransactionCollection class available
Test Steps	1. Create an empty Collection. 2. Instantiate TransactionCollection with empty Collection. 3. Call toArray() with a Request object. 4. Verify result is an empty array.
Test Data	- Collection: [] - Request: new Request()
Expected Result	- Result is an empty array.
Actual Result	- toArray returns empty array for empty collection.
Status	Pass
Severity	Medium

Test Case ID	T-015
Title	TransactionCollection toArray returns array and delegates to parent
Objective	Confirm that TransactionCollection toArray produces an array output and delegates logic
Preconditions	- Application running - Crater\Http\Resources\TransactionCollection class available
Test Steps	1. Create empty Collection. 2. Instantiate TransactionCollection. 3. Call toArray with a Request object. 4. Confirm result is an array.
Test Data	- Collection: [] - Request: new Request()
Expected Result	- Output is an array.
Actual Result	- toArray returns array as expected.
Status	Pass
Severity	Medium

Test Case ID	T-016

Title	TransactionResource can be instantiated
Objective	Verify TransactionResource instantiates correctly with valid data
Preconditions	- Application running - Crater\Http\Resources\TransactionResource class available
Test Steps	1. Create a data object with id and status properties. 2. Instantiate TransactionResource with the data. 3. Confirm TransactionResource instance is created.
Test Data	- Data: (object)['id' => 1, 'status' => 'SUCCESS']
Expected Result	- TransactionResource instance is created.
Actual Result	- TransactionResource successfully instantiated with data.
Status	Pass
Severity	Medium

Test Case ID	T-017
Title	TransactionResource extends JsonResource
Objective	Confirm TransactionResource inherits from JsonResource
Preconditions	- Application running - Crater\Http\Resources\TransactionResource class available
Test Steps	1. Create a generic object. 2. Instantiate TransactionResource with generic data. 3. Verify TransactionResource is an instance of JsonResource.
Test Data	- Data: (object)[]
Expected Result	- TransactionResource is instance of Illuminate\Http\Resources\Json\JsonResource.
Actual Result	- TransactionResource extends JsonResource as expected.
Status	Pass
Severity	Medium

Test Case ID	T-018
Title	TransactionResource toArray includes all basic fields
Objective	Ensure toArray() returns basic fields for TransactionResource
Preconditions	- Application running - Crater\Http\Resources\TransactionResource source code available
Test Steps	1. Use reflection to access TransactionResource class file content. 2. Confirm toArray includes: id, transaction_id, type, status, transaction_date, invoice_id.
Test Data	- Expected keys: 'id', 'transaction_id', 'type', 'status', 'transaction_date', 'invoice_id'
Expected Result	- toArray includes all specified fields.
Actual Result	- All basic fields present in toArray output.
Status	Pass
Severity	High

Test Case ID	T-019
Title	TransactionResource toArray includes conditional invoice relationship
Objective	Confirm that toArray includes 'invoice' only if relationship exists
Preconditions	- Application running - Crater\Http\Resources\TransactionResource source code available

Test Steps	1. Use reflection to locate source code for TransactionResource. 2. Confirm toArray checks if invoice()->exists(), and includes 'invoice' field conditionally. 3. Verify that InvoiceResource is returned for 'invoice' field.
Test Data	- Relationship: invoice - Conditional code: when($this->invoice()->exists())
Expected Result	- 'invoice' field present only when invoice relationship exists and returns InvoiceResource.
Actual Result	- Conditional invoice relationship included correctly in toArray.
Status	Pass
Severity	High

Test Case ID	T-020
Title	TransactionResource toArray includes conditional company relationship
Objective	Confirm that toArray includes 'company' only if relationship exists
Preconditions	- Application running - Crater\Http\Resources\TransactionResource source code available
Test Steps	1. Use reflection to locate source code for TransactionResource. 2. Confirm toArray checks if company()->exists(), and includes 'company' field conditionally. 3. Verify that CompanyResource is returned for 'company' field.
Test Data	- Relationship: company - Conditional code: when($this->company()->exists())
Expected Result	- 'company' field present only when company relationship exists and returns CompanyResource.
Actual Result	- Conditional company relationship included correctly in toArray.
Status	Pass
Severity	High

File: TrimStrings-Test.txt

Test Case ID	TS-001
Title	Verify subclassing of base TrimStrings middleware
Objective	Ensure that the Crater TrimStrings middleware correctly extends the Illuminate base TrimStrings middleware
Preconditions	- Application running - Required PHP dependencies installed - Crater middleware available - No prior test run interference
Test Steps	1. Instantiate an object of Crater\Http\Middleware\TrimStrings. 2. Check if the instantiated object is an instance of Illuminate\Foundation\Http\Middleware\TrimStrings.
Test Data	- Object: Crater\Http\Middleware\TrimStrings - Class to check: Illuminate\Foundation\Http\Middleware\TrimStrings
Expected Result	- The Crater TrimStrings middleware object is an instance of the base TrimStrings middleware class.
Actual Result	The middleware was verified as an instance of the base TrimStrings class.
Status	Pass
Severity	Medium

Test Case ID	TS-002
Title	Verify the except attributes of TrimStrings middleware
Objective	Ensure that the TrimStrings middleware defines the correct attributes ('password' and 'password_confirmation') that are not trimmed
Preconditions	- Application running - Required PHP dependencies installed - Crater middleware available - Access to ReflectionClass for white-box property access
Test Steps	1. Instantiate an object of Crater\Http\Middleware\TrimStrings. 2. Use PHP reflection to access the protected 'except' property of the middleware object. 3. Retrieve the 'except' property value. 4. Assert that the value is an array. 5. Assert that the array contains 'password' and 'password_confirmation'. 6. Assert that the array has exactly 2 elements.
Test Data	- Expected 'except' attributes: ['password', 'password_confirmation']
Expected Result	- The 'except' property is an array. - The array contains 'password' and 'password_confirmation'. - The array has a total of 2 elements.
Actual Result	The 'except' property was an array containing only 'password' and 'password_confirmation'. The array length was verified as 2.
Status	Pass
Severity	Medium

File: TrustProxies-Test.txt

Test Case ID	TP-001
Title	Verify TrustProxies Extends Fideloper TrustProxies Middleware
Objective	Ensure that TrustProxies middleware correctly extends the Fideloper TrustProxies middleware class.
Preconditions	- Application running - Crater\Http\Middleware\TrustProxies class available - Fideloper\Proxy\TrustProxies middleware installed - Database seeded (if required)
Test Steps	1. Use PHP ReflectionClass to inspect the TrustProxies class. 2. Retrieve the parent class of TrustProxies using getParentClass(). 3. Assert that the parent class name is 'Fideloper\Proxy\TrustProxies'.
Test Data	- TrustProxies class - Expected parent class: 'Fideloper\Proxy\TrustProxies'
Expected Result	- The parent class of TrustProxies is 'Fideloper\Proxy\TrustProxies'.
Actual Result	Parent class was found to be 'Fideloper\Proxy\TrustProxies' as expected.
Status	Pass
Severity	High

Test Case ID	TP-002
Title	Verify TrustProxies Namespace
Objective	Ensure that the TrustProxies class resides in the correct namespace.
Preconditions	- Application running - Crater\Http\Middleware\TrustProxies class available
Test Steps	1. Use PHP ReflectionClass to inspect the TrustProxies class. 2. Retrieve the namespace name of TrustProxies using getNamespaceName(). 3. Assert that the namespace is 'Crater\Http\Middleware'.
Test Data	- TrustProxies class - Expected namespace: 'Crater\Http\Middleware'
Expected Result	- TrustProxies class is located in 'Crater\Http\Middleware' namespace.
Actual Result	Namespace was confirmed to be 'Crater\Http\Middleware'.
Status	Pass
Severity	Medium

Test Case ID	TP-003
Title	Verify TrustProxies Has Proxies and Headers Properties
Objective	Ensure TrustProxies class defines 'proxies' and 'headers' properties.
Preconditions	- Application running - Crater\Http\Middleware\TrustProxies class available
Test Steps	1. Use PHP ReflectionClass to inspect the TrustProxies class. 2. Use hasProperty() to check if the 'proxies' property exists. 3. Use hasProperty() to check if the 'headers' property exists. 4. Assert both properties are present.
Test Data	- TrustProxies class - Expected properties: 'proxies', 'headers'
Expected Result	- TrustProxies contains a 'proxies' property. - TrustProxies contains a 'headers' property.
Actual Result	Both 'proxies' and 'headers' properties are present in TrustProxies.

Status	Pass
Severity	Medium

Test Case ID	TP-004
Title	Verify Proxies and Headers Properties Visibility in TrustProxies
Objective	Ensure 'proxies' and 'headers' properties in TrustProxies are protected.
Preconditions	- Application running - Crater\Http\Middleware\TrustProxies class available
Test Steps	1. Use PHP ReflectionClass to inspect the TrustProxies class. 2. Retrieve 'proxies' and 'headers' properties with getProperty(). 3. Check if both properties are protected using isProtected(). 4. Assert both properties have protected visibility.
Test Data	- TrustProxies class - Properties: 'proxies', 'headers' - Expected visibility: protected
Expected Result	- 'proxies' property is protected. - 'headers' property is protected.
Actual Result	Both 'proxies' and 'headers' properties confirmed to be protected.
Status	Pass
Severity	Medium

Test Case ID	TP-005
Title	Verify Usage of Request::HEADER_X_FORWARDED_ALL Constant in TrustProxies
Objective	Ensure TrustProxies uses the Request::HEADER_X_FORWARDED_ALL constant in its code.
Preconditions	- Application running - Crater\Http\Middleware\TrustProxies class available - Illuminate\Http\Request available
Test Steps	1. Use PHP ReflectionClass to get the filename of the TrustProxies class. 2. Read the class source file to retrieve its contents. 3. Search the source code for the string 'Request::HEADER_X_FORWARDED_ALL'. 4. Assert that the constant is present in the file.
Test Data	- TrustProxies class source code - Expected string: 'Request::HEADER_X_FORWARDED_ALL'
Expected Result	- Source code of TrustProxies contains 'Request::HEADER_X_FORWARDED_ALL'.
Actual Result	String 'Request::HEADER_X_FORWARDED_ALL' found in TrustProxies source code.
Status	Pass
Severity	High

File: Unit-Test.txt

Test Case ID	U-001
Title	Verify Unit Model Fillable Attributes
Objective	Ensure that the Unit model exposes the correct fillable attributes: "name" and "company_id".
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Instantiate a new Unit model object. 2. Retrieve the fillable attributes using getFillable(). 3. Verify the fillable attributes are ["name", "company_id"].
Test Data	- No input data required - Expected fillable attributes: ["name", "company_id"]
Expected Result	The fillable attributes of the Unit model are ["name", "company_id"].
Actual Result	The fillable attributes matched: ["name", "company_id"].
Status	Pass
Severity	Medium

Test Case ID	U-002
Title	Verify Items Relationship in Unit Model
Objective	Ensure that the "items" relationship in the Unit model is a HasMany relationship with correct configuration.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Instantiate a new Unit model object. 2. Call the items() relationship method. 3. Check that the returned relation is an instance of HasMany. 4. Ensure that the related model is Item. 5. Verify the foreign key name is "unit_id". 6. Verify the local key name is "id".
Test Data	- No input data required
Expected Result	- The items() relationship returns an instance of HasMany. - The related model is Item. - The foreign key is "unit_id". - The local key is "id".
Actual Result	The relationship is HasMany, related model is Item, foreign key "unit_id", local key "id".
Status	Pass
Severity	Medium

Test Case ID	U-003
Title	Verify Company Relationship in Unit Model
Objective	Ensure the "company" relationship in the Unit model is a BelongsTo relationship with correct configuration.
Preconditions	- Application running - Database seeded - Laravel models loaded

Test Steps	1. Instantiate a Unit model object. 2. Call the company() relationship method. 3. Check that the returned relation is an instance of BelongsTo. 4. Ensure that the related model is Company. 5. Verify the foreign key name is "company_id". 6. Verify the owner key name is "id".
Test Data	- No input data required
Expected Result	- The company() relationship returns an instance of BelongsTo. - The related model is Company. - The foreign key is "company_id". - The owner key is "id".
Actual Result	The relationship is BelongsTo, related model is Company, foreign key "company_id", owner key "id".
Status	Pass
Severity	Medium

Test Case ID	U-004
Title	Verify scopeWhereUnit Applies orWhere Filter with Unit ID
Objective	Validate that scopeWhereUnit applies an orWhere filter using the provided unit ID.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect orWhere('id', unitId). 3. Instantiate Unit model. 4. Call scopeWhereUnit with mock Builder and unitId = 456.
Test Data	- unitId: 456
Expected Result	The orWhere method on Builder is called with parameters ('id', 456) exactly once.
Actual Result	The orWhere method was called with ('id', 456) as expected.
Status	Pass
Severity	Medium

Test Case ID	U-005
Title	Verify scopeWhereUnit Applies orWhere Filter with Null Unit ID
Objective	Validate that scopeWhereUnit applies an orWhere filter using null when unit ID is not provided.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect orWhere('id', null). 3. Instantiate Unit model. 4. Call scopeWhereUnit with mock Builder and unitId = null.
Test Data	- unitId: null
Expected Result	The orWhere method on Builder is called with parameters ('id', null) exactly once.
Actual Result	The orWhere method was called with ('id', null) as expected.
Status	Pass
Severity	Medium

Test Case ID	U-006
Title	Verify scopeWhereSearch Applies Name LIKE Filter
Objective	Ensure scopeWhereSearch applies a "name LIKE %search%" filter when a search string is provided.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect where('name', 'LIKE', '%test unit%'). 3. Instantiate Unit model. 4. Call scopeWhereSearch with mock Builder and search = 'test unit'.
Test Data	- search: 'test unit'
Expected Result	- The where method on Builder is called with parameters ('name', 'LIKE', '%test unit%') once. - The returned value is the same Builder instance.
Actual Result	where('name', 'LIKE', '%test unit%') was called, Builder instance returned.
Status	Pass
Severity	Medium

Test Case ID	U-007
Title	Verify scopeWhereSearch Applies Name LIKE Filter with Empty Search String
Objective	Ensure scopeWhereSearch applies "name LIKE %%" when the search string is empty.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect where('name', 'LIKE', '%%'). 3. Instantiate Unit model. 4. Call scopeWhereSearch with mock Builder and search = ''.
Test Data	- search: ''
Expected Result	- The where method on Builder is called with ('name', 'LIKE', '%%') once. - The returned value is the same Builder instance.
Actual Result	where('name', 'LIKE', '%%') was called, Builder instance returned.
Status	Pass
Severity	Medium

Test Case ID	U-008
Title	Verify scopeApplyFilters Applies Search Filter
Objective	Ensure scopeApplyFilters applies the search filter using whereSearch when "search" is present in filters.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect whereSearch with 'desk'. 3. Instantiate Unit model. 4. Call scopeApplyFilters with mock Builder and filters: ['search' => 'desk'].
Test Data	- filters: ['search' => 'desk']
Expected Result	- The whereSearch method on Builder is called with 'desk' once. - The returned value is the same Builder instance.
Actual Result	whereSearch('desk') called, Builder instance returned.

Status	Pass
Severity	Medium

Test Case ID	U-009
Title	Verify scopeApplyFilters Applies Unit ID Filter
Objective	Ensure scopeApplyFilters applies the unit_id filter using whereUnit when "unit_id" is present in filters.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect whereUnit with value 1. 3. Instantiate Unit model. 4. Call scopeApplyFilters with mock Builder and filters: ['unit_id' => 1].
Test Data	- filters: ['unit_id' => 1]
Expected Result	- The whereUnit method on Builder is called with 1 once. - The returned value is the same Builder instance.
Actual Result	whereUnit(1) called, Builder instance returned.
Status	Pass
Severity	Medium

Test Case ID	U-010
Title	Verify scopeApplyFilters Applies No Filters When None Provided
Objective	Ensure scopeApplyFilters does not apply any filters when none are provided.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to not expect any filter methods: whereSearch, whereUnit, or where. 3. Instantiate Unit model. 4. Call scopeApplyFilters with mock Builder and filters: [].
Test Data	- filters: []
Expected Result	- No calls to whereSearch, whereUnit, or where are made. - The returned value is the same Builder instance.
Actual Result	No filter methods called, Builder instance returned.
Status	Pass
Severity	Medium

Test Case ID	U-011
Title	Verify scopeApplyFilters Ignores Unknown Filters
Objective	Ensure that unknown filter keys are ignored by scopeApplyFilters and do not cause any filter method calls.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to not expect whereSearch, whereUnit, or where. 3. Instantiate Unit model. 4. Call scopeApplyFilters with mock Builder and filters: {'unknown_filter' => 'value', 'another_unknown' => 123}.

Test Data	- filters: {'unknown_filter' => 'value', 'another_unknown' => 123}
Expected Result	- No calls to whereSearch, whereUnit, or where are made. - The returned value is the same Builder instance.
Actual Result	No filter methods called, Builder instance returned.
Status	Pass
Severity	Medium

Test Case ID	U-012
Title	Verify scopePaginateData Returns All Items When Limit is 'all'
Objective	Confirm that scopePaginateData calls get() and returns all results when limit is "all".
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect get(). 3. Instantiate Unit model. 4. Call scopePaginateData with mock Builder and limit: 'all'.
Test Data	- limit: 'all' - Expected result: 'all_results'
Expected Result	- get() method is called once on Builder. - paginate() is NOT called. - Returned value is 'all_results'.
Actual Result	get() called, returned 'all_results'.
Status	Pass
Severity	Medium

Test Case ID	U-013
Title	Verify scopePaginateData Paginates Items When Limit is Integer
Objective	Ensure scopePaginateData calls paginate with integer limit and returns paginated results.
Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect paginate(10). 3. Instantiate Unit model. 4. Call scopePaginateData with mock Builder and limit: 10.
Test Data	- limit: 10 - Expected result: 'paginated_results'
Expected Result	- paginate(10) is called once and get() is NOT called. - Returned value is 'paginated_results'.
Actual Result	paginate(10) called, returned 'paginated_results'.
Status	Pass
Severity	Medium

Test Case ID	U-014
Title	Verify scopePaginateData Paginates When Limit is Arbitrary String
Objective	Ensure scopePaginateData calls paginate when limit is a string other than "all".

Preconditions	- Application running - Database seeded - Laravel models loaded
Test Steps	1. Initialize a mock Builder object. 2. Set mock to expect paginate('invalid_string'). 3. Instantiate Unit model. 4. Call scopePaginateData with mock Builder and limit: 'invalid_string'.
Test Data	- limit: 'invalid_string' - Expected result: 'paginated_results_string'
Expected Result	- paginate('invalid_string') is called once and get() is NOT called. - Returned value is 'paginated_results_string'.
Actual Result	paginate('invalid_string') called, returned 'paginated_results_string'.
Status	Pass
Severity	Medium

File: UnitPolicyReqAndResource-Test.txt

Test Case ID	UPRAR-001
Title	Instantiate UnitPolicy
Objective	Verify that the UnitPolicy class can be instantiated successfully.
Preconditions	- Application running - Crater\Policies\UnitPolicy class available
Test Steps	1. Instantiate a new object of the UnitPolicy class. 2. Verify the object is an instance of UnitPolicy.
Test Data	- Class: UnitPolicy
Expected Result	- The instantiated object is an instance of Crater\Policies\UnitPolicy.
Actual Result	The object is correctly instantiated as an instance of UnitPolicy.
Status	Pass
Severity	Medium

Test Case ID	UPRAR-002
Title	Verify HandlesAuthorization trait usage in UnitPolicy
Objective	Ensure that UnitPolicy uses the Illuminate\Auth\Access\HandlesAuthorization trait.
Preconditions	- Application running - Crater\Policies\UnitPolicy class available
Test Steps	1. Reflect on the UnitPolicy class. 2. Retrieve the trait names used in the class. 3. Check if HandlesAuthorization is present in the trait list.
Test Data	- Trait: Illuminate\Auth\Access\HandlesAuthorization
Expected Result	- The trait list contains 'Illuminate\Auth\Access\HandlesAuthorization'.
Actual Result	HandlesAuthorization trait is present in UnitPolicy.
Status	Pass
Severity	Medium

Test Case ID	UPRAR-003
Title	Validate namespace of UnitPolicy
Objective	Confirm that UnitPolicy resides in the 'Crater\Policies' namespace.
Preconditions	- Application running - Crater\Policies\UnitPolicy class available
Test Steps	1. Reflect on the UnitPolicy class. 2. Retrieve the namespace name. 3. Compare with expected value.
Test Data	- Expected namespace: Crater\Policies
Expected Result	- The namespace of UnitPolicy is 'Crater\Policies'.
Actual Result	UnitPolicy is in the 'Crater\Policies' namespace.
Status	Pass
Severity	Medium

Test Case ID	UPRAR-004
Title	Verify all required authorization methods in UnitPolicy
Objective	Ensure all required authorization methods exist in UnitPolicy.

Preconditions	- Application running - Crater\Policies\UnitPolicy class available
Test Steps	1. Reflect on the UnitPolicy class. 2. Check for presence of methods: viewAny, view, create, update, delete, restore, forceDelete.
Test Data	- List of method names
Expected Result	- UnitPolicy class contains all methods: viewAny, view, create, update, delete, restore, forceDelete.
Actual Result	All required methods are present in UnitPolicy.
Status	Pass
Severity	High

Test Case ID	UPRAR-005
Title	UnitPolicy uses BouncerFacade for authorization
Objective	Check that UnitPolicy calls BouncerFacade for authorization.
Preconditions	- Application running - Crater\Policies\UnitPolicy class available
Test Steps	1. Reflect on UnitPolicy. 2. Read class source file. 3. Ensure the file contains call to BouncerFacade::can('view-item', Item::class).
Test Data	- Source code content
Expected Result	- The source code contains "BouncerFacade::can('view-item', Item::class)".
Actual Result	The BouncerFacade authorization call is present in UnitPolicy source code.
Status	Pass
Severity	High

Test Case ID	UPRAR-006
Title	UnitPolicy checks company ownership for specific unit actions
Objective	Validate that UnitPolicy checks user's ownership of the company for unit actions.
Preconditions	- Application running - Crater\Policies\UnitPolicy class available
Test Steps	1. Reflect on UnitPolicy. 2. Read class source file. 3. Verify presence of company ownership check: $user->hasCompany($unit->company_id).
Test Data	- Source code content
Expected Result	- The source contains '$user->hasCompany($unit->company_id)'.
Actual Result	Company ownership check is present in UnitPolicy source code.
Status	Pass
Severity	High

Test Case ID	UPRAR-007
Title	Validate parameters of viewAny method in UnitPolicy
Objective	Ensure viewAny only has the 'user' parameter.
Preconditions	- Application running - Crater\Policies\UnitPolicy class available

Test Steps	1. Reflect on UnitPolicy. 2. Retrieve the viewAny method. 3. Inspect the parameters. 4. Confirm there is exactly one parameter named 'user'.
Test Data	- Method signature of viewAny
Expected Result	- viewAny accepts only one parameter named 'user'.
Actual Result	viewAny only has the 'user' parameter as expected.
Status	Pass
Severity	High

Test Case ID	UPRAR-008
Title	Validate parameters of view method in UnitPolicy
Objective	Ensure the view method has parameters 'user' and 'unit'.
Preconditions	- Application running - Crater\Policies\UnitPolicy class available
Test Steps	1. Reflect on UnitPolicy. 2. Retrieve the view method. 3. Inspect the parameters. 4. Confirm parameters are 'user', 'unit' (in order) and total parameter count is 2.
Test Data	- Method signature of view
Expected Result	- view has two parameters: 'user' and 'unit'.
Actual Result	view method correctly has parameters 'user' and 'unit'.
Status	Pass
Severity	High

Test Case ID	UPRAR-009
Title	Instantiate UnitRequest
Objective	Confirm that UnitRequest can be instantiated.
Preconditions	- Application running - Crater\Http\Requests\UnitRequest class available
Test Steps	1. Instantiate a new UnitRequest object. 2. Verify that the object is an instance of UnitRequest.
Test Data	- Class: UnitRequest
Expected Result	- The object instantiates as a UnitRequest instance.
Actual Result	UnitRequest successfully instantiated.
Status	Pass
Severity	Medium

Test Case ID	UPRAR-010
Title	Verify UnitRequest extends FormRequest
Objective	Ensure UnitRequest inherits from FormRequest.
Preconditions	- Application running - Crater\Http\Requests\UnitRequest class available
Test Steps	1. Instantiate UnitRequest. 2. Check if the object is an instance of Illuminate\Foundation\Http\FormRequest.
Test Data	- Class hierarchy
Expected Result	- UnitRequest is an instance of FormRequest.

Actual Result	UnitRequest correctly extends FormRequest.
Status	Pass
Severity	Medium

Test Case ID	UPRAR-011
Title	UnitRequest authorize method returns true
Objective	Validate that calling authorize on UnitRequest returns true.
Preconditions	- Application running - Crater\Http\Requests\UnitRequest class available
Test Steps	1. Instantiate UnitRequest. 2. Call the authorize() method. 3. Verify the return value is true.
Test Data	- Method: authorize()
Expected Result	- authorize() returns true.
Actual Result	authorize() returned true.
Status	Pass
Severity	High

Test Case ID	UPRAR-012
Title	UnitRequest rules include unique validation with company_id
Objective	Confirm rules() returns an array with unique constraint using company_id from header.
Preconditions	- Application running - Crater\Http\Requests\UnitRequest class available - Header 'company' is set
Test Steps	1. Instantiate UnitRequest. 2. Set 'company' header to '123'. 3. Call rules(). 4. Check that result is an array, contains key 'name', and 'name' maps to an array.
Test Data	- Header: company=123
Expected Result	- rules() returns array containing 'name' as a key, and 'name' is an array (array of constraints).
Actual Result	rules() returned correct array structure with unique validation including company_id.
Status	Pass
Severity	High

Test Case ID	UPRAR-013
Title	getUnitPayload merges company_id from header in UnitRequest
Objective	Ensure getUnitPayload merges 'company_id' from header with form data.
Preconditions	- Application running - Crater\Http\Requests\UnitRequest class available - Header 'company' is set - Request body with 'name' field
Test Steps	1. Instantiate UnitRequest. 2. Set 'company' header to '456'. 3. Merge ['name' => 'Test Unit'] into the request. 4. Mock validated() to merge company_id from header. 5. Collect request data and verify presence and value of company_id and name.

Test Data	- Header: company=456 - Payload: name='Test Unit'
Expected Result	- Resulting payload contains 'company_id' equal to 456 and 'name' equal to 'Test Unit'.
Actual Result	Payload contains 'company_id' = '456' and 'name' = 'Test Unit' as expected.
Status	Pass
Severity	High

Test Case ID	UPRAR-014
Title	Instantiate UnitResource with data
Objective	Confirm that UnitResource can be instantiated with valid data.
Preconditions	- Application running - Crater\Http\Resources\UnitResource class available - Data object with id and name
Test Steps	1. Create data object with id=1, name='Piece'. 2. Instantiate UnitResource with the data object. 3. Verify the object is instance of UnitResource.
Test Data	- Data: id=1, name='Piece'
Expected Result	- UnitResource is instantiated successfully.
Actual Result	UnitResource instance created successfully with provided data.
Status	Pass
Severity	Medium

Test Case ID	UPRAR-015
Title	Verify UnitResource extends JsonResource
Objective	Ensure that UnitResource is a subclass of JsonResource.
Preconditions	- Application running - Crater\Http\Resources\UnitResource class available
Test Steps	1. Instantiate UnitResource with an empty object. 2. Verify the object is an instance of Illuminate\Http\Resources\Json\JsonResource.
Test Data	- Data: empty object
Expected Result	- UnitResource is a subclass of JsonResource.
Actual Result	UnitResource correctly extends JsonResource.
Status	Pass
Severity	Medium

Test Case ID	UPRAR-016
Title	UnitResource toArray includes all basic fields
Objective	Confirm that toArray returns an array with keys 'id', 'name', 'company_id'.
Preconditions	- Application running - Crater\Http\Resources\UnitResource class available
Test Steps	1. Reflect on UnitResource to get source code. 2. Locate toArray implementation. 3. Verify 'id', 'name', and 'company_id' are present in the returned array.
Test Data	- Array field keys
Expected Result	- The array returned by toArray contains keys: 'id', 'name', 'company_id'.
Actual Result	toArray includes 'id', 'name', and 'company_id' as expected.

Status	Pass
Severity	Medium

Test Case ID	UPRAR-017
Title	UnitResource toArray includes conditional company relationship
Objective	Ensure toArray conditionally adds 'company' using CompanyResource when company exists.
Preconditions	- Application running - Crater\Http\Resources\UnitResource class available
Test Steps	1. Reflect on UnitResource source code. 2. Locate toArray implementation. 3. Check for conditional inclusion of 'company' using when() and CompanyResource.
Test Data	- Condition: $this->company()->exists() - Resource: CompanyResource
Expected Result	- toArray includes 'company' key when company relationship exists, using CompanyResource.
Actual Result	Conditional relationship to CompanyResource present in toArray as expected.
Status	Pass
Severity	Medium

Test Case ID	UPRAR-018
Title	UnitResource has public toArray method with correct signature
Objective	Verify that toArray method is public and takes exactly one parameter.
Preconditions	- Application running - Crater\Http\Resources\UnitResource class available
Test Steps	1. Reflect on UnitResource. 2. Retrieve the toArray method. 3. Ensure it is public. 4. Check that it has exactly one parameter.
Test Data	- Method: toArray
Expected Result	- toArray is a public method with exactly one parameter.
Actual Result	toArray method is public and has one parameter.
Status	Pass
Severity	Medium

File: UnzipModuleController-Test.txt

Test Case ID	UMC-001
Title	Successfully unzip a module and return the path
Objective	Verify that a valid module can be unzipped and the correct path is returned in the response.
Preconditions	- Application running - Database seeded - Module 'test-module' zip available in /tmp/modules - Proper file system permissions - Mocking framework is configured
Test Steps	1. Mock UnzipUpdateRequest with module = 'test-module' and path = '/tmp/modules'. 2. Mock ModuleInstaller::unzip to expect 'test-module' and '/tmp/modules', returning '/tmp/modules/test-module'. 3. Mock UnzipModuleController and authorize 'manage modules'. 4. Call controller's __invoke method with the mocked request. 5. Assert that the response is an instance of JsonResponse. 6. Extract response data and verify: - 'success' is true. - 'path' is '/tmp/modules/test-module'.
Test Data	- Input: module = 'test-module', path = '/tmp/modules' - Expected values: success = true, path = '/tmp/modules/test-module'
Expected Result	- Response is a JsonResponse object. - Response data is: { success: true, path: '/tmp/modules/test-module' }
Actual Result	Response is a JsonResponse; response data matches expected values.
Status	Pass
Severity	High

Test Case ID	UMC-002
Title	Exception is thrown if unzip fails
Objective	Verify that an exception is thrown when ModuleInstaller::unzip fails due to an error.
Preconditions	- Application running - Database seeded - Corrupted or missing module zip: 'bad-module' - Proper file system permissions - Mocking framework is configured
Test Steps	1. Mock UnzipUpdateRequest with module = 'bad-module' and path = '/tmp/modules'. 2. Mock ModuleInstaller::unzip to expect 'bad-module' and '/tmp/modules', and throw RuntimeException with message 'Failed to extract module zip file due to disk error.'. 3. Mock UnzipModuleController and authorize 'manage modules'. 4. Call controller's __invoke method with the mocked request. 5. Assert that RuntimeException is thrown with the expected message.
Test Data	- Input: module = 'bad-module', path = '/tmp/modules' - Exception message: 'Failed to extract module zip file due to disk error.'
Expected Result	- RuntimeException is thrown. - Exception message matches: 'Failed to extract module zip file due to disk error.'
Actual Result	RuntimeException is thrown as expected; exception message matches.
Status	Pass
Severity	High

Test Case ID	UMC-003
Title	Handle null path returned by unzip
Objective	Verify that the controller handles a null path returned by ModuleInstaller::unzip gracefully.
Preconditions	- Application running - Database seeded - Module zip 'empty-module' present in /tmp/modules - Proper file system permissions - Mocking framework is configured
Test Steps	1. Mock UnzipUpdateRequest with module = 'empty-module' and path = '/tmp/modules'. 2. Mock ModuleInstaller::unzip to expect 'empty-module' and '/tmp/modules', returning null. 3. Mock UnzipModuleController and authorize 'manage modules'. 4. Call controller's __invoke method with the mocked request. 5. Assert that the response is an instance of JsonResponse. 6. Extract response data and verify: - 'success' is true. - 'path' is null.
Test Data	- Input: module = 'empty-module', path = '/tmp/modules' - Expected values: success = true, path = null
Expected Result	- Response is a JsonResponse object. - Response data is: { success: true, path: null }
Actual Result	Response is a JsonResponse; response data matches expected values.
Status	Pass
Severity	High

File: UnzipUpdateController-Test.txt

Test Case ID	UUC-001
Title	Unauthorized Access When User is Not Logged In
Objective	Verify the system denies update access and returns a 401 status when no user is logged in.
Preconditions	- Application running - Database seeded - No user is currently authenticated
Test Steps	1. Simulate an HTTP request with no authenticated user. 2. Invoke the UnzipUpdateController with the request. 3. Capture the response.
Test Data	- Request: No authenticated user, empty request data. - Expected values: Response code 401, success = false, message = "You are not allowed to update this app."

Expected Result	- Response is an instance of JsonResponse.
	- Response HTTP status code is 401.
	- Response JSON contains:
	{
	"success": false,
	"message": "You are not allowed to update this app."
	}

Actual Result	Response is JsonResponse with status 401 and expected message.
Status	Pass
Severity	High

Test Case ID	UUC-002
Title	Unauthorized Access When User is Not Owner
Objective	Verify the system denies update access and returns a 401 status when user is authenticated but not the owner.
Preconditions	- Application running - Database seeded - Authenticated user exists but does not have ownership rights
Test Steps	1. Simulate an authenticated user who is not an owner. 2. Simulate an HTTP request containing that user. 3. Invoke the UnzipUpdateController with the request. 4. Capture the response.
Test Data	- Request: User authenticated, isOwner returns false, empty request data. - Expected values: Response code 401, success = false, message = "You are not allowed to update this app."

Expected Result	- Response is an instance of JsonResponse.
	- Response HTTP status code is 401.
	- Response JSON contains:
	{
	"success": false,
	"message": "You are not allowed to update this app."
	}

Actual Result	Response is JsonResponse with status 401 and expected message.
Status	Pass
Severity	High

Test Case ID	UUC-003
Title	Validation Exception When Path is Missing and User is Owner

Objective	Ensure request validation fails and returns a validation exception when no 'path' is provided, even for owner users.
Preconditions	- Application running - Database seeded - Authenticated user is owner
Test Steps	1. Simulate an authenticated user who is an owner. 2. Simulate an HTTP request without 'path' in the payload. 3. Set validation to fail on missing 'path'. 4. Invoke the UnzipUpdateController with the request. 5. Capture if ValidationException is thrown.
Test Data	- Request: User is owner, request data empty, 'path' missing. - Expected values: ValidationException thrown for missing 'path' with error "The path field is required."
Expected Result	- ValidationException is thrown indicating the 'path' field is required.
Actual Result	ValidationException is thrown for missing 'path' as expected.
Status	Pass
Severity	High

Test Case ID	UUC-004
Title	Successful Unzip When User is Owner and Path is Provided
Objective	Ensure that, when an owner submits a valid 'path' to an update file, the update is successfully unzipped and proper response is returned.
Preconditions	- Application running - Database seeded - Authenticated user is owner - Update zip file exists at '/tmp/update.zip'
Test Steps	1. Simulate an authenticated user who is an owner. 2. Simulate an HTTP request with valid 'path' '/tmp/update.zip'. 3. Mock the Updater::unzip to return '/app/unzipped_path' on success. 4. Invoke the UnzipUpdateController with the request. 5. Capture the response.
Test Data	- Request: User is owner, request data {'path': '/tmp/update.zip'} - Updater::unzip returns '/app/unzipped_path' - Expected values: Response code 200, success = true, path = '/app/unzipped_path'

Expected Result	- Response is an instance of JsonResponse.
	- Response HTTP status code is 200.
	- Response JSON contains:
	{
	"success": true,
	"path": "/app/unzipped_path"
	}

Actual Result	Response is JsonResponse with status 200 and the unzipped path as expected.
Status	Pass
Severity	High

Test Case ID	UUC-005
Title	Graceful Handling of Unzip Failure When User is Owner and Path is Provided
Objective	Verify that, if unzipping the update file fails for an owner, a descriptive error is returned with HTTP status 500.
Preconditions	- Application running - Database seeded - Authenticated user is owner - Update zip file exists at '/tmp/bad_update.zip'

Test Steps	1. Simulate an authenticated user who is an owner. 2. Simulate an HTTP request with 'path' set to '/tmp/bad_update.zip'. 3. Mock the Updater::unzip method to throw an exception with specific error message. 4. Invoke the UnzipUpdateController with the request. 5. Capture the response.
Test Data	- Request: User is owner, request data {'path': '/tmp/bad_update.zip'} - Updater::unzip throws Exception('Failed to unzip archive: Corrupted file.') - Expected values: Response code 500, success = false, error = 'Failed to unzip archive: Corrupted file.'

Expected Result	- Response is an instance of JsonResponse.
	- Response HTTP status code is 500.
	- Response JSON contains:
	{
	"success": false,
	"error": "Failed to unzip archive: Corrupted file."
	}

Actual Result	Response is JsonResponse with status 500 and expected error message.
Status	Pass
Severity	High

File: UnzipUpdateRequest-Test.txt

Test Case ID	UUR-001
Title	Authorize Method Always Returns True
Objective	Verify that the authorize() method in UnzipUpdateRequest always returns true, allowing the request to be authorized every time.
Preconditions	- Application is running - UnzipUpdateRequest class is loaded and accessible - No user-specific authorization constraints are present
Test Steps	1. Instantiate an object of the UnzipUpdateRequest class. 2. Call the authorize() method on the instantiated object. 3. Verify the returned value from the authorize() method.
Test Data	- UnzipUpdateRequest object without any parameters
Expected Result	- The authorize() method returns true.
Actual Result	authorize() method returned true as expected.
Status	Pass
Severity	High

Test Case ID	UUR-002
Title	Rules Method Returns Correct Validation Rules
Objective	Verify that the rules() method in UnzipUpdateRequest returns an array with correct validation rules for 'path' and 'module' fields.
Preconditions	- Application is running - UnzipUpdateRequest class is loaded and accessible
Test Steps	1. Instantiate an object of the UnzipUpdateRequest class. 2. Call the rules() method on the instantiated object and store the returned value. 3. Verify that the returned value is an array. 4. Check that the array contains the keys 'path' and 'module'. 5. For the 'path' key: a. Verify value is an array. b. Verify the array contains 'required'. c. Verify the array contains 'regex:/^[\.\/\w\-]+$/'. d. Verify the array has exactly 2 items. 6. For the 'module' key: a. Verify value is an array. b. Verify the array contains 'required'. c. Verify the array contains 'string'. d. Verify the array has exactly 2 items.
Test Data	- UnzipUpdateRequest object without any parameters - Expected 'path' rules: ['required', 'regex:/^[\.\/\w\-]+$/'] - Expected 'module' rules: ['required', 'string']
Expected Result	- rules() returns an array. - Array has keys: 'path' and 'module'. - 'path' value is array containing 'required' and 'regex:/^[\.\/\w\-]+$/', count = 2. - 'module' value is array containing 'required' and 'string', count = 2.
Actual Result	rules() returned array with correct structure and expected validation rules for both 'path' and 'module'.
Status	Pass
Severity	High

File: UpdateController-Test.txt

Test Case ID	UC-001
Title	Instantiate UpdateCompanySettingsController
Objective	Verify that the UpdateCompanySettingsController can be successfully instantiated.
Preconditions	- Application running - Classes loaded - Composer autoload configured
Test Steps	1. Attempt to create a new instance of UpdateCompanySettingsController. 2. Confirm the instance is of the correct class.
Test Data	- Controller class: UpdateCompanySettingsController
Expected Result	- A new object of type UpdateCompanySettingsController is instantiated.
Actual Result	UpdateCompanySettingsController was successfully instantiated.
Status	Pass
Severity	Medium

Test Case ID	UC-002
Title	UpdateCompanySettingsController extends base Controller
Objective	Ensure UpdateCompanySettingsController inherits from the core Controller class.
Preconditions	- Application running - Classes loaded
Test Steps	1. Instantiate UpdateCompanySettingsController. 2. Check if the instance is a subclass of \Crater\Http\Controllers\Controller.
Test Data	- Expected parent class: \Crater\Http\Controllers\Controller
Expected Result	- Instance is verified to extend the Controller base class.
Actual Result	UpdateCompanySettingsController is confirmed to extend Controller.
Status	Pass
Severity	Medium

Test Case ID	UC-003
Title	UpdateCompanySettingsController is invokable
Objective	Verify the presence of the __invoke method, making the controller invokable.
Preconditions	- Application running - Classes loaded
Test Steps	1. Create a ReflectionClass for UpdateCompanySettingsController. 2. Check if the class defines the '__invoke' method.
Test Data	- Method expected: __invoke
Expected Result	- '__invoke' method is present in the controller.
Actual Result	__invoke method exists in UpdateCompanySettingsController.
Status	Pass
Severity	Medium

Test Case ID	UC-004
Title	Authorization usage in UpdateCompanySettingsController
Objective	Ensure that the controller uses authorization for managing company.

Preconditions	- Application running - Source code available
Test Steps	1. Read the source file of UpdateCompanySettingsController. 2. Search for '$this->authorize('manage company', $company)' in the code.
Test Data	- Authorization string: 'manage company'
Expected Result	- Authorization code for managing company is present.
Actual Result	Authorization code for managing company was found.
Status	Pass
Severity	High

Test Case ID	UC-005
Title	Currency change logic is checked with transactions
Objective	Verify currency change logic checks for existing transactions before updating currency.
Preconditions	- Application running - Source code available
Test Steps	1. Access UpdateCompanySettingsController source code. 2. Look for currency existence check via 'Arr::exists($data, 'currency')'. 3. Confirm use of 'CompanySetting::getSetting('currency')'. 4. Check for transactions check via '$company->hasTransactions()'.
Test Data	- Currency parameter: $data['currency'] - Transaction check method: $company->hasTransactions()
Expected Result	- The code checks for an attempted currency change and determines if the company has transactions before updating.
Actual Result	Currency change logic with transactions was correctly implemented.
Status	Pass
Severity	High

Test Case ID	UC-006
Title	Controller returns error for currency change after transactions
Objective	Ensure controller responds with error if trying to change currency after transactions exist.
Preconditions	- Application running - Source code available - Company with existing transactions
Test Steps	1. Read UpdateCompanySettingsController source. 2. Confirm code returns 'success' => false in response. 3. Verify response contains message: 'Cannot update company currency after transactions are created'.
Test Data	- Error message: 'Cannot update company currency after transactions are created'
Expected Result	- Controller responds with an error and sets success to false when currency change attempted after transactions.
Actual Result	Error message and success=false returned for currency change with existing transactions.
Status	Pass
Severity	High

Test Case ID	UC-007
Title	Usage of CompanySetting::setSettings

Objective	Verify that CompanySetting::setSettings is used to update company settings.
Preconditions	- Application running - Source code available
Test Steps	1. Read UpdateCompanySettingsController source code. 2. Confirm presence of 'CompanySetting::setSettings($data, $request->header('company'))'.
Test Data	- CompanySetting::setSettings function call
Expected Result	- Controller uses CompanySetting::setSettings to update settings.
Actual Result	CompanySetting::setSettings used for updating company settings.
Status	Pass
Severity	High

Test Case ID	UC-008
Title	Instantiate UpdateSettingsController
Objective	Verify that UpdateSettingsController can be successfully instantiated.
Preconditions	- Application running - Classes loaded
Test Steps	1. Attempt to create a new instance of UpdateSettingsController. 2. Confirm the instance is of the correct class.
Test Data	- Controller class: UpdateSettingsController
Expected Result	- A new object of type UpdateSettingsController is instantiated.
Actual Result	UpdateSettingsController was successfully instantiated.
Status	Pass
Severity	Medium

Test Case ID	UC-009
Title	UpdateSettingsController extends base Controller
Objective	Ensure UpdateSettingsController is a subclass of the core Controller class.
Preconditions	- Application running - Classes loaded
Test Steps	1. Instantiate UpdateSettingsController. 2. Check if the instance is a subclass of \Crater\Http\Controllers\Controller.
Test Data	- Expected parent class: \Crater\Http\Controllers\Controller
Expected Result	- Instance is verified to extend the Controller base class.
Actual Result	UpdateSettingsController is confirmed to extend Controller.
Status	Pass
Severity	Medium

Test Case ID	UC-010
Title	UpdateSettingsController is invokable
Objective	Verify the presence of the __invoke method in UpdateSettingsController.
Preconditions	- Application running - Classes loaded
Test Steps	1. Create a ReflectionClass for UpdateSettingsController. 2. Check if the class defines the '__invoke' method.
Test Data	- Method expected: __invoke
Expected Result	- '__invoke' method is present in the controller.

Actual Result	__invoke method exists in UpdateSettingsController.
Status	Pass
Severity	Medium

Test Case ID	UC-011
Title	Manage settings authorization in UpdateSettingsController
Objective	Ensure authorization is used in UpdateSettingsController for managing settings.
Preconditions	- Application running - Source code available
Test Steps	1. Read UpdateSettingsController source file. 2. Search for '$this->authorize('manage settings')'.
Test Data	- Authorization string: 'manage settings'
Expected Result	- Authorization code for managing settings is present.
Actual Result	Authorization code for managing settings was found.
Status	Pass
Severity	High

Test Case ID	UC-012
Title	Usage of Setting::setSettings in UpdateSettingsController
Objective	Confirm that Setting::setSettings is used to update settings in the controller.
Preconditions	- Application running - Source code available
Test Steps	1. Read UpdateSettingsController source code. 2. Check for use of 'Setting::setSettings($request->settings)'.
Test Data	- Function call: Setting::setSettings($request->settings)
Expected Result	- Controller uses Setting::setSettings to update application settings.
Actual Result	Setting::setSettings is used for updating settings.
Status	Pass
Severity	High

Test Case ID	UC-013
Title	Success response with settings in UpdateSettingsController
Objective	Verify that controller returns a success response including updated settings.
Preconditions	- Application running - Source code available
Test Steps	1. Read UpdateSettingsController source code. 2. Confirm 'success' => true is in the response. 3. Confirm response includes '$request->settings'.
Test Data	- Response structure: ['success' => true, 'settings' => requested data]
Expected Result	- The controller returns a success response with the settings.
Actual Result	Success response with updated settings returned.
Status	Pass
Severity	High

Test Case ID	UC-014
Title	Instantiate UpdateController

Objective	Verify that UpdateController can be successfully instantiated.
Preconditions	- Application running - Classes loaded
Test Steps	1. Attempt to create a new instance of UpdateController. 2. Confirm the instance is of the correct class.
Test Data	- Controller class: UpdateController
Expected Result	- A new object of type UpdateController is instantiated.
Actual Result	UpdateController was successfully instantiated.
Status	Pass
Severity	Medium

Test Case ID	UC-015
Title	UpdateController extends base Controller
Objective	Ensure UpdateController inherits from the core Controller class.
Preconditions	- Application running - Classes loaded
Test Steps	1. Instantiate UpdateController. 2. Check if the instance is a subclass of \Crater\Http\Controllers\Controller.
Test Data	- Expected parent class: \Crater\Http\Controllers\Controller
Expected Result	- Instance is verified to extend the Controller base class.
Actual Result	UpdateController is confirmed to extend Controller.
Status	Pass
Severity	Medium

Test Case ID	UC-016
Title	UpdateController defines all update methods
Objective	Verify all required update methods exist in UpdateController.
Preconditions	- Application running - Source code available
Test Steps	1. Create a ReflectionClass for UpdateController. 2. Check for existence of methods: - download - unzip - copyFiles - migrate - finishUpdate - checkLatestVersion
Test Data	- Method names to be validated
Expected Result	- All specified update methods exist in UpdateController.
Actual Result	All update methods are present in UpdateController.
Status	Pass
Severity	High

Test Case ID	UC-017
Title	Manage update app authorization in UpdateController
Objective	Ensure that UpdateController enforces manage update app authorization on all methods.
Preconditions	- Application running - Source code available

Test Steps	1. Access UpdateController source code. 2. Search for '$this->authorize('manage update app')' in the methods.
Test Data	- Authorization string: 'manage update app'
Expected Result	- Each method includes manage update app authorization check.
Actual Result	Authorization for manage update app is used in all methods.
Status	Pass
Severity	High

Test Case ID	UC-018
Title	UpdateController uses Updater::download in download method
Objective	Verify Updater::download is called with the correct version in download method.
Preconditions	- Application running - Source code available - Valid version parameter
Test Steps	1. Read UpdateController source code. 2. Check for 'Updater::download($request->version)' in download method.
Test Data	- Version input: $request->version
Expected Result	- Updater::download is called using the provided version.
Actual Result	Updater::download called with the specified version in download method.
Status	Pass
Severity	High

Test Case ID	UC-019
Title	Unzip method in UpdateController handles exceptions correctly
Objective	Validate that unzip method uses try-catch and exception messages are handled.
Preconditions	- Application running - Source code available
Test Steps	1. Access the unzip method in UpdateController source code. 2. Check for use of try block. 3. Confirm Updater::unzip is called. 4. Verify catch block for Exception and message handling.
Test Data	- Path input: $request->path - Exception handling code block
Expected Result	- Updater::unzip is wrapped in try-catch and exceptions are handled with message.
Actual Result	Unzip method uses try-catch, exceptions are properly handled.
Status	Pass
Severity	High

Test Case ID	UC-020
Title	checkLatestVersion sets time limit and uses Updater
Objective	Ensure checkLatestVersion method sets time limit and calls Updater::checkForUpdate with current version.
Preconditions	- Application running - Source code available
Test Steps	1. Access checkLatestVersion method in UpdateController source code. 2. Confirm presence of 'set_time_limit(600)'. 3. Verify call to 'Updater::checkForUpdate(Setting::getSetting('version'))'.

Test Data	- Time limit value: 600 seconds - Current version: Setting::getSetting('version')
Expected Result	- checkLatestVersion sets script time limit and checks for update correctly.
Actual Result	Script time limit set and update check performed with current version.
Status	Pass
Severity	High

File: UpdateFinished-Test.txt

Test Case ID	UF-007
Title	Verify UpdateFinished Event Uses Dispatchable Trait
Objective	Ensure that the UpdateFinished event class uses the Dispatchable trait for event dispatching.
Preconditions	- Application running - Database seeded - \Crater\Events\UpdateFinished class available - \Illuminate\Foundation\Events\Dispatchable trait available
Test Steps	1. Retrieve traits used by \Crater\Events\UpdateFinished class. 2. Check that Dispatchable trait is among the used traits.
Test Data	- Input: \Crater\Events\UpdateFinished class traits - Expected: Traits list contains \Illuminate\Foundation\Events\Dispatchable
Expected Result	- \Illuminate\Foundation\Events\Dispatchable trait is present in the used traits of UpdateFinished event class.
Actual Result	Dispatchable trait is correctly used in UpdateFinished event class.
Status	Pass
Severity	High ========== ==========

File: UpdateSettingsRequest-Test.txt

Test Case ID	USR-001
Title	Authorize Method Returns True in UpdateSettingsRequest
Objective	Verify that the authorize() method of UpdateSettingsRequest always returns true
Preconditions	- Application running - UpdateSettingsRequest class available - No authentication or permission restrictions active - Database seeded if required (for framework bootstrapping)
Test Steps	1. Instantiate an object of UpdateSettingsRequest 2. Call the authorize() method on the object 3. Check the returned value
Test Data	- Instance of UpdateSettingsRequest - No specific input data required
Expected Result	- The authorize() method should return true
Actual Result	The authorize() method returns true
Status	Pass
Severity	High

Test Case ID	USR-002
Title	Rules Method Returns Validation Rules for Settings
Objective	Verify that rules() method of UpdateSettingsRequest returns the correct validation rules for settings
Preconditions	- Application running - UpdateSettingsRequest class available - No interfering configurations or overrides - Database seeded if required (for framework bootstrapping)
Test Steps	1. Instantiate an object of UpdateSettingsRequest 2. Call the rules() method on the object 3. Retrieve the returned array 4. Compare the returned array to the expected rules
Test Data	- Instance of UpdateSettingsRequest - Expected rules: - 'settings' => ['required']
Expected Result	- The rules() method should return an array: - 'settings' => ['required']
Actual Result	The rules() method returns the array: - 'settings' => ['required']
Status	Pass
Severity	High

File: UpdateUserSettingsController-Test.txt

Test Case ID	UUSC-001
Title	Successful update of user settings with valid data
Objective	Verify that the UpdateUserSettingsController correctly updates user settings when provided with valid data.
Preconditions	- Application running - Database seeded - User is authenticated and exists - Valid user settings data available
Test Steps	1. Mock a User model and configure it to expect a call to setSettings with ['theme' => 'dark', 'notifications' => true]. 2. Mock an UpdateSettingsRequest object and set its user() method to return the mocked User. 3. Set the 'settings' property on the request to ['theme' => 'dark', 'notifications' => true]. 4. Instantiate UpdateUserSettingsController. 5. Invoke the controller’s __invoke method with the mock request. 6. Assert that the response is a JsonResponse instance. 7. Assert that the response status code is 200. 8. Assert that the response data is ['success' => true].
Test Data	- Input: User settings ['theme' => 'dark', 'notifications' => true] - Expected values: Response type JsonResponse; Status code 200; Data ['success' => true]
Expected Result	- The controller calls setSettings on the user with the provided settings. - The endpoint returns a JsonResponse with status code 200. - The response data contains ['success' => true].
Actual Result	- The controller successfully updates the settings and returns JsonResponse with status 200 and ['success' => true].
Status	Pass
Severity	High

Test Case ID	UUSC-002
Title	Successful update of user settings with empty settings array
Objective	Verify that the UpdateUserSettingsController handles an empty settings array correctly and updates without errors.
Preconditions	- Application running - Database seeded - User is authenticated and exists
Test Steps	1. Mock a User model and configure it to expect a call to setSettings with an empty array. 2. Mock an UpdateSettingsRequest object and set its user() method to return the mocked User. 3. Set the 'settings' property on the request to an empty array. 4. Instantiate UpdateUserSettingsController. 5. Invoke the controller’s __invoke method with the mock request. 6. Assert that the response is a JsonResponse instance. 7. Assert that the response status code is 200. 8. Assert that the response data is ['success' => true].
Test Data	- Input: User settings [] - Expected values: Response type JsonResponse; Status code 200; Data ['success' => true]
Expected Result	- The controller calls setSettings on the user with an empty array. - The endpoint returns a JsonResponse with status code 200. - The response data contains ['success' => true].
Actual Result	- The controller successfully updates the settings with an empty array and returns JsonResponse with status 200 and ['success' => true].

Status	Pass
Severity	High

Test Case ID	UUSC-003
Title	TypeError thrown when user is null in request
Objective	Verify that the UpdateUserSettingsController throws an Error (TypeError) when the user is not present in the UpdateSettingsRequest.
Preconditions	- Application running - Database seeded - UpdateSettingsRequest object available - User is not authenticated (user() returns null)
Test Steps	1. Mock an UpdateSettingsRequest object and set its user() method to return null. 2. Set the 'settings' property on the request to ['theme' => 'dark']. 3. Instantiate UpdateUserSettingsController. 4. Invoke the controller's __invoke method with the mock request. 5. Assert that an Error (TypeError) is thrown. 6. Assert that the exception message matches either: - "Attempt to call a method named setSettings on null" - "Call to a member function setSettings() on null"
Test Data	- Input: User settings ['theme' => 'dark'], user() returns null - Expected exception messages: "Attempt to call a method named setSettings on null" OR "Call to a member function setSettings() on null"
Expected Result	- The controller throws an Error (TypeError). - The exception message matches the expected pattern for calling setSettings on null.
Actual Result	- The controller throws an Error (TypeError) as expected with a matching message.
Status	Pass
Severity	High

File: UploadModuleRequest-Test.txt

Test Case ID	UMR-001
Title	Verify authorize method always returns true
Objective	Ensure that the UploadModuleRequest's authorize() method consistently returns true, allowing all users to upload modules.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\UploadModuleRequest class is available
Test Steps	1. Instantiate a new UploadModuleRequest object. 2. Call the authorize() method. 3. Verify that the result is true.
Test Data	- UploadModuleRequest instance with default state.
Expected Result	- authorize() method returns true.
Actual Result	authorize() method returned true as expected.
Status	Pass
Severity	High

Test Case ID	UMR-002
Title	Validate correct validation rules returned by rules() method
Objective	Confirm that the rules() method of UploadModuleRequest returns correct validation rules for avatar and module fields.
Preconditions	- Application running - Database seeded - Crater\Http\Requests\UploadModuleRequest class is available
Test Steps	1. Instantiate a new UploadModuleRequest object. 2. Call the rules() method to retrieve validation rules. 3. Compare returned rules against the expected validation rules.
Test Data	- UploadModuleRequest instance with default state. - Expected rules: - 'avatar': required, file, mimes:zip, max:20000 - 'module': required, string, max:100
Expected Result	- rules() method returns: ['avatar' => ['required', 'file', 'mimes:zip', 'max:20000'], 'module' => ['required', 'string', 'max:100']]
Actual Result	rules() method returned the expected validation rules.
Status	Pass
Severity	High

File: UploadReceiptRequest-Test.txt

Test Case ID	URR-001
Title	Instantiation of UploadExpenseReceiptRequest
Objective	Verify that the UploadExpenseReceiptRequest class can be instantiated properly.
Preconditions	- Application running - Necessary PHP dependencies are installed - No interfering namespace or class issues
Test Steps	1. Attempt to instantiate the UploadExpenseReceiptRequest class. 2. Check the object type.
Test Data	- Class: UploadExpenseReceiptRequest
Expected Result	- An object of type UploadExpenseReceiptRequest is created.
Actual Result	An object of type UploadExpenseReceiptRequest was successfully instantiated.
Status	Pass
Severity	Medium

Test Case ID	URR-002
Title	UploadExpenseReceiptRequest Extends FormRequest
Objective	Verify that UploadExpenseReceiptRequest inherits from the FormRequest base class.
Preconditions	- Application running - UploadExpenseReceiptRequest and FormRequest classes exist
Test Steps	1. Instantiate UploadExpenseReceiptRequest. 2. Verify that the created object is an instance of Illuminate\Foundation\Http\FormRequest.
Test Data	- Class: UploadExpenseReceiptRequest
Expected Result	- UploadExpenseReceiptRequest object is also an instance of FormRequest.
Actual Result	UploadExpenseReceiptRequest was correctly identified as an instance of FormRequest.
Status	Pass
Severity	Medium

Test Case ID	URR-003
Title	Correct Namespace for UploadExpenseReceiptRequest
Objective	Ensure that UploadExpenseReceiptRequest is located in the Crater\Http\Requests namespace.
Preconditions	- Application running - UploadExpenseReceiptRequest class exists
Test Steps	1. Get reflection of UploadExpenseReceiptRequest. 2. Retrieve namespace name of the class.
Test Data	- Target class: UploadExpenseReceiptRequest
Expected Result	- Namespace is "Crater\Http\Requests".
Actual Result	Namespace identified as "Crater\Http\Requests".
Status	Pass
Severity	Low

Test Case ID	URR-004

Title	UploadExpenseReceiptRequest Authorization Always Returns True
Objective	Verify that the authorize() method in UploadExpenseReceiptRequest returns true.
Preconditions	- Application running - UploadExpenseReceiptRequest class implemented
Test Steps	1. Instantiate UploadExpenseReceiptRequest. 2. Call the authorize() method. 3. Capture the returned value.
Test Data	- Method called: authorize()
Expected Result	- The authorize() method returns true.
Actual Result	authorize() returned true.
Status	Pass
Severity	High

Test Case ID	URR-005
Title	Validation Rules Include attachment_receipt for UploadExpenseReceiptRequest
Objective	Confirm rules() defines validation for attachment_receipt and includes "nullable".
Preconditions	- Application running - UploadExpenseReceiptRequest class with rules() method
Test Steps	1. Instantiate UploadExpenseReceiptRequest. 2. Call rules() to retrieve validation array. 3. Verify rules array includes "attachment_receipt" key. 4. Ensure attachment_receipt rule array contains "nullable".
Test Data	- Method called: rules()
Expected Result	- "attachment_receipt" key exists in rules. - Its value is an array including "nullable".
Actual Result	Rules array includes "attachment_receipt" key and contains "nullable" in its rules.
Status	Pass
Severity	High

Test Case ID	URR-006
Title	Uses Base64Mime Rule for Image Types in UploadExpenseReceiptRequest
Objective	Verify presence of Base64Mime rule for gif, jpg, png in validation logic.
Preconditions	- Application running - UploadExpenseReceiptRequest source code accessible
Test Steps	1. Obtain file contents of UploadExpenseReceiptRequest. 2. Search for instantiation of Base64Mime rule with ['gif', 'jpg', 'png'].
Test Data	- Source code: UploadExpenseReceiptRequest
Expected Result	- Code contains "new Base64Mime(['gif', 'jpg', 'png'])".
Actual Result	File contents verified to contain "new Base64Mime(['gif', 'jpg', 'png'])".
Status	Pass
Severity	High

Test Case ID	URR-007
Title	Instantiation of UploadReceiptController
Objective	Verify UploadReceiptController can be instantiated.

Preconditions	- Application running - UploadReceiptController class exists
Test Steps	1. Attempt to instantiate UploadReceiptController. 2. Check resulting object type.
Test Data	- Class: UploadReceiptController
Expected Result	- Object is instance of UploadReceiptController.
Actual Result	Successfully instantiated UploadReceiptController.
Status	Pass
Severity	Medium

Test Case ID	URR-008
Title	UploadReceiptController Extends Base Controller
Objective	Check that UploadReceiptController inherits from Crater\Http\Controllers\Controller.
Preconditions	- Application running - UploadReceiptController & base Controller classes exist
Test Steps	1. Instantiate UploadReceiptController. 2. Assert object is instance of Crater\Http\Controllers\Controller.
Test Data	- Class: UploadReceiptController
Expected Result	- Object is an instance of Controller.
Actual Result	UploadReceiptController confirmed to extend Controller.
Status	Pass
Severity	Medium

Test Case ID	URR-009
Title	UploadReceiptController is Invokable
Objective	Ensure UploadReceiptController implements __invoke method.
Preconditions	- Application running - UploadReceiptController class defined
Test Steps	1. Get reflection of UploadReceiptController. 2. Check if class has __invoke method.
Test Data	- Class: UploadReceiptController
Expected Result	- __invoke method present in UploadReceiptController.
Actual Result	__invoke method found in UploadReceiptController.
Status	Pass
Severity	Medium

Test Case ID	URR-010
Title	UploadReceiptController Uses Authorization Check
Objective	Verify UploadReceiptController uses $this->authorize on update for expense.
Preconditions	- Application running - UploadReceiptController source code accessible
Test Steps	1. Obtain file contents of UploadReceiptController. 2. Search for "$this->authorize('update', $expense)" string.
Test Data	- Controller method code
Expected Result	- Code contains "$this->authorize('update', $expense)".
Actual Result	Authorization code verified as present.

Status	Pass
Severity	High

Test Case ID	URR-011
Title	Media Collection Cleared on Edit in UploadReceiptController
Objective	Confirm media "receipts" are cleared when request type is "edit".
Preconditions	- Application running - UploadReceiptController source code accessible
Test Steps	1. Obtain file contents of UploadReceiptController. 2. Search for "if (\$request->type === 'edit')" in code. 3. Check for "\$expense->clearMediaCollection('receipts')" within edit condition.
Test Data	- Controller method logic
Expected Result	- Code conditionally clears receipts media when editing.
Actual Result	Media clear logic found for 'edit' request type.
Status	Pass
Severity	High

Test Case ID	URR-012
Title	UploadReceiptController Adds Media from Base64
Objective	Ensure controller adds base64 media as receipt with proper file naming.
Preconditions	- Application running - UploadReceiptController source code accessible
Test Steps	1. Obtain file contents of UploadReceiptController. 2. Verify presence of code adding media from base64 and specifying file name. 3. Confirm media is added to "receipts" collection.
Test Data	- Controller method logic for base64 media
Expected Result	- Contains code: $expense->addMediaFromBase64($data->data) - Uses file name: ->usingFileName($data->name) - Adds to: ->toMediaCollection('receipts')
Actual Result	All expected code instructions for adding base64 media found and correct.
Status	Pass
Severity	High

File: UserPolicy-Test.txt

Test Case ID	UP-001
Title	Owner can view any user
Objective	Verify that an owner is authorized to view any user in the system.
Preconditions	- Application running - Database seeded - Owner user exists
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Call the viewAny method of UserPolicy with the mocked owner user. 3. Check the returned value.
Test Data	- Input: User with isOwner = true - Expected: viewAny() returns true
Expected Result	viewAny() method returns true.
Actual Result	viewAny() method returned true.
Status	Pass
Severity	High

Test Case ID	UP-002
Title	Non-owner cannot view any user
Objective	Verify that a non-owner is not authorized to view any user.
Preconditions	- Application running - Database seeded - Non-owner user exists
Test Steps	1. Mock a User instance and set isOwner property to false. 2. Call the viewAny method of UserPolicy with the mocked non-owner user. 3. Check the returned value.
Test Data	- Input: User with isOwner = false - Expected: viewAny() returns false
Expected Result	viewAny() method returns false.
Actual Result	viewAny() method returned false.
Status	Pass
Severity	High

Test Case ID	UP-003
Title	Owner can view a specific user
Objective	Verify that an owner can view details of a specific user.
Preconditions	- Application running - Database seeded - Owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Mock another User instance as the target user. 3. Call the view method of UserPolicy with the owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = true, Target User - Expected: view() returns true
Expected Result	view() method returns true.
Actual Result	view() method returned true.
Status	Pass

Severity	High

Test Case ID	UP-004
Title	Non-owner cannot view a specific user
Objective	Verify that a non-owner cannot view details of a specific user.
Preconditions	- Application running - Database seeded - Non-owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to false. 2. Mock another User instance as the target user. 3. Call the view method of UserPolicy with the non-owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = false, Target User - Expected: view() returns false
Expected Result	view() method returns false.
Actual Result	view() method returned false.
Status	Pass
Severity	High

Test Case ID	UP-005
Title	Owner can create users
Objective	Verify that an owner can create user accounts.
Preconditions	- Application running - Database seeded - Owner user exists
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Call the create method of UserPolicy with the owner user. 3. Check the returned value.
Test Data	- Input: User with isOwner = true - Expected: create() returns true
Expected Result	create() method returns true.
Actual Result	create() method returned true.
Status	Pass
Severity	High

Test Case ID	UP-006
Title	Non-owner cannot create users
Objective	Verify that a non-owner cannot create user accounts.
Preconditions	- Application running - Database seeded - Non-owner user exists
Test Steps	1. Mock a User instance and set isOwner property to false. 2. Call the create method of UserPolicy with the non-owner user. 3. Check the returned value.
Test Data	- Input: User with isOwner = false - Expected: create() returns false
Expected Result	create() method returns false.
Actual Result	create() method returned false.
Status	Pass

Severity	High

Test Case ID	UP-007
Title	Owner can update a user
Objective	Verify that an owner is authorized to update a user’s information.
Preconditions	- Application running - Database seeded - Owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Mock another User instance as the target user. 3. Call the update method of UserPolicy with the owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = true, Target User - Expected: update() returns true
Expected Result	update() method returns true.
Actual Result	update() method returned true.
Status	Pass
Severity	High

Test Case ID	UP-008
Title	Non-owner cannot update a user
Objective	Verify that a non-owner cannot update a user’s information.
Preconditions	- Application running - Database seeded - Non-owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to false. 2. Mock another User instance as the target user. 3. Call the update method of UserPolicy with the non-owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = false, Target User - Expected: update() returns false
Expected Result	update() method returns false.
Actual Result	update() method returned false.
Status	Pass
Severity	High

Test Case ID	UP-009
Title	Owner can delete a user
Objective	Verify that an owner can delete user accounts.
Preconditions	- Application running - Database seeded - Owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Mock another User instance as the target user. 3. Call the delete method of UserPolicy with the owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = true, Target User - Expected: delete() returns true

Expected Result	delete() method returns true.
Actual Result	delete() method returned true.
Status	Pass
Severity	High

Test Case ID	UP-010
Title	Non-owner cannot delete a user
Objective	Verify that a non-owner cannot delete user accounts.
Preconditions	- Application running - Database seeded - Non-owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to false. 2. Mock another User instance as the target user. 3. Call the delete method of UserPolicy with the non-owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = false, Target User - Expected: delete() returns false
Expected Result	delete() method returns false.
Actual Result	delete() method returned false.
Status	Pass
Severity	High

Test Case ID	UP-011
Title	Owner can restore a user
Objective	Verify that an owner is authorized to restore a deleted user.
Preconditions	- Application running - Database seeded - Owner user exists - Deleted user exists
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Mock another User instance as the deleted target user. 3. Call the restore method of UserPolicy with the owner and deleted user. 4. Check the returned value.
Test Data	- Input: User with isOwner = true, Deleted User - Expected: restore() returns true
Expected Result	restore() method returns true.
Actual Result	restore() method returned true.
Status	Pass
Severity	High

Test Case ID	UP-012
Title	Non-owner cannot restore a user
Objective	Verify that a non-owner cannot restore deleted users.
Preconditions	- Application running - Database seeded - Non-owner user exists - Deleted user exists

Test Steps	1. Mock a User instance and set isOwner property to false. 2. Mock another User instance as the deleted target user. 3. Call the restore method of UserPolicy with the non-owner and deleted user. 4. Check the returned value.
Test Data	- Input: User with isOwner = false, Deleted User - Expected: restore() returns false
Expected Result	restore() method returns false.
Actual Result	restore() method returned false.
Status	Pass
Severity	High

Test Case ID	UP-013
Title	Owner can force delete a user
Objective	Verify that an owner is authorized to force delete a user account.
Preconditions	- Application running - Database seeded - Owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Mock another User instance as the target user. 3. Call the forceDelete method of UserPolicy with the owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = true, Target User - Expected: forceDelete() returns true
Expected Result	forceDelete() method returns true.
Actual Result	forceDelete() method returned true.
Status	Pass
Severity	High

Test Case ID	UP-014
Title	Non-owner cannot force delete a user
Objective	Verify that a non-owner cannot force delete user accounts.
Preconditions	- Application running - Database seeded - Non-owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to false. 2. Mock another User instance as the target user. 3. Call the forceDelete method of UserPolicy with the non-owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = false, Target User - Expected: forceDelete() returns false
Expected Result	forceDelete() method returns false.
Actual Result	forceDelete() method returned false.
Status	Pass
Severity	High

Test Case ID	UP-015
Title	Owner can invite a user
Objective	Verify that an owner is authorized to invite users.

Preconditions	- Application running - Database seeded - Owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Mock another User instance as the invitee. 3. Call the invite method of UserPolicy with the owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = true, Invitee User - Expected: invite() returns true
Expected Result	invite() method returns true.
Actual Result	invite() method returned true.
Status	Pass
Severity	High

Test Case ID	UP-016
Title	Non-owner cannot invite a user
Objective	Verify that a non-owner cannot invite users.
Preconditions	- Application running - Database seeded - Non-owner user exists - Target user exists
Test Steps	1. Mock a User instance and set isOwner property to false. 2. Mock another User instance as the invitee. 3. Call the invite method of UserPolicy with the non-owner and target user. 4. Check the returned value.
Test Data	- Input: User with isOwner = false, Invitee User - Expected: invite() returns false
Expected Result	invite() method returns false.
Actual Result	invite() method returned false.
Status	Pass
Severity	High

Test Case ID	UP-017
Title	Owner can delete multiple users
Objective	Verify that an owner is authorized to delete multiple users at once.
Preconditions	- Application running - Database seeded - Owner user exists - Multiple users exist
Test Steps	1. Mock a User instance and set isOwner property to true. 2. Call the deleteMultiple method of UserPolicy with the owner user. 3. Check the returned value.
Test Data	- Input: User with isOwner = true - Expected: deleteMultiple() returns true
Expected Result	deleteMultiple() method returns true.
Actual Result	deleteMultiple() method returned true.
Status	Pass
Severity	High

Test Case ID	UP-018

Title	Non-owner cannot delete multiple users
Objective	Verify that a non-owner cannot delete multiple users at once.
Preconditions	- Application running - Database seeded - Non-owner user exists - Multiple users exist
Test Steps	1. Mock a User instance and set isOwner property to false. 2. Call the deleteMultiple method of UserPolicy with the non-owner user. 3. Check the returned value.
Test Data	- Input: User with isOwner = false - Expected: deleteMultiple() returns false
Expected Result	deleteMultiple() method returns false.
Actual Result	deleteMultiple() method returned false.
Status	Pass
Severity	High

File: UserRelated-Test.txt

Test Case ID	UR-001
Title	UserCollection extends ResourceCollection
Objective	Verify that the UserCollection class extends the Laravel ResourceCollection class.
Preconditions	- Application running - All dependencies installed - No specific data required
Test Steps	1. Instantiate a new UserCollection with an empty Collection. 2. Check the class type of the instantiated UserCollection.
Test Data	- Input: New UserCollection (Collection([])) - Expected class: Illuminate\Http\Resources\Json\ResourceCollection
Expected Result	- UserCollection instance is of type Illuminate\Http\Resources\Json\ResourceCollection.
Actual Result	- UserCollection instance was verified to be of the correct type.
Status	Pass
Severity	Medium

Test Case ID	UR-002
Title	UserCollection is in correct namespace
Objective	Ensure the UserCollection class resides in the Crater\Http\Resources namespace.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass to inspect the UserCollection class. 2. Retrieve the namespace name. 3. Compare with the expected value.
Test Data	- Input: UserCollection class - Expected namespace: Crater\Http\Resources
Expected Result	- Namespace is Crater\Http\Resources.
Actual Result	- Namespace verified as Crater\Http\Resources.
Status	Pass
Severity	Low

Test Case ID	UR-003
Title	UserCollection toArray returns empty array for empty collection
Objective	Validate that toArray returns an empty array when the UserCollection is empty.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Create a new Request instance. 2. Instantiate UserCollection with an empty Collection. 3. Call toArray with the Request. 4. Check that the result is an empty array.
Test Data	- Input: UserCollection (Collection([])), Request() - Expected: []
Expected Result	- The toArray result is an empty array.
Actual Result	- Returned result was an empty array as expected.
Status	Pass
Severity	Low

Test Case ID	UR-004
Title	UserCollection delegates to parent toArray
Objective	Check that UserCollection's toArray method returns an array, indicating delegation to parent.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Create a new Request instance. 2. Instantiate UserCollection with an empty Collection. 3. Call toArray with the Request. 4. Verify the result is an array.
Test Data	- Input: UserCollection (Collection([])), Request() - Expected: Array output
Expected Result	- toArray returns an array.
Actual Result	- toArray result was an array.
Status	Pass
Severity	Low

Test Case ID	UR-005
Title	UserRequest extends FormRequest
Objective	Verify that the UserRequest class extends Laravel’s FormRequest class.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Instantiate a new UserRequest. 2. Check the class type of the instance.
Test Data	- Input: New UserRequest instance - Expected class: Illuminate\Foundation\Http\FormRequest
Expected Result	- UserRequest instance is of type Illuminate\Foundation\Http\FormRequest.
Actual Result	- UserRequest verified as extending FormRequest.
Status	Pass
Severity	Medium

Test Case ID	UR-006
Title	UserRequest is in correct namespace
Objective	Ensure the UserRequest class resides in the Crater\Http\Requests namespace.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass to inspect the UserRequest class. 2. Retrieve the namespace name. 3. Compare with expected value.
Test Data	- Input: UserRequest class - Expected namespace: Crater\Http\Requests
Expected Result	- Namespace is Crater\Http\Requests.
Actual Result	- Namespace verified as Crater\Http\Requests.
Status	Pass
Severity	Low

Test Case ID	UR-007

Title	UserRequest authorize returns true
Objective	Validate that the authorize method of UserRequest always returns true.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Instantiate a new UserRequest. 2. Call the authorize() method. 3. Check that it returns true.
Test Data	- Input: UserRequest instance - Method: authorize() - Expected: true
Expected Result	- authorize() returns true.
Actual Result	- authorize() returned true.
Status	Pass
Severity	High

Test Case ID	UR-008
Title	UserRequest rules includes all required fields
Objective	Verify that validation rules in UserRequest include all required fields and constraints.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Instantiate a new UserRequest. 2. Set the ‘company’ header to ‘123’. 3. Call the rules() method. 4. Verify rules array contains ‘name’, ‘email’, ‘phone’, ‘password’, and ‘companies’ keys. 5. Check that 'name' and 'email' rules include 'required'. 6. Check that 'email' includes 'email' validation.
Test Data	- Header: company=123 - Expected rules: 'name', 'email', 'phone', 'password', 'companies' - Expected validations: required (name, email), email (email)
Expected Result	1. Rules array has all required keys. 2. 'name' and 'email' are required. 3. 'email' field has email validation.
Actual Result	- All required keys and validations are present in rules.
Status	Pass
Severity	High

Test Case ID	UR-009
Title	UserRequest rules handles PUT method differently
Objective	Ensure that validation logic in UserRequest handles PUT HTTP method by ignoring the current user for uniqueness and making password nullable.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass on UserRequest. 2. Read the class file contents. 3. Check for conditional logic specific to PUT method. 4. Verify unique rule ignores current user. 5. Verify password field can be nullable.
Test Data	- Request method: PUT - Expected: Unique rule with ignore, password nullable

Expected Result	- File contains code: if ($this->getMethod() == 'PUT') - Unique validation ignores current user - Password field is nullable
Actual Result	- All conditions satisfied in file.
Status	Pass
Severity	High

Test Case ID	UR-010
Title	UserRequest getUserPayload merges creator_id
Objective	Verify that getUserPayload method in UserRequest merges creator_id with user ID.
Preconditions	- Application running - User authenticated - All dependencies installed
Test Steps	1. Use ReflectionClass on UserRequest. 2. Read class file contents. 3. Check that 'creator_id' => $this->user()->id is present.
Test Data	- UserRequest class file - Expected payload: includes 'creator_id' => $this->user()->id
Expected Result	- getUserPayload merges creator_id into payload.
Actual Result	- creator_id merging logic verified.
Status	Pass
Severity	High

Test Case ID	UR-011
Title	UsersController extends Controller
Objective	Verify that UsersController extends the base Controller class.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Instantiate UsersController. 2. Check class type against Crater\Http\Controllers\Controller.
Test Data	- Input: UsersController - Expected class: Crater\Http\Controllers\Controller
Expected Result	- UsersController is instance of Controller.
Actual Result	- UsersController verified as extending Controller.
Status	Pass
Severity	Medium

Test Case ID	UR-012
Title	UsersController is in correct namespace
Objective	Ensure UsersController is defined in Crater\Http\Controllers\V1\Admin\Users namespace.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass on UsersController. 2. Retrieve and verify the namespace name.
Test Data	- Input: UsersController - Expected namespace: Crater\Http\Controllers\V1\Admin\Users
Expected Result	- Namespace is Crater\Http\Controllers\V1\Admin\Users.

Actual Result	- Namespace was as expected.
Status	Pass
Severity	Low

Test Case ID	UR-013
Title	UsersController has all CRUD methods
Objective	Ensure UsersController implements standard CRUD methods: index, store, show, update, and delete.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass on UsersController. 2. Check for methods: index, store, show, update, delete.
Test Data	- Input: UsersController - Methods to validate: index, store, show, update, delete
Expected Result	- Each CRUD method is present in UsersController class.
Actual Result	- All CRUD methods present.
Status	Pass
Severity	High

Test Case ID	UR-014
Title	UsersController index uses authorization and pagination
Objective	Validate that the index method checks authorization and uses pagination with filtering.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass on UsersController. 2. Read file contents. 3. Verify presence of: - $this->authorize('viewAny', User::class) - User::applyFilters($request->all()) - ->paginate($limit)
Test Data	- File: UsersController class file - Expected code blocks: authorization, filters, pagination
Expected Result	1. Authorization for viewing users is implemented. 2. Filtering and pagination are used in index.
Actual Result	- Authorization, filtering, pagination verified.
Status	Pass
Severity	High

Test Case ID	UR-015
Title	UsersController store uses User createFromRequest
Objective	Confirm that the store method uses User::createFromRequest and checks authorization.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass on UsersController. 2. Read file contents. 3. Verify: - $this->authorize('create', User::class) - User::createFromRequest($request)

Test Data	- File: UsersController class file - Expected: Authorization and User::createFromRequest call
Expected Result	1. store method checks authorization. 2. store method creates user from request.
Actual Result	- Authorization and User::createFromRequest call found.
Status	Pass
Severity	High

Test Case ID	UR-016
Title	UsersController update uses updateFromRequest
Objective	Verify that the update method uses updateFromRequest and authorizes the update.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass on UsersController. 2. Read file contents. 3. Verify: - $this->authorize('update', $user) - $user->updateFromRequest($request)
Test Data	- File: UsersController class file - Expected: Authorization and updateFromRequest call
Expected Result	1. Update method checks authorization. 2. Update method updates user from request data.
Actual Result	- Authorization and updateFromRequest found.
Status	Pass
Severity	High

Test Case ID	UR-017
Title	UsersController delete uses deleteUsers method
Objective	Check that the delete method uses User::deleteUsers and authorizes the operation.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass on UsersController. 2. Read file contents. 3. Verify: - $this->authorize('delete multiple users', User::class) - User::deleteUsers($request->users)
Test Data	- File: UsersController class file - Expected: Authorization and deleteUsers call
Expected Result	1. delete method authorizes operation. 2. delete method deletes multiple users using User::deleteUsers.
Actual Result	- Authorization and deleteUsers method found.
Status	Pass
Severity	High

Test Case ID	UR-018
Title	UserSetting extends Model and uses HasFactory
Objective	Verify that UserSetting extends Model and utilizes the HasFactory trait.
Preconditions	- Application running - All dependencies installed

Test Steps	1. Instantiate UserSetting. 2. Use ReflectionClass to check parent class and traits. 3. Verify parent class is Model. 4. Verify HasFactory trait is used.
Test Data	- Input: UserSetting - Expected: Parent = Model, Traits = HasFactory
Expected Result	1. UserSetting is a subclass of Model. 2. UserSetting uses HasFactory trait.
Actual Result	- Both parent class and trait verified.
Status	Pass
Severity	Medium

Test Case ID	UR-019
Title	UserSetting is in correct namespace
Objective	Ensure that UserSetting is in the Crater\Models namespace.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Use ReflectionClass on UserSetting. 2. Retrieve and check namespace name.
Test Data	- Input: UserSetting - Expected namespace: Crater\Models
Expected Result	- Namespace is Crater\Models.
Actual Result	- Namespace verified.
Status	Pass
Severity	Low

Test Case ID	UR-020
Title	UserSetting user relationship returns BelongsTo
Objective	Verify the user() relationship on UserSetting returns a BelongsTo instance.
Preconditions	- Application running - All dependencies installed
Test Steps	1. Instantiate UserSetting. 2. Call user() method. 3. Verify return type is BelongsTo.
Test Data	- Input: UserSetting instance - Expected type: Illuminate\Database\Eloquent\Relations\BelongsTo
Expected Result	- user() returns BelongsTo relationship.
Actual Result	- BelongsTo relationship verified.
Status	Pass
Severity	Low

File: VerificationController-Test.txt

Test Case ID	VC-001
Title	VerificationController constructor applies correct middleware with proper chaining
Objective	Verify that the VerificationController constructor correctly applies the expected middlewares ('auth', 'signed', 'throttle:6,1') with proper chaining (.only() method), ensuring route protection and rate limiting.
Preconditions	- Application running - Database seeded - Mockery available for mocking - VerificationController and required classes loaded
Test Steps	1. Mock PendingMiddleware objects for 'signed' and 'throttle:6,1' middlewares. 2. Create a partial mock of the VerificationController. 3. Set up middleware expectations: a. middleware('auth') should be called once and return the controller itself. b. middleware('signed') should be called once and return the mocked PendingMiddleware for signed. c. middleware('throttle:6,1') should be called once and return the mocked PendingMiddleware for throttle. 4. Set up .only('verify') to be called on signed middleware mock and .only('verify', 'resend') on throttle middleware mock. 5. Invoke the constructor of the controller. 6. Verify all expected middleware and chaining method calls occurred.
Test Data	- PendingMiddleware mocks - Middleware names: 'auth', 'signed', 'throttle:6,1' - Methods: only('verify'), only('verify', 'resend')
Expected Result	- Constructor applies 'auth' middleware. - Constructor applies 'signed' middleware and chains .only('verify'). - Constructor applies 'throttle:6,1' middleware and chains .only('verify', 'resend'). - All middleware and chaining methods are called exactly once as expected.
Actual Result	Constructor correctly applies all expected middlewares with proper chaining; all expectations are met and assertions pass.
Status	Pass
Severity	High

Test Case ID	VC-002
Title	redirectTo property is set to RouteServiceProvider::HOME
Objective	Verify that the protected redirectTo property of VerificationController is set to the expected value (RouteServiceProvider::HOME), ensuring correct redirection after verification.
Preconditions	- Application running - Database seeded - VerificationController and RouteServiceProvider loaded
Test Steps	1. Instantiate the VerificationController. 2. Use reflection helper function to access the protected redirectTo property. 3. Compare the value of redirectTo with RouteServiceProvider::HOME.
Test Data	- redirectTo property - Expected value: \Crater\Providers\RouteServiceProvider::HOME
Expected Result	- redirectTo property equals \Crater\Providers\RouteServiceProvider::HOME.
Actual Result	redirectTo property is correctly set to RouteServiceProvider::HOME.
Status	Pass
Severity	Medium

Test Case ID	VC-003
Title	VerificationController correctly uses VerifiesEmails trait methods
Objective	Verify that VerificationController implements the VerifiesEmails trait and provides its required methods: show, verify, and resend.
Preconditions	- Application running - Database seeded - VerificationController and VerifiesEmails trait loaded
Test Steps	1. Instantiate the VerificationController. 2. Check if the method show exists on the controller. 3. Check if the method verify exists on the controller. 4. Check if the method resend exists on the controller.
Test Data	- Methods to check: show, verify, resend - Trait: VerifiesEmails
Expected Result	- The show method exists on the VerificationController. - The verify method exists on the VerificationController. - The resend method exists on the VerificationController.
Actual Result	All specified methods (show, verify, resend) exist on VerificationController; trait functions verified.
Status	Pass
Severity	Medium

File: VerifyCsrfToken-Test.txt

Test Case ID	VCT-001
Title	Verify $addHttpCookie property is set to true in VerifyCsrfToken middleware
Objective	Ensure that the VerifyCsrfToken middleware correctly sets the $addHttpCookie property to true.
Preconditions	- Application running - Crater middleware class VerifyCsrfToken is implemented - ReflectionClass available in the PHP environment - Database seeded (if the middleware logic requires it) - Mockery available for mocking Application and Encrypter
Test Steps	1. Create mock instances of Application and Encrypter classes using Mockery. 2. Instantiate the VerifyCsrfToken middleware with the mocked dependencies. 3. Use PHP ReflectionClass to access the protected $addHttpCookie property of the middleware instance. 4. Retrieve the value of $addHttpCookie property. 5. Verify that the value of $addHttpCookie is true.
Test Data	- $addHttpCookie property value (expected: true)
Expected Result	- The $addHttpCookie property of the VerifyCsrfToken middleware is set to true.
Actual Result	The $addHttpCookie property is set to true, as expected.
Status	Pass
Severity	High

Test Case ID	VCT-002
Title	Verify $except property contains specific URIs in VerifyCsrfToken middleware
Objective	Ensure that the VerifyCsrfToken middleware correctly sets the $except property to the array ['login'].
Preconditions	- Application running - Crater middleware class VerifyCsrfToken is implemented - ReflectionClass available in the PHP environment - Database seeded (if the middleware logic requires it) - Mockery available for mocking Application and Encrypter
Test Steps	1. Create mock instances of Application and Encrypter classes using Mockery. 2. Instantiate the VerifyCsrfToken middleware with the mocked dependencies. 3. Use PHP ReflectionClass to access the protected $except property of the middleware instance. 4. Retrieve the value of $except property. 5. Verify that the value of $except property equals ['login'].
Test Data	- $except property value (expected: ['login'])
Expected Result	- The $except property of the VerifyCsrfToken middleware equals ['login'].
Actual Result	The $except property equals ['login'], as expected.
Status	Pass
Severity	High

File: helpers-Test.txt

Test Case ID	HEL-001
Title	Verify existence of get_company_setting function
Objective	To confirm that the get_company_setting function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('get_company_setting'). 2. Check that the result is true.
Test Data	- Function name: get_company_setting
Expected Result	- function_exists('get_company_setting') returns true.
Actual Result	function_exists('get_company_setting') returned true.
Status	Pass
Severity	High

Test Case ID	HEL-002
Title	Verify existence of get_app_setting function
Objective	To confirm that the get_app_setting function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('get_app_setting'). 2. Check that the result is true.
Test Data	- Function name: get_app_setting
Expected Result	- function_exists('get_app_setting') returns true.
Actual Result	function_exists('get_app_setting') returned true.
Status	Pass
Severity	High

Test Case ID	HEL-003
Title	Verify existence of get_page_title function
Objective	To confirm that the get_page_title function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('get_page_title'). 2. Check that the result is true.
Test Data	- Function name: get_page_title
Expected Result	- function_exists('get_page_title') returns true.
Actual Result	function_exists('get_page_title') returned true.
Status	Pass
Severity	High

Test Case ID	HEL-004
Title	Verify existence of set_active function
Objective	To confirm that the set_active function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory

Test Steps	1. Attempt to call function_exists('set_active'). 2. Check that the result is true.
Test Data	- Function name: set_active
Expected Result	- function_exists('set_active') returns true.
Actual Result	function_exists('set_active') returned true.
Status	Pass
Severity	Medium

Test Case ID	HEL-005
Title	Verify existence of is_url function
Objective	To confirm that the is_url function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('is_url'). 2. Check that the result is true.
Test Data	- Function name: is_url
Expected Result	- function_exists('is_url') returns true.
Actual Result	function_exists('is_url') returned true.
Status	Pass
Severity	Medium

Test Case ID	HEL-006
Title	Verify existence of getCustomFieldValueKey function
Objective	To confirm that the getCustomFieldValueKey function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('getCustomFieldValueKey'). 2. Check that the result is true.
Test Data	- Function name: getCustomFieldValueKey
Expected Result	- function_exists('getCustomFieldValueKey') returns true.
Actual Result	function_exists('getCustomFieldValueKey') returned true.
Status	Pass
Severity	Medium

Test Case ID	HEL-007
Title	Verify existence of format_money_pdf function
Objective	To confirm that the format_money_pdf function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('format_money_pdf'). 2. Check that the result is true.
Test Data	- Function name: format_money_pdf
Expected Result	- function_exists('format_money_pdf') returns true.
Actual Result	function_exists('format_money_pdf') returned true.
Status	Pass
Severity	High

Test Case ID	HEL-008
Title	Verify existence of clean_slug function
Objective	To confirm that the clean_slug function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('clean_slug'). 2. Check that the result is true.
Test Data	- Function name: clean_slug
Expected Result	- function_exists('clean_slug') returns true.
Actual Result	function_exists('clean_slug') returned true.
Status	Pass
Severity	Medium

Test Case ID	HEL-009
Title	Verify existence of getRelatedSlugs function
Objective	To confirm that the getRelatedSlugs function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('getRelatedSlugs'). 2. Check that the result is true.
Test Data	- Function name: getRelatedSlugs
Expected Result	- function_exists('getRelatedSlugs') returns true.
Actual Result	function_exists('getRelatedSlugs') returned true.
Status	Pass
Severity	Medium

Test Case ID	HEL-010
Title	Verify existence of respondJson function
Objective	To confirm that the respondJson function exists in the helpers file.
Preconditions	- Application running - helpers.php file present in app/Space directory
Test Steps	1. Attempt to call function_exists('respondJson'). 2. Check that the result is true.
Test Data	- Function name: respondJson
Expected Result	- function_exists('respondJson') returns true.
Actual Result	function_exists('respondJson') returned true.
Status	Pass
Severity	High

Test Case ID	HEL-011
Title	Verify parameter count for get_company_setting
Objective	To check that get_company_setting accepts two parameters: key and company_id.
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on get_company_setting. 2. Retrieve the list of parameters. 3. Check that there are two parameters named 'key' and 'company_id'.

Test Data	- Function name: get_company_setting
Expected Result	- Parameter count is 2. - First parameter is 'key'. - Second parameter is 'company_id'.
Actual Result	Function accepted two parameters named 'key' and 'company_id'.
Status	Pass
Severity	High

Test Case ID	HEL-012
Title	Verify parameter count for get_app_setting
Objective	To check that get_app_setting accepts one parameter: key.
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on get_app_setting. 2. Retrieve the list of parameters. 3. Check that there is one parameter named 'key'.
Test Data	- Function name: get_app_setting
Expected Result	- Parameter count is 1. - Parameter is 'key'.
Actual Result	Function accepted one parameter named 'key'.
Status	Pass
Severity	High

Test Case ID	HEL-013
Title	Verify parameter count for get_page_title
Objective	To check that get_page_title accepts one parameter: company_id.
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on get_page_title. 2. Retrieve the list of parameters. 3. Check that there is one parameter named 'company_id'.
Test Data	- Function name: get_page_title
Expected Result	- Parameter count is 1. - Parameter is 'company_id'.
Actual Result	Function accepted one parameter named 'company_id'.
Status	Pass
Severity	High

Test Case ID	HEL-014
Title	Verify default parameter for set_active function
Objective	To verify set_active accepts two parameters: path and active (with default value 'active').
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on set_active. 2. Retrieve the parameters. 3. Check that there are two parameters: path and active. 4. Confirm 'active' parameter's default value is 'active'.
Test Data	- Function name: set_active
Expected Result	- Parameter count is 2. - First parameter is 'path'. - Second parameter is 'active' with default value 'active'.

Actual Result	Both parameters present. 'active' has default value 'active'.
Status	Pass
Severity	Medium

Test Case ID	HEL-015
Title	Verify parameter count for is_url
Objective	To check that is_url accepts one parameter: path.
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on is_url. 2. Retrieve the list of parameters. 3. Check for single parameter 'path'.
Test Data	- Function name: is_url
Expected Result	- Parameter count is 1. - Parameter is 'path'.
Actual Result	Function accepted a single parameter named 'path'.
Status	Pass
Severity	Medium

Test Case ID	HEL-016
Title	Verify parameter type for getCustomFieldValueKey
Objective	To confirm getCustomFieldValueKey accepts one parameter named 'type'.
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on getCustomFieldValueKey. 2. Retrieve the list of parameters. 3. Confirm parameter is named 'type'.
Test Data	- Function name: getCustomFieldValueKey
Expected Result	- Parameter count is 1. - Parameter name is 'type'.
Actual Result	Function accepted one parameter named 'type'.
Status	Pass
Severity	Medium

Test Case ID	HEL-017
Title	Verify default parameter for format_money_pdf
Objective	To verify format_money_pdf accepts two parameters: money and currency; currency defaults to null.
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on format_money_pdf. 2. Retrieve parameters. 3. Confirm the second parameter is 'currency' and has default value null.
Test Data	- Function name: format_money_pdf
Expected Result	- Parameter count is 2. - First parameter is 'money'. - Second parameter is 'currency' with default null.
Actual Result	Both parameters present; 'currency' has default value null.
Status	Pass
Severity	High

Test Case ID	HEL-018
Title	Verify default parameter for clean_slug
Objective	To verify clean_slug accepts three parameters: model, title, id (id defaults to 0).
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on clean_slug. 2. Retrieve parameters. 3. Confirm third parameter is 'id' and has default value 0.
Test Data	- Function name: clean_slug
Expected Result	- Parameter count is 3. - Third parameter is 'id' with default 0.
Actual Result	Three parameters present; 'id' default value is 0.
Status	Pass
Severity	Medium

Test Case ID	HEL-019
Title	Verify default parameter for getRelatedSlugs
Objective	To verify getRelatedSlugs accepts three parameters: type, slug, id (id defaults to 0).
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on getRelatedSlugs. 2. Retrieve parameters. 3. Confirm third parameter is 'id' with default 0.
Test Data	- Function name: getRelatedSlugs
Expected Result	- Parameter count is 3. - Third parameter is 'id' with default 0.
Actual Result	Three parameters present; 'id' default value is 0.
Status	Pass
Severity	Medium

Test Case ID	HEL-020
Title	Verify parameter count for respondJson
Objective	To check that respondJson accepts two parameters: error and message.
Preconditions	- Application running
Test Steps	1. Use ReflectionFunction on respondJson. 2. Retrieve parameters. 3. Confirm parameters are 'error' and 'message'.
Test Data	- Function name: respondJson
Expected Result	- Parameter count is 2. - Parameters are 'error' and 'message'.
Actual Result	Both parameters present.
Status	Pass
Severity	High

Test Case ID	HEL-021
Title	Verify getCustomFieldValueKey returns string_answer for Input
Objective	To ensure getCustomFieldValueKey('Input') returns 'string_answer'.
Preconditions	- Application running

Test Steps	1. Call getCustomFieldValueKey('Input'). 2. Check return value.
Test Data	- Input: 'Input'
Expected Result	- Returns 'string_answer'.
Actual Result	Return value is 'string_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-022
Title	Verify getCustomFieldValueKey returns string_answer for TextArea
Objective	To ensure getCustomFieldValueKey('TextArea') returns 'string_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('TextArea'). 2. Check return value.
Test Data	- Input: 'TextArea'
Expected Result	- Returns 'string_answer'.
Actual Result	Return value is 'string_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-023
Title	Verify getCustomFieldValueKey returns number_answer for Phone
Objective	To ensure getCustomFieldValueKey('Phone') returns 'number_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('Phone'). 2. Check return value.
Test Data	- Input: 'Phone'
Expected Result	- Returns 'number_answer'.
Actual Result	Return value is 'number_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-024
Title	Verify getCustomFieldValueKey returns string_answer for Url
Objective	To ensure getCustomFieldValueKey('Url') returns 'string_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('Url'). 2. Check return value.
Test Data	- Input: 'Url'
Expected Result	- Returns 'string_answer'.
Actual Result	Return value is 'string_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-025
Title	Verify getCustomFieldValueKey returns number_answer for Number

Objective	To ensure getCustomFieldValueKey('Number') returns 'number_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('Number'). 2. Check return value.
Test Data	- Input: 'Number'
Expected Result	- Returns 'number_answer'.
Actual Result	Return value is 'number_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-026
Title	Verify getCustomFieldValueKey returns string_answer for Dropdown
Objective	To ensure getCustomFieldValueKey('Dropdown') returns 'string_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('Dropdown'). 2. Check return value.
Test Data	- Input: 'Dropdown'
Expected Result	- Returns 'string_answer'.
Actual Result	Return value is 'string_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-027
Title	Verify getCustomFieldValueKey returns boolean_answer for Switch
Objective	To ensure getCustomFieldValueKey('Switch') returns 'boolean_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('Switch'). 2. Check return value.
Test Data	- Input: 'Switch'
Expected Result	- Returns 'boolean_answer'.
Actual Result	Return value is 'boolean_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-028
Title	Verify getCustomFieldValueKey returns date_answer for Date
Objective	To ensure getCustomFieldValueKey('Date') returns 'date_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('Date'). 2. Check return value.
Test Data	- Input: 'Date'
Expected Result	- Returns 'date_answer'.
Actual Result	Return value is 'date_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-029
Title	Verify getCustomFieldValueKey returns time_answer for Time
Objective	To ensure getCustomFieldValueKey('Time') returns 'time_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('Time'). 2. Check return value.
Test Data	- Input: 'Time'
Expected Result	- Returns 'time_answer'.
Actual Result	Return value is 'time_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-030
Title	Verify getCustomFieldValueKey returns date_time_answer for DateTime
Objective	To ensure getCustomFieldValueKey('DateTime') returns 'date_time_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('DateTime'). 2. Check return value.
Test Data	- Input: 'DateTime'
Expected Result	- Returns 'date_time_answer'.
Actual Result	Return value is 'date_time_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-031
Title	Verify getCustomFieldValueKey returns string_answer for unknown type
Objective	To ensure getCustomFieldValueKey for unknown types defaults to 'string_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('UnknownType'). 2. Check return value.
Test Data	- Input: 'UnknownType'
Expected Result	- Returns 'string_answer'.
Actual Result	Return value is 'string_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-032
Title	Verify getCustomFieldValueKey is case sensitive
Objective	To confirm getCustomFieldValueKey handles varied capitalizations by returning 'string_answer'.
Preconditions	- Application running
Test Steps	1. Call getCustomFieldValueKey('input'). 2. Check return value. 3. Call getCustomFieldValueKey('INPUT'). 4. Check return value.
Test Data	- 'input', 'INPUT' as input strings

Expected Result	- Returns 'string_answer' for both cases.
Actual Result	Return value for both 'input' and 'INPUT' was 'string_answer'.
Status	Pass
Severity	Medium

Test Case ID	HEL-033
Title	Verify helpers file uses required classes
Objective	To confirm usage of necessary class imports in helpers.php.
Preconditions	- helpers.php file present
Test Steps	1. Load file contents of helpers.php. 2. Check for usage of Crater\Models\CompanySetting, Currency, CustomField, Setting, Illuminate\Support\Str.
Test Data	- File: helpers.php
Expected Result	- File contains necessary class import statements.
Actual Result	All required class imports were found.
Status	Pass
Severity	Medium

Test Case ID	HEL-034
Title	Verify helpers file line count is reasonable
Objective	To ensure file size is within reasonable range for maintainability.
Preconditions	- helpers.php file present
Test Steps	1. Load helpers.php. 2. Count lines in file. 3. Check line count >100 and <300.
Test Data	- File: helpers.php
Expected Result	- Line count is between 100 and 300.
Actual Result	File contained 187 lines.
Status	Pass
Severity	Low

Test Case ID	HEL-035
Title	Verify get_company_setting checks database_created file
Objective	To ensure get_company_setting implementation checks for database_created file.
Preconditions	- get_company_setting function implemented
Test Steps	1. Get source code of get_company_setting. 2. Check that it contains "Storage::disk('local')->has('database_created')". 3. Verify usage of "CompanySetting::getSetting".
Test Data	- Source code scan
Expected Result	- Implementation checks for database_created file and uses CompanySetting::getSetting.
Actual Result	Both checks confirmed.
Status	Pass
Severity	High

Test Case ID	HEL-036

Title	Verify get_app_setting checks database_created file
Objective	To ensure get_app_setting implementation checks for database_created file.
Preconditions	- get_app_setting function implemented
Test Steps	1. Get source code of get_app_setting. 2. Check that it contains "Storage::disk('local')->has('database_created')". 3. Verify usage of "Setting::getSetting".
Test Data	- Source code scan
Expected Result	- Implementation checks for database_created file and uses Setting::getSetting.
Actual Result	Both checks confirmed.
Status	Pass
Severity	High

Test Case ID	HEL-037
Title	Verify get_page_title uses Route facade
Objective	To ensure get_page_title uses Route::currentRouteName().
Preconditions	- get_page_title function implemented
Test Steps	1. Get source code of get_page_title. 2. Check for usage of "Route::currentRouteName()".
Test Data	- Source code scan
Expected Result	- Implementation uses Route::currentRouteName().
Actual Result	Confirmed Route usage.
Status	Pass
Severity	Medium

Test Case ID	HEL-038
Title	Verify get_page_title has default page title
Objective	To ensure get_page_title provides default page title string.
Preconditions	- get_page_title function implemented
Test Steps	1. Get source code of get_page_title. 2. Check for "Crater - Self Hosted Invoicing Platform" string.
Test Data	- Source code scan
Expected Result	- Implementation contains default title.
Actual Result	Default title string present.
Status	Pass
Severity	Medium

Test Case ID	HEL-039
Title	Verify get_page_title checks customer dashboard route
Objective	To ensure get_page_title checks for customer.dashboard route and sets customer_portal_page_title.
Preconditions	- get_page_title function implemented
Test Steps	1. Get source code of get_page_title. 2. Check for "customer.dashboard" and "customer_portal_page_title".
Test Data	- Source code scan
Expected Result	- Implementation checks for customer.dashboard route and sets title accordingly.

Actual Result	Confirmed both checks.
Status	Pass
Severity	High

Test Case ID	HEL-040
Title	Verify set_active uses call_user_func_array
Objective	To ensure set_active uses call_user_func_array for dynamic argument passing.
Preconditions	- set_active function implemented
Test Steps	1. Get source code of set_active. 2. Check for usage of call_user_func_array. 3. Verify use of Request::is.
Test Data	- Source code scan
Expected Result	- call_user_func_array and Request::is used in implementation.
Actual Result	Both functions present in source.
Status	Pass
Severity	Medium

Test Case ID	HEL-041
Title	Verify is_url uses call_user_func_array
Objective	To ensure is_url uses call_user_func_array for dynamic argument passing.
Preconditions	- is_url function implemented
Test Steps	1. Get source code of is_url. 2. Check for usage of call_user_func_array. 3. Verify use of Request::is.
Test Data	- Source code scan
Expected Result	- call_user_func_array and Request::is used in implementation.
Actual Result	Both calls present.
Status	Pass
Severity	Medium

Test Case ID	HEL-042
Title	Verify getCustomFieldValueKey uses switch statement
Objective	To ensure type mapping uses switch for decision logic.
Preconditions	- getCustomFieldValueKey function implemented
Test Steps	1. Get source code of getCustomFieldValueKey. 2. Check for "switch ($type)" in implementation.
Test Data	- Source code scan
Expected Result	- switch statement used for mapping type.
Actual Result	Switch statement found.
Status	Pass
Severity	Medium

Test Case ID	HEL-043
Title	Verify format_money_pdf divides money by 100
Objective	To verify money values are normalized by division in format_money_pdf.

Preconditions	- format_money_pdf function implemented
Test Steps	1. Get source code of format_money_pdf. 2. Check for operation "$money = $money / 100".
Test Data	- Source code scan
Expected Result	- Money is divided by 100.
Actual Result	Division operation present.
Status	Pass
Severity	High

Test Case ID	HEL-044
Title	Verify format_money_pdf uses number_format
Objective	To ensure number formatting is applied in format_money_pdf.
Preconditions	- format_money_pdf function implemented
Test Steps	1. Get source code of format_money_pdf. 2. Check for usage of number_format function.
Test Data	- Source code scan
Expected Result	- number_format used in implementation.
Actual Result	number_format function detected.
Status	Pass
Severity	Low

Test Case ID	HEL-045
Title	Verify format_money_pdf checks swap_currency_symbol
Objective	To verify currency symbol swapping logic is present in format_money_pdf.
Preconditions	- format_money_pdf function implemented
Test Steps	1. Get source code of format_money_pdf. 2. Check for "swap_currency_symbol" and "DejaVu Sans" usage.
Test Data	- Source code scan
Expected Result	- swap_currency_symbol and DejaVu Sans present in implementation.
Actual Result	Both checks confirmed.
Status	Pass
Severity	Medium

Test Case ID	HEL-046
Title	Verify clean_slug uses Str::upper
Objective	To confirm clean_slug uses case conversion and slugging helpers.
Preconditions	- clean_slug function implemented
Test Steps	1. Get source code of clean_slug. 2. Check for "Str::upper" and "Str::slug" usage.
Test Data	- Source code scan
Expected Result	- Str::upper and Str::slug used.
Actual Result	Both helpers present.
Status	Pass
Severity	Medium

Test Case ID	HEL-047

Title	Verify clean_slug creates CUSTOM prefix
Objective	To verify custom prefixes are added in slug generation.
Preconditions	- clean_slug function implemented
Test Steps	1. Get source code of clean_slug. 2. Check for "CUSTOM_" string in implementation.
Test Data	- Source code scan
Expected Result	- "CUSTOM_" prefix added to slug.
Actual Result	Prefix detected.
Status	Pass
Severity	Medium

Test Case ID	HEL-048
Title	Verify clean_slug uses getRelatedSlugs
Objective	To verify uniqueness logic involves related slug lookup.
Preconditions	- clean_slug function implemented
Test Steps	1. Get source code of clean_slug. 2. Check for usage of "getRelatedSlugs".
Test Data	- Source code scan
Expected Result	- getRelatedSlugs used in implementation.
Actual Result	Confirmed usage present.
Status	Pass
Severity	Medium

Test Case ID	HEL-049
Title	Verify clean_slug throws exception for too many variations
Objective	To confirm exception thrown when unable to create unique slug within limit.
Preconditions	- clean_slug function implemented
Test Steps	1. Get source code of clean_slug. 2. Check for "throw new \Exception" with message "Can not create a unique slug".
Test Data	- Source code scan
Expected Result	- Exception thrown when unique slug cannot be created.
Actual Result	Exception logic present.
Status	Pass
Severity	High

Test Case ID	HEL-050
Title	Verify clean_slug loops up to 10 times
Objective	To check clean_slug attempts up to 10 variations for slug uniqueness.
Preconditions	- clean_slug function implemented
Test Steps	1. Get source code of clean_slug. 2. Find loop construct "for ($i = 1; $i <= 10; $i++)".
Test Data	- Source code scan
Expected Result	- Loop iterates up to 10 times for slug variations.
Actual Result	Loop structure confirmed.
Status	Pass

Severity	Medium

Test Case ID	HEL-051
Title	Verify getRelatedSlugs uses CustomField model
Objective	To verify related slug query uses CustomField model and applies necessary conditions.
Preconditions	- getRelatedSlugs function implemented
Test Steps	1. Get source code of getRelatedSlugs. 2. Check for "CustomField::select('slug')" and related where conditions.
Test Data	- Source code scan
Expected Result	- CustomField model and correct where clauses used.
Actual Result	Model and clauses detected.
Status	Pass
Severity	Medium

Test Case ID	HEL-052
Title	Verify respondJson uses response helper
Objective	To confirm respondJson uses response()->json for output.
Preconditions	- respondJson function implemented
Test Steps	1. Get source code of respondJson. 2. Check for "response()->json".
Test Data	- Source code scan
Expected Result	- response()->json used in implementation.
Actual Result	Confirmed usage.
Status	Pass
Severity	High

Test Case ID	HEL-053
Title	Verify respondJson returns error and message
Objective	To ensure error and message fields are returned in json response.
Preconditions	- respondJson function implemented
Test Steps	1. Get source code of respondJson. 2. Check for "'error' => \$error" and "'message' => \$message".
Test Data	- Source code scan
Expected Result	- Both fields present in response array.
Actual Result	Both keys present in response.
Status	Pass
Severity	High

Test Case ID	HEL-054
Title	Verify respondJson uses 422 status code
Objective	To confirm respondJson returns HTTP status code 422 for errors.
Preconditions	- respondJson function implemented
Test Steps	1. Get source code of respondJson. 2. Check for ", 422)" statement.
Test Data	- Source code scan

Expected Result	- 422 status code returned in response.
Actual Result	Status code confirmed.
Status	Pass
Severity	High

Test Case ID	HEL-055
Title	Verify helpers file defines exactly 11 functions
Objective	To ensure helpers.php defines all necessary functions.
Preconditions	- helpers.php file present
Test Steps	1. List all function names in helpers.php. 2. Count number of functions. 3. Verify all listed functions exist.
Test Data	- Function list: get_company_setting, get_app_setting, get_page_title, set_active, is_url, getCustomFieldValueKey, format_money_pdf, clean_slug, getRelatedSlugs, respondJson
Expected Result	- All listed functions exist and total count is exactly 10.
Actual Result	10 functions present as expected.
Status	Pass
Severity	High

Test Case ID	HEL-056
Title	Verify get_company_setting has documentation
Objective	To ensure get_company_setting function is documented.
Preconditions	- get_company_setting function implemented
Test Steps	1. Use ReflectionFunction on get_company_setting. 2. Check for doc comment.
Test Data	- Function name: get_company_setting
Expected Result	- Doc comment exists.
Actual Result	Doc comment found.
Status	Pass
Severity	Low

Test Case ID	HEL-057
Title	Verify get_app_setting has documentation
Objective	To ensure get_app_setting function is documented.
Preconditions	- get_app_setting function implemented
Test Steps	1. Use ReflectionFunction on get_app_setting. 2. Check for doc comment.
Test Data	- Function name: get_app_setting
Expected Result	- Doc comment exists.
Actual Result	Doc comment found.
Status	Pass
Severity	Low

Test Case ID	HEL-058
Title	Verify get_page_title has documentation

Objective	To ensure get_page_title function is documented.
Preconditions	- get_page_title function implemented
Test Steps	1. Use ReflectionFunction on get_page_title. 2. Check for doc comment.
Test Data	- Function name: get_page_title
Expected Result	- Doc comment exists.
Actual Result	Doc comment found.
Status	Pass
Severity	Low

Test Case ID	HEL-059
Title	Verify set_active has documentation
Objective	To ensure set_active function is documented.
Preconditions	- set_active function implemented
Test Steps	1. Use ReflectionFunction on set_active. 2. Check for doc comment.
Test Data	- Function name: set_active
Expected Result	- Doc comment exists.
Actual Result	Doc comment found.
Status	Pass
Severity	Low

Test Case ID	HEL-060
Title	Verify is_url has documentation
Objective	To ensure is_url function is documented.
Preconditions	- is_url function implemented
Test Steps	1. Use ReflectionFunction on is_url. 2. Check for doc comment.
Test Data	- Function name: is_url
Expected Result	- Doc comment exists.
Actual Result	Doc comment found.
Status	Pass
Severity	Low

Test Case ID	HEL-061
Title	Verify getCustomFieldValueKey has documentation
Objective	To ensure getCustomFieldValueKey function is documented.
Preconditions	- getCustomFieldValueKey function implemented
Test Steps	1. Use ReflectionFunction on getCustomFieldValueKey. 2. Check for doc comment.
Test Data	- Function name: getCustomFieldValueKey
Expected Result	- Doc comment exists.
Actual Result	Doc comment found.
Status	Pass
Severity	Low

Test Case ID	HEL-062
Title	Verify format_money_pdf has documentation
Objective	To ensure format_money_pdf function is documented.
Preconditions	- format_money_pdf function implemented
Test Steps	1. Use ReflectionFunction on format_money_pdf. 2. Check for doc comment.
Test Data	- Function name: format_money_pdf
Expected Result	- Doc comment exists.
Actual Result	Doc comment found.
Status	Pass
Severity	Low

Test Case ID	HEL-063
Title	Verify clean_slug has documentation
Objective	To ensure clean_slug function is documented.
Preconditions	- clean_slug function implemented
Test Steps	1. Use ReflectionFunction on clean_slug. 2. Check for doc comment.
Test Data	- Function name: clean_slug
Expected Result	- Doc comment exists.
Actual Result	Doc comment found.
Status	Pass
Severity	Low

Test Case ID	HEL-064
Title	Verify getCustomFieldValueKey returns string
Objective	To ensure getCustomFieldValueKey returns a string type value.
Preconditions	- getCustomFieldValueKey function implemented
Test Steps	1. Call getCustomFieldValueKey('Input'). 2. Check that return type is string.
Test Data	- Input: 'Input'
Expected Result	- Return type is string.
Actual Result	Returned value was string.
Status	Pass
Severity	Medium

Test Case ID	HEL-065
Title	Verify getCustomFieldValueKey handles all documented types
Objective	To verify getCustomFieldValueKey returns correct answer type for all supported field types.
Preconditions	- getCustomFieldValueKey function implemented
Test Steps	1. For each type in the documented list, call getCustomFieldValueKey(type). 2. Verify returned value matches expected.

Test Data	- Types: 'Input' => 'string_answer', 'TextArea' => 'string_answer', 'Phone' => 'number_answer', 'Url' => 'string_answer', 'Number' => 'number_answer', 'Dropdown' => 'string_answer', 'Switch' => 'boolean_answer', 'Date' => 'date_answer', 'Time' => 'time_answer', 'DateTime' => 'date_time_answer'
Expected Result	- For each input type, returned value matches expected answer key.
Actual Result	All mappings returned expected answer keys.
Status	Pass
Severity	Medium