Enhanced Test Cases Report - 65 Test Cases

Total Test Cases	65
Categories	Security, Usability, Integration, Performance, Boundary, Error Handling, Functiona
Priority Levels	High, Medium, Low, Critical
Generated On	2025-08-03 14:35:51

Test Case 1: Successful Admin Login with Valid Credentials

Test ID	TC_001
Module	User Authentication > Login
Category	Functional
Priority	Critical
Test Type	Positive
Risk Level	Critical
Estimated Time	5 minutes
Description	Verifies that an admin can log in using valid credentials and is redirected to the dashboard.
Objective	Ensure only valid admin credentials allow access to the admin portal.
Preconditions	Admin user account exists with username 'admin_egypt' and password 'EgYpt@2024'; Admin portal
Expected Result	Admin user is logged in and redirected to the dashboard.

Detailed Test Steps with Data:

Step Number	1
Description	Open Chrome browser and navigate to the admin login page.
Test Data	{ "browser": "Chrome", "url": "https://corp.example.com/admin/login" }
Expected Behavior	Login page loads successfully with username and password fields.

Step Number	2
Description	Enter valid admin username and password.
Test Data	{ "username": "admin_egypt", "password": "EgYpt@2024" }

Expected Behavior	Fields accept input and login button becomes enabled.

Step Number	3
Description	Click the 'Login' button.
Test Data	{ "action": "click" }
Expected Behavior	System authenticates credentials and redirects to admin dashboard.

Step Number	4
Description	Verify the admin dashboard is displayed with the correct admin name.
Test Data	{ "expected_admin_name": "Sherin Samir" }
Expected Behavior	Dashboard loads with 'Welcome, Sherin Samir' displayed.

Test Data Summary:

{ "overall_input": "admin_egypt / EgYpt@2024", "key_parameters": "username, password, browser, url" }

Validation Criteria	Successful authentication; Redirection to dashboard; Correct admin name displayed
Dependencies	Admin user must exist in the database
Notes	Test with Egypt-specific admin as per documentation.

Test Case 2: Form Validation - Mandatory and Conditional Fields

Test ID	TC_002
Module	Customer Onboarding > Entitlement Assignment
Category	Functional
Priority	High
Test Type	Negative
Risk Level	High
Estimated Time	7 minutes
Description	Checks that the onboarding form enforces mandatory and conditional mandatory fields.
Objective	Ensure that form validation prevents submission when required fields are missing.
Preconditions	User is logged in as admin; Onboarding form is accessible
Expected Result	Form only submits when all mandatory and conditional fields are filled.

Detailed Test Steps with Data:

Step Number	1

Description	Navigate to the customer onboarding form.
Test Data	{ "url": "https://corp.example.com/admin/onboarding" }
Expected Behavior	Onboarding form loads with all fields visible.

Step Number	2
Description	Leave all mandatory fields (e.g., 'Customer Name', 'GCIF', 'Country') empty and attempt to submit.
Test Data	{ "Customer Name": "", "GCIF": "", "Country": "" }
Expected Behavior	Form displays validation errors for each empty mandatory field.

Step Number	3
Description	Select 'Country' as 'Egypt' and leave the 'GCIF' field empty.
Test Data	{ "Country": "Egypt", "GCIF": "" }
Expected Behavior	Form displays a conditional mandatory error for 'GCIF' field.

Step Number	4
Description	Fill all mandatory and conditional fields with valid data and submit.
Test Data	{ "Customer Name": "Cairo Corp", "GCIF": "EG123456", "Country": "Egypt" }
Expected Behavior	Form submits successfully and confirmation message is displayed.

Test Data Summary:

{ "overall_input": "Customer Name, GCIF, Country", "key_parameters": "Mandatory and conditional fields" }

Validation Criteria	Mandatory and conditional validation enforced; Appropriate error messages displayed
Dependencies	Admin login; Onboarding form available
Notes	Covers Egypt-specific conditional logic.

Test Case 3: Boundary Test - Amount Field Validation

Test ID	TC_003
Module	Product Entitlement > Payment Module Configuration

Category	Boundary
Priority	High
Test Type	Boundary
Risk Level	Medium
Estimated Time	6 minutes
Description	Tests the amount field with minimum, maximum, and out-of-bound values.
Objective	Ensure amount fields accept only valid values as per defined data type (AMT(14,2)).
Preconditions	User is logged in as admin; Payment module configuration form is accessible
Expected Result	Amount field enforces lower/upper bounds and decimal precision.

Detailed Test Steps with Data:

Step Number	1
Description	Enter minimum allowed amount (0.01) in the 'Transaction Amount' field.
Test Data	{ "Transaction Amount": "0.01" }
Expected Behavior	Field accepts input and no error is displayed.

Step Number	2
Description	Enter maximum allowed amount (999999999999.99) in the 'Transaction Amount' field.
Test Data	{ "Transaction Amount": "999999999999.99" }
Expected Behavior	Field accepts input and no error is displayed.

Step Number	3
Description	Enter an amount exceeding the maximum (1000000000000.00).
Test Data	{ "Transaction Amount": "1000000000000.00" }
Expected Behavior	Field displays an error indicating value exceeds allowed maximum.

Step Number	4
Description	Enter a negative amount (-10.00).
Test Data	{ "Transaction Amount": "-10.00" }
Expected Behavior	Field displays an error indicating negative values are not allowed.

Step Number	5
Description	Enter an amount with more than two decimal places (100.123).

Test Data	{ "Transaction Amount": "100.123" }
Expected Behavior	Field displays an error indicating only two decimal places are allowed.

Test Data Summary:

{ "overall_input": "0.01, 999999999999.99, 1000000000000.00, -10.00, 100.123", "key_parameters": "AMT(14,2) validation" }

Validation Criteria	Min/max/precision validation enforced; Appropriate error messages
Dependencies	Admin login; Payment module form
Notes	Covers numeric and format boundaries.

Test Case 4: Security - SQL Injection on Beneficiary Addition

Test ID	TC_004
Module	Admin Portal > Beneficiary Management
Category	Security
Priority	Critical
Test Type	Negative
Risk Level	Critical
Estimated Time	8 minutes
Description	Attempts to add a beneficiary with SQL injection payload in the 'Beneficiary Name' field.
Objective	Ensure the system sanitizes input and prevents SQL injection.
Preconditions	User is logged in as admin; Beneficiary addition form is accessible
Expected Result	System rejects or sanitizes SQL injection attempts.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the 'Add Beneficiary' form.
Test Data	{ "url": "https://corp.example.com/admin/beneficiaries/add" }
Expected Behavior	Form loads with beneficiary input fields.

Step Number	2
Description	Enter SQL injection payload in 'Beneficiary Name' field.
Test Data	{ "Beneficiary Name": "Robert'); DROP TABLE Beneficiaries;--" }

Expected Behavior	Field accepts input, but payload is sanitized.

Step Number	3
Description	Fill other required fields with valid data and submit the form.
Test Data	{ "Account Number": "1234567890", "Bank": "National Bank of Egypt" }
Expected Behavior	Form submission is processed without executing malicious SQL.

Step Number	4
Description	Verify that the beneficiary is not added with malicious content and database is intact.
Test Data	{ "query": "SELECT * FROM Beneficiaries WHERE Name LIKE '%DROP TABLE%'" }
Expected Behavior	No beneficiary with malicious name is found; database structure is unchanged.

Test Data Summary:

{ "overall_input": "Robert'); DROP TABLE Beneficiaries;--, 1234567890, National Bank of Egypt", "key_parameters": "Beneficiary Name input" }

Validation Criteria	Input sanitization; No SQL injection executed
Dependencies	Admin login; Database access for verification
Notes	Critical for database security.

Test Case 5: Integration - Product Entitlement Assignment and Database Verification

Test ID	TC_005
Module	Customer Onboarding > Product Entitlement
Category	Integration
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	10 minutes
Description	Assigns a product entitlement during onboarding and verifies the record in the database.
Objective	Ensure product entitlement is correctly stored in the backend.
Preconditions	User is logged in as admin; Onboarding form is accessible; Database access is available
Expected Result	Product entitlement is correctly assigned and stored in the database.

Detailed Test Steps with Data:

Step Number	1
Description	Fill onboarding form with new customer details and select 'Tax Collection' as product.

Test Data	{
	"Customer Name": "Delta Industries",
	"GCIF": "EG987654",
	"Country": "Egypt",
	"Product Entitlement": "Tax Collection"
	}

Expected Behavior	Form fields accept input and product entitlement is selectable.

Step Number	2
Description	Submit the onboarding form.

Test Data	{
	"action": "submit"
	}

Expected Behavior	Submission is successful and confirmation message is displayed.

Step Number	3
Description	Query the database for the new customer's product entitlements.

Test Data	{
	"query": "SELECT entitlement FROM customer_entitlements WHERE gcif='EG987654'"
	}

Expected Behavior	Database returns 'Tax Collection' for the specified GCIF.

Test Data Summary:

{ "overall_input": "Delta Industries, EG987654, Egypt, Tax Collection", "key_parameters": "Customer details, product entitlement" }

Validation Criteria	Entitlement assigned in UI; Entitlement present in DB
Dependencies	Admin login; Database access
Notes	Validates UI-to-database integration.

Test Case 6: Usability - Admin Portal Add Beneficiary Workflow

Test ID	TC_006
Module	Admin Portal > Beneficiary Management
Category	Usability
Priority	Medium
Test Type	Positive
Risk Level	Medium
Estimated Time	6 minutes
Description	Tests the usability of the add beneficiary workflow, ensuring clear feedback and logical navigation.
Objective	Ensure the workflow is user-friendly and provides clear feedback.

Preconditions	User is logged in as admin
Expected Result	Admin can easily add a beneficiary with clear feedback and logical navigation.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the 'Beneficiaries' section in the admin portal.
Test Data	{ "menu_option": "Beneficiaries" }
Expected Behavior	Beneficiaries page loads with 'Add Beneficiary' button visible.

Step Number	2
Description	Click 'Add Beneficiary' and observe the form layout.
Test Data	{ "action": "click" }
Expected Behavior	Add Beneficiary form appears with all required fields clearly labeled.

Step Number	3
Description	Enter valid beneficiary details and submit.
Test Data	{ "Beneficiary Name": "Ahmed Mostafa", "Account Number": "2345678901", "Bank": "Banque Misr" }
Expected Behavior	Form accepts input and displays a loading indicator upon submission.

Step Number	4
Description	Verify that a success message is shown and the new beneficiary appears in the list.
Test Data	{ "expected_beneficiary": "Ahmed Mostafa" }
Expected Behavior	Success message is displayed and beneficiary is listed.

Test Data Summary:

{ "overall_input": "Ahmed Mostafa, 2345678901, Banque Misr", "key_parameters": "Beneficiary details" }

Validation Criteria	Clear form layout; Success feedback; Beneficiary appears in list
Dependencies	Admin login
Notes	Focus on user experience.

Test Case 7: Error Handling - Invalid Data Types in Numeric Fields

Test ID	TC_007
Module	Product Entitlement > Payment Module Configuration
Category	Error Handling
Priority	Medium
Test Type	Negative
Risk Level	Medium
Estimated Time	5 minutes
Description	Attempts to submit a form with alphabetic characters in a numeric field.
Objective	Ensure system rejects invalid data types and displays appropriate error messages.
Preconditions	User is logged in as admin; Payment module configuration form is accessible
Expected Result	System enforces numeric data type and displays clear error messages.

Detailed Test Steps with Data:

Step Number	1
Description	Enter alphabetic characters in a numeric-only field (e.g., 'Verifier Code').
Test Data	{ "Verifier Code": "ABC123" }
Expected Behavior	Field displays an error indicating only numbers are allowed.

Step Number	2
Description	Enter special characters in the same field.
Test Data	{ "Verifier Code": "!@#456" }
Expected Behavior	Field displays an error indicating only numbers are allowed.

Step Number	3
Description	Enter a valid numeric value and submit the form.
Test Data	{ "Verifier Code": "789456" }
Expected Behavior	Field accepts input and form submission proceeds.

Test Data Summary:

{ "overall_input": "ABC123, !@#456, 789456", "key_parameters": "Verifier Code field" }

Validation Criteria	Invalid input rejected; Clear error messages
Dependencies	Admin login; Payment module form
Notes	Covers data type enforcement.

Test Case 8: Performance - Bulk Beneficiary Upload

Test ID	TC_008
Module	Admin Portal > Beneficiary Management
Category	Performance
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	10 minutes
Description	Tests the system's ability to handle bulk uploads of beneficiaries via CSV file.
Objective	Ensure the system processes large beneficiary uploads efficiently.
Preconditions	User is logged in as admin; Bulk upload feature is enabled
Expected Result	Bulk upload completes successfully within performance expectations.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the 'Bulk Upload' section in the admin portal.
Test Data	{ "menu_option": "Bulk Upload" }
Expected Behavior	Bulk upload page loads with file upload option.

Step Number	2
Description	Select and upload a CSV file containing 1000 beneficiary records.
Test Data	{ "file_path": "/testdata/beneficiaries_1000.csv" }
Expected Behavior	File is accepted and upload process begins.

Step Number	3
Description	Monitor the upload progress and time to completion.
Test Data	{ "expected_time": "Under 60 seconds" }
Expected Behavior	Upload completes within expected time and success message is displayed.

Step Number	4
Description	Verify that all 1000 beneficiaries appear in the list.

Test Data	{ "expected_count": "1000" }
Expected Behavior	Beneficiary list displays all uploaded records.

Test Data Summary:

{ "overall_input": "beneficiaries_1000.csv", "key_parameters": "File size, record count" }

Validation Criteria	Upload completes within SLA; All records appear in list
Dependencies	Admin login; Bulk upload enabled
Notes	Tests system scalability.

Test Case 9: Negative Authentication - Invalid Password

Test ID	TC_009
Module	User Authentication > Login
Category	Functional
Priority	Medium
Test Type	Negative
Risk Level	Low
Estimated Time	3 minutes
Description	Attempts to log in with a valid username but invalid password.
Objective	Ensure system rejects invalid credentials and displays appropriate error.
Preconditions	Admin user account exists
Expected Result	Login is rejected and error message is shown.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the admin login page.
Test Data	{ "browser": "Firefox", "url": "https://corp.example.com/admin/login" }
Expected Behavior	Login page loads successfully.

Step Number	2
Description	Enter valid username and invalid password.
Test Data	{ "username": "admin_egypt", "password": "WrongPass2024" }

Expected Behavior	Fields accept input and login button becomes enabled.

Step Number	3
Description	Click the 'Login' button.
Test Data	{ "action": "click" }
Expected Behavior	System rejects login and displays 'Invalid credentials' error.

Test Data Summary:

{ "overall_input": "admin_egypt / WrongPass2024", "key_parameters": "username, password" }

Validation Criteria	Invalid credentials rejected; Error message displayed
Dependencies	Admin user exists
Notes	Covers negative authentication scenario.

Test Case 10: Future-Proofing - Add New Governmental Payment Type

Test ID	TC_010
Module	Admin Portal > Payment Type Management
Category	Functional
Priority	High
Test Type	Positive
Risk Level	Medium
Estimated Time	7 minutes
Description	Tests the admin portal's ability to add a new governmental payment type after go-live.
Objective	Ensure system supports addition of new payment types without code changes.
Preconditions	User is logged in as admin; Payment Type Management section is accessible
Expected Result	New payment type is added and available for use without code deployment.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to 'Payment Type Management' in the admin portal.
Test Data	{ "menu_option": "Payment Type Management" }
Expected Behavior	Page loads with option to add new payment type.

Step Number	2

Description	Click 'Add New Payment Type' and fill in details.
Test Data	{ "Payment Type Name": "Municipal Fees", "Description": "Payments for municipal services", "Default Country": "Egypt" }
Expected Behavior	Form accepts input and displays preview.

Step Number	3
Description	Submit the new payment type.
Test Data	{ "action": "submit" }
Expected Behavior	System adds new payment type and displays success message.

Step Number	4
Description	Verify that 'Municipal Fees' appears in the list of available payment types.
Test Data	{ "expected_payment_type": "Municipal Fees" }
Expected Behavior	New payment type is listed and selectable for entitlement assignment.

Test Data Summary:

{ "overall_input": "Municipal Fees, Payments for municipal services, Egypt", "key_parameters": "Payment type details" }

Validation Criteria	New payment type added; Available for entitlement
Dependencies	Admin login; Payment Type Management enabled
Notes	Validates future-proofing requirement.

Test Case 11: Onboard Customer with Egypt GCIF and Default Entitlements

Test ID	TC_011
Module	Customer Onboarding > Entitlement Assignment
Category	Functional
Priority	Critical
Test Type	Positive
Risk Level	Critical
Estimated Time	8 minutes
Description	Verify that a new customer with GCIF level set to Egypt is automatically assigned default product ent
Objective	Ensure correct entitlement mapping and default selections for Egypt-based customers.

Preconditions	Admin user is logged into the admin portal.; No existing customer with GCIF 'EGY12345'.
Expected Result	Customer 'Cairo Holdings' is onboarded with all Egypt-mandated entitlements assigned.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the customer onboarding form.
Test Data	{ "url": "https://admin.corpapp.com/onboarding", "browser": "Chrome" }
Expected Behavior	Customer onboarding form is displayed.

Step Number	2
Description	Enter customer details with GCIF set to Egypt.
Test Data	{ "customer_name": "Cairo Holdings", "GCIF": "EGY12345", "country": "Egypt", "contact_email": "admin@cairoholdings.eg" }
Expected Behavior	Form accepts Egypt-specific GCIF and country.

Step Number	3
Description	Proceed to product entitlement section.
Test Data	{ "navigation": "Next" }
Expected Behavior	Product entitlement section loads with default values.

Step Number	4
Description	Verify that Governmental Payments, Tax Collection, and Custom Collection are pre-selected.
Test Data	{ "expected_entitlements": "Governmental Payments, Tax Collection, Custom Collection" }
Expected Behavior	All Egypt-required entitlements are pre-selected.

Step Number	5
Description	Submit onboarding form.
Test Data	{ "action": "Submit" }
Expected Behavior	Customer is created and entitlements are saved.

Test Data Summary:

{ "overall_input": "GCIF: EGY12345, Country: Egypt, Default entitlements", "key_parameters": "GCIF, Country, Entitlement selection" }

Validation Criteria	Entitlements match Egypt defaults; Customer record created
Dependencies	Admin portal availability; Entitlement configuration for Egypt
Notes	Validates localization and entitlement logic for Egypt.

Test Case 12: File Upload - Accept Only Supported File Types for Beneficiary Import

Test ID	TC_012
Module	Beneficiary Management > Bulk Upload
Category	Functional
Priority	High
Test Type	Negative
Risk Level	High
Estimated Time	6 minutes
Description	Ensure that only supported file types (CSV, XLSX) are accepted during beneficiary bulk upload.
Objective	Prevent unsupported file types from being uploaded.
Preconditions	Admin user is logged in.; Beneficiary upload page is accessible.
Expected Result	Only CSV and XLSX files are accepted; all other file types are rejected.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the beneficiary bulk upload page.
Test Data	{ "url": "https://admin.corpapp.com/beneficiaries/upload", "browser": "Firefox" }
Expected Behavior	Bulk upload interface is displayed.

Step Number	2
Description	Attempt to upload a PDF file.
Test Data	{ "file_name": "beneficiaries_list.pdf", "file_type": "application/pdf", "file_size": "120KB" }
Expected Behavior	System rejects the file and displays an error: 'Unsupported file type.'

Step Number	3

Description	Attempt to upload a CSV file.

Test Data	{
	"file_name": "beneficiaries_list.csv",
	"file_type": "text/csv",
	"file_size": "45KB"
	}

Expected Behavior	System accepts the file and proceeds to validation.

Step Number	4
Description	Attempt to upload an XLSX file.

Test Data	{
	"file_name": "beneficiaries_list.xlsx",
	"file_type": "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
	"file_size": "60KB"
	}

Expected Behavior	System accepts the file and proceeds to validation.

Test Data Summary:

{ "overall_input": "PDF, CSV, XLSX files for upload", "key_parameters": "File type, File name" }

Validation Criteria	Only supported file types accepted; Clear error for unsupported types
Dependencies	File validation logic; UI file input
Notes	Covers file type validation for security and data integrity.

Test Case 13: API - Add New Governmental Payment Type (Future-Proofing)

Test ID	TC_013
Module	Admin Portal > Payment Type Management
Category	Integration
Priority	High
Test Type	Positive/Negative
Risk Level	Medium
Estimated Time	7 minutes
Description	Test the API endpoint for adding a new governmental payment type, ensuring the system supports fu
Objective	Validate extensibility and correct API behavior for new payment types.
Preconditions	Admin API credentials are available.; No payment type named 'Municipal Fees' exists.
Expected Result	New payment type is added and visible; duplicates are rejected.

Detailed Test Steps with Data:

Step Number	1

Description	Send POST request to /api/payment-types with valid payload.

Test Data	{
	 "endpoint": "/api/payment-types",
	 "method": "POST",
	 "payload": "{\"name\": \"Municipal Fees\", \"category\": \"Governmental\", \"default_entitlement\": true, \"country\": \"Egy
	 "headers": "{\"Authorization\": \"Bearer admin-token-123\"}"
	 }

Expected Behavior	API responds with 201 Created and new payment type ID.

Step Number	2
Description	Send GET request to /api/payment-types to verify addition.

Test Data	{
	 "endpoint": "/api/payment-types",
	 "method": "GET",
	 "headers": "{\"Authorization\": \"Bearer admin-token-123\"}"
	 }

Expected Behavior	'Municipal Fees' appears in the payment type list.

Step Number	3
Description	Attempt to add duplicate payment type.

Test Data	{
	 "endpoint": "/api/payment-types",
	 "method": "POST",
	 "payload": "{\"name\": \"Municipal Fees\", \"category\": \"Governmental\", \"default_entitlement\": false, \"country\": \"Eg
	 "headers": "{\"Authorization\": \"Bearer admin-token-123\"}"
	 }

Expected Behavior	API responds with 409 Conflict and error message.

Test Data Summary:

{ "overall_input": "API payload for new payment type", "key_parameters": "name, category, country, default_entitlement" }

Validation Criteria	Payment type added; Duplicate prevented
Dependencies	API endpoint availability; Admin token
Notes	Validates future-proofing and API idempotency.

Test Case 14: Role-Based Access Control - Unauthorized User Cannot Add Beneficiary

Test ID	TC_014
Module	Security > Role-Based Permissions
Category	Security
Priority	Critical
Test Type	Negative

Risk Level	Critical
Estimated Time	5 minutes
Description	Verify that users without admin or beneficiary management roles cannot add beneficiaries.
Objective	Ensure RBAC is enforced for beneficiary management.
Preconditions	User 'readonly_user' exists with no beneficiary permissions.; Readonly user is logged in.
Expected Result	Readonly users cannot add beneficiaries via UI or API.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the beneficiary management page.
Test Data	{ "url": "https://admin.corpapp.com/beneficiaries", "username": "readonly_user" }
Expected Behavior	Beneficiary management page is displayed with limited options.

Step Number	2
Description	Attempt to access 'Add Beneficiary' form.
Test Data	{ "action": "Click 'Add Beneficiary' button" }
Expected Behavior	Access is denied with error: 'Insufficient permissions.'

Step Number	3
Description	Attempt to POST to /api/beneficiaries directly.
Test Data	{ "endpoint": "/api/beneficiaries", "method": "POST", "payload": "{\"name\": \"Test Beneficiary\", \"account_number\": \"1234567890\", \"bank\": \"National Bank\"}", "headers": "{\"Authorization\": \"Bearer readonly-token-456\"}" }
Expected Behavior	API responds with 403 Forbidden.

Test Data Summary:

{ "overall_input": "Readonly user, beneficiary add attempt", "key_parameters": "User role, API token" }

Validation Criteria	Access denied for unauthorized users
Dependencies	RBAC configuration; API endpoint
Notes	Covers both UI and API RBAC enforcement.

Test Case 15: Boundary Test - Maximum Length for Customer Name Field

Test ID	TC_015
Module	Customer Onboarding > Field Validation
Category	Boundary
Priority	Medium
Test Type	Boundary
Risk Level	Medium
Estimated Time	4 minutes
Description	Test the upper boundary for the customer name field to ensure system accepts up to the maximum a
Objective	Validate field length enforcement.
Preconditions	Onboarding form is accessible.
Expected Result	System enforces 50-character limit for customer name.

Detailed Test Steps with Data:

Step Number	1
Description	Enter a customer name with exactly 50 characters.
Test Data	{ "customer_name": "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMN123456789012" }
Expected Behavior	Field accepts the input without error.

Step Number	2
Description	Enter a customer name with 51 characters.
Test Data	{ "customer_name": "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMN123456789012X" }
Expected Behavior	Field displays error: 'Maximum 50 characters allowed.'

Test Data Summary:

{ "overall_input": "50-char and 51-char strings", "key_parameters": "customer_name" }

Validation Criteria	No error for 50 chars; Error for 51 chars
Dependencies	Field validation logic
Notes	Ensures data integrity for customer name field.

Test Case 16: Auto-Rejection Workflow - Transaction Not Released in 45 Days

Test ID	TC_016
Module	Transaction Processing > Workflow Automation
Category	Functional
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	10 minutes
Description	Verify that transactions not approved or released within 45 days are automatically rejected.
Objective	Ensure auto-rejection logic is enforced as per business rules.
Preconditions	Transaction exists in 'Pending Release' status for 44 days.
Expected Result	Transaction is auto-rejected after 45 days of inactivity.

Detailed Test Steps with Data:

Step Number	1
Description	Identify a transaction pending release for 44 days.
Test Data	{ "transaction_id": "TXN987654", "status": "Pending Release", "created_date": "2024-05-20" }
Expected Behavior	Transaction is found and status is 'Pending Release'.

Step Number	2
Description	Advance system date by 2 days to simulate 46 days since creation.
Test Data	{ "system_date": "2024-07-05" }
Expected Behavior	System date is updated; transaction is now 46 days old.

Step Number	3
Description	Trigger scheduled job for auto-rejection.
Test Data	{ "job_name": "AutoRejectionJob" }
Expected Behavior	Job runs and processes pending transactions.

Step Number	4
Description	Check transaction status after job execution.
Test Data	{ "transaction_id": "TXN987654" }
Expected Behavior	Transaction status is updated to 'Rejected'.

Test Data Summary:

{ "overall_input": "Transaction pending 44 days, system date advanced", "key_parameters": "transaction_id, system_date" }

Validation Criteria	Transaction is rejected after 45 days
Dependencies	Scheduled job configuration; Date manipulation capability
Notes	Validates business-critical workflow automation.

Test Case 17: Usability - Admin Portal Add Beneficiary Flow

Test ID	TC_017
Module	Admin Portal > Beneficiary Management
Category	Usability
Priority	Medium
Test Type	Positive
Risk Level	Low
Estimated Time	6 minutes
Description	Assess the usability and clarity of the add beneficiary flow in the admin portal.
Objective	Ensure the process is intuitive and all mandatory fields are clearly marked.
Preconditions	Admin user is logged in.
Expected Result	Admin can intuitively add a beneficiary; all required fields are clear.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to 'Add Beneficiary' page.
Test Data	{ "url": "https://admin.corpapp.com/beneficiaries/add", "browser": "Edge" }
Expected Behavior	'Add Beneficiary' form loads with all input fields visible.

Step Number	2
Description	Verify all mandatory fields are marked with a red asterisk.
Test Data	{ "fields": "Beneficiary Name, Account Number, Bank Name" }
Expected Behavior	Mandatory fields are visually indicated.

Step Number	3
Description	Enter valid beneficiary details.

Test Data	{ "beneficiary_name": "Delta Trading", "account_number": "9876543210", "bank_name": "Commercial Bank", "email": "finance@deltatrading.com" }
Expected Behavior	Form accepts input and enables 'Save' button.

Step Number	4
Description	Submit the form.
Test Data	{ "action": "Click Save" }
Expected Behavior	Beneficiary is added and confirmation message is shown.

Test Data Summary:

{ "overall_input": "Beneficiary details, UI navigation", "key_parameters": "Mandatory fields, input values" }

Validation Criteria	Clear mandatory field indicators; Smooth form submission
Dependencies	Admin portal UI
Notes	Focuses on user experience and field clarity.

Test Case 18: Error Handling - Invalid Amount Field Format

Test ID	TC_018
Module	Transaction Processing > Field Validation
Category	Error Handling
Priority	Medium
Test Type	Negative/Positive
Risk Level	Medium
Estimated Time	5 minutes
Description	Test system response when entering an invalid amount (more than two decimal places) in the transa
Objective	Ensure proper error messages for invalid amount formats.
Preconditions	Transaction form is accessible.
Expected Result	System rejects invalid formats and accepts valid amounts.

Detailed Test Steps with Data:

Step Number	1
Description	Enter amount with three decimal places.

Test Data	{ "amount": "1234.567" }
Expected Behavior	Form displays error: 'Amount must have at most two decimal places.'

Step Number	2
Description	Enter amount with letters included.
Test Data	{ "amount": "12AB.34" }
Expected Behavior	Form displays error: 'Amount must be a valid number.'

Step Number	3
Description	Enter valid amount.
Test Data	{ "amount": "9876.54" }
Expected Behavior	Form accepts the amount and enables submission.

Test Data Summary:

{ "overall_input": "Amounts with invalid and valid formats", "key_parameters": "amount" }

Validation Criteria	Error for invalid formats; Acceptance of valid format
Dependencies	Amount field validation
Notes	Covers both negative and positive scenarios for amount input.

Test Case 19: Performance - Bulk Beneficiary Upload with 10,000 Records

Test ID	TC_019
Module	Beneficiary Management > Bulk Upload
Category	Performance
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	10 minutes
Description	Assess system performance and response time when uploading a large beneficiary file (10,000 reco
Objective	Ensure the system can handle high-volume uploads efficiently.
Preconditions	Admin user is logged in.; Bulk upload feature is enabled.
Expected Result	System processes 10,000 records within 3 minutes without errors.

Detailed Test Steps with Data:

Step Number	1
Description	Prepare a CSV file with 10,000 beneficiary records.
Test Data	{ "file_name": "beneficiaries_10k.csv", "file_type": "text/csv", "record_count": "10000", "file_size": "2MB" }
Expected Behavior	File is ready for upload.

Step Number	2
Description	Upload the CSV file via the bulk upload interface.
Test Data	{ "file_path": "/testdata/beneficiaries_10k.csv" }
Expected Behavior	System begins processing the file.

Step Number	3
Description	Measure time taken for upload and processing.
Test Data	{ "start_time": "10:00:00", "end_time": "10:02:30" }
Expected Behavior	Processing completes within 3 minutes.

Step Number	4
Description	Verify that all 10,000 records are imported successfully.
Test Data	{ "expected_count": "10000" }
Expected Behavior	System confirms 10,000 beneficiaries added.

Test Data Summary:

{ "overall_input": "CSV file with 10,000 records", "key_parameters": "file_size, record_count" }

Validation Criteria	Upload completes within SLA; All records imported
Dependencies	Bulk upload infrastructure
Notes	Validates scalability and performance under load.

Test Case 20: SWIFT Compliance - Allowed Special Characters in Alphanumeric Fields

Test ID	TC_020

Module	Field Validation > SWIFT Compliance
Category	Functional
Priority	Medium
Test Type	Negative/Positive
Risk Level	Medium
Estimated Time	5 minutes
Description	Test that only SWIFT-compliant special characters are accepted in alphanumeric fields.
Objective	Ensure compliance with SWIFT character rules.
Preconditions	Form with alphanumeric field is accessible.
Expected Result	Only SWIFT-compliant characters are accepted; others are rejected.

Detailed Test Steps with Data:

Step Number	1
Description	Enter a value with allowed SWIFT characters.
Test Data	{ "input_value": "Alpha-Num_123./?:()'-,+" }
Expected Behavior	Field accepts the input without error.

Step Number	2
Description	Enter a value with disallowed characters (e.g., #, $, %).
Test Data	{ "input_value": "Invalid#Value$%" }
Expected Behavior	Field displays error: 'Invalid character(s) detected.'

Step Number	3
Description	Enter a value with only letters and numbers.
Test Data	{ "input_value": "Test123" }
Expected Behavior	Field accepts the input.

Test Data Summary:

{ "overall_input": "Inputs with allowed/disallowed special characters", "key_parameters": "input_value" }

Validation Criteria	Allowed characters accepted; Disallowed characters rejected
Dependencies	SWIFT character validation logic
Notes	Ensures compliance with international standards.

Test Case 21: Verify Mandatory Field Enforcement During Customer Onboarding

Test ID	TC_021
Module	Corporate Module > Customer Onboarding
Category	Functional
Priority	Critical
Test Type	Negative
Risk Level	High
Estimated Time	10 minutes
Description	Ensure all mandatory fields are enforced during customer onboarding and appropriate errors are sho
Objective	Validate that the system enforces mandatory field requirements and provides clear error messages.
Preconditions	User is logged in as onboarding admin; Onboarding form is accessible
Expected Result	System enforces all mandatory fields and provides clear, field-specific error messages.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the customer onboarding form.
Test Data	{ "url": "https://corp-portal.example.com/onboarding" }
Expected Behavior	Onboarding form loads successfully.

Step Number	2
Description	Leave all mandatory fields empty and attempt to submit the form.
Test Data	{ "company_name": "", "GCIF": "", "country": "", "contact_email": "" }
Expected Behavior	System displays error messages for each empty mandatory field.

Step Number	3
Description	Enter valid data in all fields except 'GCIF' and submit.
Test Data	{ "company_name": "Acme Corp", "GCIF": "", "country": "Egypt", "contact_email": "admin@acme.com" }
Expected Behavior	System displays an error message indicating 'GCIF' is required.

Step Number	4
Description	Fill all mandatory fields with valid data and submit.
Test Data	{ "company_name": "Acme Corp", "GCIF": "EG123456", "country": "Egypt", "contact_email": "admin@acme.com" }
Expected Behavior	Form submission is successful and onboarding proceeds to the next step.

Test Data Summary:

{ "overall_input": "Onboarding form fields", "key_parameters": "company_name, GCIF, country, contact_email" }

Validation Criteria	Error messages for missing mandatory fields; Successful submission when all fields are filled
Dependencies	Onboarding form UI; Field validation logic
Notes	Test covers both negative and positive scenarios for mandatory fields.

Test Case 22: Integration: Product Entitlement Assignment Based on Country and GCIF

Test ID	TC_022
Module	Corporate Module > Product Entitlement
Category	Integration
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	12 minutes
Description	Verify that product entitlements are correctly assigned based on the customer's country and GCIF lev
Objective	Ensure entitlement logic is correctly integrated with onboarding data.
Preconditions	Customer onboarding form is completed with valid data; Entitlement configuration is set up
Expected Result	Product entitlements are assigned according to country and GCIF logic.

Detailed Test Steps with Data:

Step Number	1
Description	Onboard a new customer with GCIF 'EG123456' and country 'Egypt'.
Test Data	{ "GCIF": "EG123456", "country": "Egypt" }

Expected Behavior	Customer profile is created with Egypt-specific attributes.

Step Number	2
Description	Check assigned product entitlements for the new customer.
Test Data	{ "customer_id": "CUST1001" }
Expected Behavior	Entitlements include default Governmental Payments, Tax Collection, and Custom Collection for Egypt.

Step Number	3
Description	Onboard a customer with GCIF 'US987654' and country 'USA'.
Test Data	{ "GCIF": "US987654", "country": "USA" }
Expected Behavior	Customer profile is created with USA-specific attributes.

Step Number	4
Description	Check assigned product entitlements for the USA customer.
Test Data	{ "customer_id": "CUST1002" }
Expected Behavior	Entitlements reflect default values for USA (e.g., no Egypt-specific products).

Test Data Summary:

{ "overall_input": "GCIF and country values", "key_parameters": "GCIF, country" }

Validation Criteria	Correct entitlement mapping for Egypt; Correct entitlement mapping for non-Egypt
Dependencies	Entitlement configuration; Onboarding workflow
Notes	Validates integration between onboarding and entitlement modules.

Test Case 23: Error Handling: Invalid Amount Field Entry in Payment Module

Test ID	TC_023
Module	Corporate Module > Payment Processing
Category	Error Handling
Priority	High
Test Type	Negative
Risk Level	High
Estimated Time	8 minutes
Description	Test system response to invalid data in amount fields (e.g., non-numeric, excessive decimals, negati

Objective	Ensure robust error handling for amount field inputs.
Preconditions	User is logged in and has access to payment module
Expected Result	System rejects invalid amount entries and accepts valid ones.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the payment initiation form.
Test Data	{ "url": "https://corp-portal.example.com/payments/initiate" }
Expected Behavior	Payment form loads successfully.

Step Number	2
Description	Enter non-numeric value in the amount field and attempt to submit.
Test Data	{ "amount": "abcde" }
Expected Behavior	System displays error: 'Amount must be a numeric value with two decimals.'

Step Number	3
Description	Enter a negative value in the amount field and submit.
Test Data	{ "amount": "-100.00" }
Expected Behavior	System displays error: 'Amount cannot be negative.'

Step Number	4
Description	Enter a value with more than two decimal places and submit.
Test Data	{ "amount": "100.123" }
Expected Behavior	System displays error: 'Amount must have at most two decimal places.'

Step Number	5
Description	Enter a valid amount and submit.
Test Data	{ "amount": "2500.50" }
Expected Behavior	Amount is accepted and payment proceeds to next step.

Test Data Summary:

{ "overall_input": "Amount field values", "key_parameters": "amount" }

Validation Criteria	Error messages for invalid input; Acceptance of valid input
Dependencies	Payment form validation logic
Notes	Covers numeric, decimal, and negative value validation.

Test Case 24: Security: Role-Based Access Control for Admin Portal Beneficiary Management

Test ID	TC_024
Module	Admin Portal > Beneficiary Management
Category	Security
Priority	Critical
Test Type	Negative
Risk Level	Critical
Estimated Time	10 minutes
Description	Verify that only users with admin roles can add, edit, or remove beneficiaries in the admin portal.
Objective	Ensure RBAC is enforced for beneficiary management.
Preconditions	Admin and non-admin users exist; Admin portal is accessible
Expected Result	Only admin users can manage beneficiaries; non-admins are denied access.

Detailed Test Steps with Data:

Step Number	1
Description	Login as a non-admin user and attempt to access beneficiary management.
Test Data	{ "username": "user123", "password": "UserPass!23" }
Expected Behavior	Access is denied and user is redirected or shown an error.

Step Number	2
Description	Login as an admin user.
Test Data	{ "username": "admin01", "password": "AdminPass#2024" }
Expected Behavior	Admin dashboard is displayed.

Step Number	3
Description	Navigate to beneficiary management section.

Test Data	{ "section": "Beneficiaries" }
Expected Behavior	Beneficiary management UI is accessible.

Step Number	4
Description	Add a new beneficiary with valid details.
Test Data	{ "beneficiary_name": "John Doe", "account_number": "1234567890", "bank": "Bank of Egypt" }
Expected Behavior	Beneficiary is added successfully and appears in the list.

Step Number	5
Description	Remove the newly added beneficiary.
Test Data	{ "beneficiary_name": "John Doe" }
Expected Behavior	Beneficiary is removed from the list.

Test Data Summary:

{ "overall_input": "User credentials, beneficiary details", "key_parameters": "role, beneficiary_name" }

Validation Criteria	Access denied for non-admins; Full access for admins
Dependencies	RBAC implementation; Beneficiary management UI
Notes	Validates enforcement of RBAC for sensitive admin functions.

Test Case 25: Boundary: Maximum Character Limit in Alphanumeric Fields

Test ID	TC_025
Module	Corporate Module > Field Validation
Category	Boundary
Priority	Medium
Test Type	Boundary
Risk Level	Medium
Estimated Time	7 minutes
Description	Test alphanumeric field with maximum allowed characters and one character above the limit.
Objective	Ensure field enforces maximum character limits as specified.
Preconditions	User is logged in; Form with alphanumeric field is accessible

Expected Result	Field enforces character limit and displays error for overflow.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the form containing the alphanumeric field (max 12 chars).
Test Data	{ "url": "https://corp-portal.example.com/profile/edit" }
Expected Behavior	Form loads successfully.

Step Number	2
Description	Enter exactly 12 alphanumeric characters in the field.
Test Data	{ "alphanumeric_field": "ABC123XYZ789" }
Expected Behavior	Field accepts input and no error is shown.

Step Number	3
Description	Enter 13 alphanumeric characters in the field.
Test Data	{ "alphanumeric_field": "ABC123XYZ789Q" }
Expected Behavior	System displays error: 'Maximum 12 characters allowed.'

Step Number	4
Description	Enter 11 alphanumeric characters in the field.
Test Data	{ "alphanumeric_field": "ABC123XYZ78" }
Expected Behavior	Field accepts input and no error is shown.

Test Data Summary:

{ "overall_input": "Alphanumeric field values", "key_parameters": "alphanumeric_field" }

Validation Criteria	Error for overflow; Acceptance at and below limit
Dependencies	Field validation logic
Notes	Covers boundary condition for alphanumeric field.

Test Case 26: Performance: Bulk Onboarding of Customers with Entitlement Assignment

Test ID	TC_026

Module	Corporate Module > Customer Onboarding
Category	Performance
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	15 minutes
Description	Test system performance when onboarding 100 customers in bulk, each with entitlement assignmen
Objective	Ensure system can handle bulk onboarding within acceptable time limits.
Preconditions	Bulk onboarding feature is enabled; Test data file with 100 customer records is prepared
Expected Result	Bulk onboarding completes within performance targets and entitlements are correctly assigned.

Detailed Test Steps with Data:

Step Number	1
Description	Prepare a CSV file with 100 customer records, each with unique GCIF and country.

Test Data	{
	  "file_name": "bulk_onboarding_100.csv",
	  "record_count": "100"
	}

Expected Behavior	CSV file is ready for upload.

Step Number	2
Description	Navigate to the bulk onboarding section.

Test Data	{
	  "url": "https://corp-portal.example.com/onboarding/bulk"
	}

Expected Behavior	Bulk onboarding UI is displayed.

Step Number	3
Description	Upload the CSV file and start the onboarding process.

Test Data	{
	  "file_path": "/testdata/bulk_onboarding_100.csv"
	}

Expected Behavior	System accepts file and begins processing.

Step Number	4
Description	Monitor processing time and system resource usage.

Test Data	{
	  "expected_max_time_seconds": "120"
	}

Expected Behavior	All 100 customers are onboarded with correct entitlements within 2 minutes.

Step Number	5

Description	Verify a random sample of 5 customers for correct entitlement assignment.
Test Data	{ "sample_customer_ids": "CUST2001, CUST2020, CUST2050, CUST2080, CUST2100" }
Expected Behavior	Each sampled customer has correct product entitlements.

Test Data Summary:

{ "overall_input": "CSV file with 100 records", "key_parameters": "GCIF, country" }

Validation Criteria	Processing time under 2 minutes; Correct entitlements for all customers
Dependencies	Bulk onboarding feature; Entitlement assignment logic
Notes	Validates both performance and correctness under load.

Test Case 27: Usability: Admin Portal - Add New Governmental Payment Type

Test ID	TC_027
Module	Admin Portal > Payment Type Management
Category	Usability
Priority	Medium
Test Type	Positive
Risk Level	Medium
Estimated Time	8 minutes
Description	Verify that the admin can add a new governmental payment type via the UI and it appears in the sele
Objective	Ensure UI supports adding new payment types and reflects changes immediately.
Preconditions	Admin user is logged in; Payment type management UI is accessible
Expected Result	Admin can add new payment types and changes are reflected in the UI.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the payment type management section.
Test Data	{ "url": "https://admin-portal.example.com/payments/types" }
Expected Behavior	Payment type management UI loads.

Step Number	2
Description	Click 'Add New Payment Type' and enter details.

Test Data	{ "payment_type_name": "Municipal Fees", "description": "Fees for municipal services", "active": "True" }
Expected Behavior	Form for new payment type appears and accepts input.

Step Number	3
Description	Submit the new payment type.
Test Data	{ "submit": "True" }
Expected Behavior	System saves new payment type and displays success message.

Step Number	4
Description	Verify that 'Municipal Fees' appears in the list of payment types.
Test Data	{ "expected_payment_type": "Municipal Fees" }
Expected Behavior	'Municipal Fees' is listed and selectable for entitlement assignment.

Test Data Summary:

{ "overall_input": "New payment type details", "key_parameters": "payment_type_name, description" }

Validation Criteria	New type appears in list; No UI errors
Dependencies	Admin portal UI; Payment type management logic
Notes	Ensures admin flexibility for future additions.

Test Case 28: Functional: Auto-Rejection of Unapproved Transactions After 45 Days

Test ID	TC_028
Module	Corporate Module > Transaction Workflow
Category	Functional
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	9 minutes
Description	Verify that transactions not approved or released within 45 days are auto-rejected by the system.
Objective	Ensure compliance with transaction handling rules.
Preconditions	Transaction workflow is operational; Test transaction is created with old timestamp

Expected Result	Transactions older than 45 days are auto-rejected and initiators are notified.

Detailed Test Steps with Data:

Step Number	1
Description	Create a transaction with a creation date 46 days in the past.
Test Data	{ "transaction_id": "TXN9001", "creation_date": "2024-05-15" }
Expected Behavior	Transaction is created and pending approval.

Step Number	2
Description	Run the auto-rejection batch job.
Test Data	{ "batch_job": "auto_reject_transactions" }
Expected Behavior	Batch job processes transactions older than 45 days.

Step Number	3
Description	Check the status of the transaction after batch job execution.
Test Data	{ "transaction_id": "TXN9001" }
Expected Behavior	Transaction status is updated to 'Auto-Rejected'.

Step Number	4
Description	Verify that a notification is sent to the transaction initiator.
Test Data	{ "initiator_email": "user@corp.com" }
Expected Behavior	Initiator receives an email notification about auto-rejection.

Test Data Summary:

{ "overall_input": "Transaction with old creation date", "key_parameters": "creation_date, transaction_id" }

Validation Criteria	Auto-rejection after 45 days; Notification sent
Dependencies	Batch job scheduler; Notification system
Notes	Ensures compliance with business rules.

Test Case 29: Integration: Default Sub-Product Selection Based on Entitlement Configuration

Test ID	TC_029
Module	Corporate Module > Product Entitlement
Category	Integration
Priority	Medium
Test Type	Positive
Risk Level	Medium
Estimated Time	8 minutes
Description	Verify that default sub-products are selected based on entitlement configuration during onboarding.
Objective	Ensure correct default selections for sub-products.
Preconditions	Entitlement configuration is set with default sub-products; Onboarding form is accessible
Expected Result	Default sub-products are pre-selected and user can modify selection as per rules.

Detailed Test Steps with Data:

Step Number	1
Description	Onboard a customer with entitlement to 'Governmental Payments'.
Test Data	{ "GCIF": "EG654321", "entitled_products": "Governmental Payments" }
Expected Behavior	Customer profile is created with 'Governmental Payments' entitlement.

Step Number	2
Description	Access the sub-product selection screen during onboarding.
Test Data	{ "screen": "Sub-Product Selection" }
Expected Behavior	Sub-product selection UI is displayed.

Step Number	3
Description	Verify that default sub-products (e.g., 'Tax Collection', 'Custom Collection') are pre-selected.
Test Data	{ "expected_defaults": "Tax Collection, Custom Collection" }
Expected Behavior	Specified sub-products are pre-selected according to configuration.

Step Number	4
Description	Deselect a default sub-product and attempt to proceed.
Test Data	{ "deselected_sub_product": "Custom Collection" }
Expected Behavior	System allows proceeding if at least one sub-product remains selected.

Test Data Summary:

{ "overall_input": "Entitlement and sub-product selection", "key_parameters": "entitled_products, expected_defaults" }

Validation Criteria	Correct default selection; User can modify selection
Dependencies	Entitlement configuration; Onboarding UI
Notes	Validates defaulting logic for sub-products.

Test Case 30: Error Handling: SWIFT Compliance Character Validation in Free Format Fields

Test ID	TC_030
Module	Corporate Module > Field Validation
Category	Error Handling
Priority	Medium
Test Type	Negative
Risk Level	Medium
Estimated Time	7 minutes
Description	Test that free format fields accept only SWIFT-compliant characters and reject invalid ones.
Objective	Ensure compliance with SWIFT character restrictions.
Preconditions	User is logged in; Form with free format field is accessible
Expected Result	Field accepts only SWIFT-compliant characters and rejects others.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the form with the free format field.
Test Data	{ "url": "https://corp-portal.example.com/payments/details" }
Expected Behavior	Form loads successfully.

Step Number	2
Description	Enter a string with only SWIFT-compliant characters.
Test Data	{ "free_format_field": "Payment for INV#12345/2024" }
Expected Behavior	Field accepts input and no error is shown.

Step Number	3
Description	Enter a string with a non-compliant character (e.g., emoji).

Test Data	{ "free_format_field": "Payment for order \ud83d\ude0a" }
Expected Behavior	System displays error: 'Invalid character detected. Only SWIFT-compliant characters allowed.'

Step Number	4
Description	Enter a string with a disallowed special character (e.g., backtick).
Test Data	{ "free_format_field": "Payment for \`Order\`" }
Expected Behavior	System displays error: 'Invalid character detected. Only SWIFT-compliant characters allowed.'

Test Data Summary:

{ "overall_input": "Free format field values", "key_parameters": "free_format_field" }

Validation Criteria	Error for non-compliant input; Acceptance of compliant input
Dependencies	SWIFT character validation logic
Notes	Ensures regulatory compliance for free format fields.

Test Case 31: Successful Customer Onboarding with Full Product Entitlements

Test ID	TC_031
Module	Corporate Module > Customer Onboarding
Category	Functional
Priority	Critical
Test Type	Positive
Risk Level	Critical
Estimated Time	10 minutes
Description	Validates that a new customer can be onboarded with all available payment modules and sub-produc
Objective	Ensure onboarding process assigns correct entitlements and default values per documentation.
Preconditions	Admin user is logged into the admin portal.; All payment modules and sub-products are active in the
Expected Result	Customer is onboarded with all entitlements and correct default values for Egypt GCIF.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the customer onboarding section and click 'Add New Customer'.

Test Data	{ "browser": "Chrome", "admin_username": "admin01", "portal_url": "https://corp-portal.example.com/admin" }
Expected Behavior	Customer onboarding form is displayed.

Step Number	2
Description	Enter mandatory customer details and select Egypt as GCIF level.
Test Data	{ "customer_name": "Nile Holdings", "GCIF_level": "Egypt", "company_registration": "EG123456789", "contact_email": "contact@nileholdings.com" }
Expected Behavior	Form accepts input and enables entitlement selection.

Step Number	3
Description	Assign all available payment modules and sub-products to the customer.
Test Data	{ "modules_selected": "Governmental Payments, Tax Collection, Custom Collection, Universal Collection", "sub_products": "Adhoc Bill, Next Authorizer, Verifier Intervention, Releaser Intervention" }
Expected Behavior	All modules and sub-products are selected and displayed as assigned.

Step Number	4
Description	Verify that default values are set for Egypt GCIF (e.g., Adhoc Bill = Yes, Verifier Intervention = No).
Test Data	{ "GCIF_level": "Egypt", "expected_defaults": "{\"Adhoc Bill\": \"Yes\", \"Verifier Intervention\": \"No\"}" }
Expected Behavior	Default values for Egypt are auto-filled as per configuration.

Step Number	5
Description	Submit the onboarding form.
Test Data	{ "submit_action": "True" }
Expected Behavior	Customer is created and entitlements are reflected in the customer profile.

Test Data Summary:

{ "overall_input": "Full customer profile, Egypt GCIF, all modules and sub-products selected", "key_parameters": "GCIF_level=Egypt, modules, sub_products, default values" }

Validation Criteria	All entitlements assigned; Default values match Egypt configuration; Customer profile reflects selections
Dependencies	Active modules and sub-products; Admin portal access
Notes	Covers onboarding and entitlement mapping for high-value customers.

Test Case 32: Auto-Rejection of Pending Transactions After 45 Days

Test ID	TC_032
Module	Corporate Module > Transaction Workflow
Category	Functional
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	7 minutes
Description	Ensures transactions not approved or released within 45 days are automatically rejected.
Objective	Verify workflow automation for transaction expiry.
Preconditions	A transaction is pending approval for more than 45 days.; Auto-rejection feature is enabled.
Expected Result	Transaction is auto-rejected after 45 days with correct status and reason.

Detailed Test Steps with Data:

Step Number	1
Description	Create a new payment transaction and set its creation date to 46 days ago.
Test Data	{ "transaction_type": "Tax Payment", "amount": "5000.0", "currency": "EGP", "creation_date": "2024-05-15" }
Expected Behavior	Transaction is created and appears in pending state.

Step Number	2
Description	Do not approve or release the transaction.
Test Data	{ "action": "No action" }
Expected Behavior	Transaction remains pending.

Step Number	3
Description	Trigger the auto-rejection batch process.
Test Data	{ "batch_job": "AutoRejectPendingTransactions", "run_date": "2024-06-30" }
Expected Behavior	Batch process scans for transactions older than 45 days.

Step Number	4
Description	Check the status of the transaction after batch execution.

Test Data	{
	  "transaction_id": "TXN123456"
	}

Expected Behavior	Transaction status is updated to 'Rejected' with reason 'Auto-rejection: 45 days expired'.

Test Data Summary:

{ "overall_input": "Transaction with creation date >45 days ago", "key_parameters": "creation_date, batch_job, transaction_id" }

Validation Criteria	Transaction is rejected after 45 days; Rejection reason is logged
Dependencies	Batch job scheduler; Transaction workflow
Notes	Validates compliance with workflow handling rules.

Test Case 33: Boundary Test: Amount Field Maximum Value

Test ID	TC_033
Module	Corporate Module > Payment Entry
Category	Boundary
Priority	High
Test Type	Boundary
Risk Level	Medium
Estimated Time	6 minutes
Description	Tests the upper boundary for the AMT (XX, XX) field, ensuring system accepts the maximum allowe
Objective	Ensure amount field enforces maximum value and decimal precision.
Preconditions	User is logged in and has access to payment entry.
Expected Result	System accepts and processes maximum allowed amount with correct precision.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the payment entry form.

Test Data	{
	  "user_role": "FinanceUser",
	  "portal_url": "https://corp-portal.example.com/payments"
	}

Expected Behavior	Payment entry form is displayed.

Step Number	2
Description	Enter the maximum allowed amount in the 'Amount' field.

Test Data	{ "amount": "99999999.99" }
Expected Behavior	Amount field accepts the value with two decimals.

Step Number	3
Description	Complete remaining mandatory fields and submit the payment.
Test Data	{ "beneficiary": "ABC Supplies", "payment_type": "Custom Collection", "reference": "INV20240630" }
Expected Behavior	Payment is submitted successfully.

Step Number	4
Description	Verify the transaction record for correct amount and status.
Test Data	{ "transaction_reference": "INV20240630" }
Expected Behavior	Transaction record shows amount 99999999.99 and status 'Pending Approval'.

Test Data Summary:

{ "overall_input": "Amount=99999999.99, valid beneficiary, payment type, reference", "key_parameters": "amount, payment_type, beneficiary" }

Validation Criteria	Amount field enforces max value; Transaction is created with correct amount
Dependencies	Amount field configuration
Notes	Covers numeric field boundary for payment entry.

Test Case 34: Negative Test: Mandatory Field Validation on Beneficiary Addition

Test ID	TC_034
Module	Admin Portal > Beneficiary Management
Category	Error Handling
Priority	Medium
Test Type	Negative
Risk Level	Medium
Estimated Time	4 minutes
Description	Ensures system enforces mandatory field validation when admin adds a new beneficiary.
Objective	Validate error handling for missing required fields.
Preconditions	Admin user is logged into admin portal.

Expected Result	System prevents submission and displays error for missing mandatory fields.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to 'Add Beneficiary' in the admin portal.
Test Data	{ "admin_username": "admin02", "portal_url": "https://corp-portal.example.com/admin/beneficiaries" }
Expected Behavior	Add Beneficiary form is displayed.

Step Number	2
Description	Leave the 'Beneficiary Name' field blank and fill other fields.
Test Data	{ "beneficiary_name": "", "account_number": "123456789012", "bank_code": "CIBE1234" }
Expected Behavior	Form highlights 'Beneficiary Name' as required.

Step Number	3
Description	Attempt to submit the form.
Test Data	{ "submit_action": "True" }
Expected Behavior	System displays error: 'Beneficiary Name is mandatory.'

Test Data Summary:

{ "overall_input": "Blank beneficiary name, valid account number and bank code", "key_parameters": "beneficiary_name, account_number, bank_code" }

Validation Criteria	Mandatory fields enforced; Error message is clear and specific
Dependencies	Beneficiary form validation
Notes	Validates admin portal field validation.

Test Case 35: Security: Role-Based Access Control for Entitlement Configuration

Test ID	TC_035
Module	Corporate Module > Entitlement Management
Category	Security
Priority	Critical

Test Type	Negative
Risk Level	Critical
Estimated Time	5 minutes
Description	Checks that only users with proper roles can access and modify product entitlements.
Objective	Ensure RBAC is enforced for sensitive configuration actions.
Preconditions	User with 'Viewer' role is logged in.
Expected Result	User without proper role cannot access or modify entitlements.

Detailed Test Steps with Data:

Step Number	1
Description	Login as a user with 'Viewer' role.
Test Data	{ "username": "viewer01", "password": "ViewOnly@2024", "role": "Viewer" }
Expected Behavior	User is logged in with limited permissions.

Step Number	2
Description	Navigate to the entitlement configuration section.
Test Data	{ "navigation_path": "Admin > Entitlement Management" }
Expected Behavior	Access to entitlement configuration is denied.

Step Number	3
Description	Attempt to access entitlement configuration via direct URL.
Test Data	{ "direct_url": "https://corp-portal.example.com/admin/entitlements" }
Expected Behavior	System displays 'Access Denied' or redirects to dashboard.

Test Data Summary:

{ "overall_input": "Viewer role credentials, entitlement config URL", "key_parameters": "role, navigation_path, direct_url" }

Validation Criteria	Unauthorized access is blocked; No entitlement changes possible
Dependencies	RBAC implementation
Notes	Ensures sensitive settings are protected.

Test Case 36: Integration: Adding New Governmental Payment Type Post Go-Live

Test ID	TC_036
Module	Admin Portal > Product Management
Category	Integration
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	8 minutes
Description	Validates that admin can add a new payment type and it becomes available for customer entitlement
Objective	Ensure system supports future additions of payment types without code changes.
Preconditions	Admin user is logged in.; At least one customer exists.
Expected Result	New payment type is added and assignable to customers without code changes.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to 'Manage Payment Types' in the admin portal.
Test Data	{ "admin_username": "admin03", "portal_url": "https://corp-portal.example.com/admin/products" }
Expected Behavior	Payment types management screen is displayed.

Step Number	2
Description	Click 'Add New Payment Type' and enter details.
Test Data	{ "payment_type_name": "Municipal Fees", "description": "Payments for local municipal services", "category": "Governmental" }
Expected Behavior	New payment type is added and listed.

Step Number	3
Description	Navigate to customer entitlement configuration and verify new type is selectable.
Test Data	{ "customer_id": "CUST1002" }
Expected Behavior	New payment type 'Municipal Fees' appears in entitlement options.

Step Number	4
Description	Assign the new payment type to the customer and save changes.

Test Data	{ "customer_id": "CUST1002", "entitlements_to_add": "Municipal Fees" }
Expected Behavior	Customer profile reflects new entitlement.

Test Data Summary:

{ "overall_input": "New payment type details, customer selection", "key_parameters": "payment_type_name, category, customer_id" }

Validation Criteria	New type is added and visible; Can be assigned to customers
Dependencies	Admin portal product management
Notes	Validates future-proofing and modularity.

Test Case 37: Usability: Clear UI Feedback for Conditional Mandatory Fields

Test ID	TC_037
Module	Corporate Module > Payment Entry
Category	Usability
Priority	Medium
Test Type	Positive
Risk Level	Low
Estimated Time	5 minutes
Description	Checks that UI clearly indicates when a field becomes mandatory based on other selections.
Objective	Ensure users are guided when conditional fields are required.
Preconditions	User is logged into payment entry form.
Expected Result	UI provides clear feedback for conditional mandatory fields.

Detailed Test Steps with Data:

Step Number	1
Description	Select 'Custom Collection' as payment type.
Test Data	{ "payment_type": "Custom Collection" }
Expected Behavior	'Customs Code' field appears and is marked as mandatory.

Step Number	2
Description	Attempt to submit the form without entering 'Customs Code'.

Test Data	{ "customs_code": "" }
Expected Behavior	UI highlights 'Customs Code' and displays 'This field is required.'

Step Number	3
Description	Enter a valid customs code and re-submit.
Test Data	{ "customs_code": "EGCUST2024" }
Expected Behavior	Form submits successfully.

Test Data Summary:

{ "overall_input": "Custom Collection selected, customs_code blank then valid", "key_parameters": "payment_type, customs_code" }

Validation Criteria	Conditional fields are clearly indicated; Error messages are user-friendly
Dependencies	Conditional field logic
Notes	Improves user experience for complex forms.

Test Case 38: Error Handling: Invalid Characters in SWIFT-Compliant Field

Test ID	TC_038
Module	Corporate Module > Payment Entry
Category	Error Handling
Priority	Medium
Test Type	Negative
Risk Level	Medium
Estimated Time	4 minutes
Description	Validates that only allowed SWIFT characters are accepted in relevant fields.
Objective	Prevent invalid data entry in SWIFT-compliant fields.
Preconditions	User is logged into payment entry form.
Expected Result	System enforces SWIFT character compliance in relevant fields.

Detailed Test Steps with Data:

Step Number	1
Description	Enter an invalid character string in the 'Payment Reference' field.

Test Data	{ "payment_reference": "INV#2024$%^" }
Expected Behavior	System rejects input and displays error: 'Invalid characters detected.'

Step Number	2
Description	Enter a valid SWIFT-compliant string in the same field.
Test Data	{ "payment_reference": "INV-2024/EG" }
Expected Behavior	System accepts the input.

Test Data Summary:

{ "overall_input": "Invalid and valid payment reference strings", "key_parameters": "payment_reference" }

Validation Criteria	Only allowed characters accepted; Clear error for invalid input
Dependencies	SWIFT character validation
Notes	Prevents downstream data issues.

Test Case 39: Performance: Bulk Onboarding of 100 Customers

Test ID	TC_039
Module	Corporate Module > Customer Onboarding
Category	Performance
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	15 minutes
Description	Tests system performance and correctness when onboarding 100 customers via bulk upload.
Objective	Ensure system can handle bulk onboarding efficiently and accurately.
Preconditions	Admin user is logged in.; Bulk upload feature is enabled.
Expected Result	Bulk onboarding completes within 2 minutes and all customers are correctly created.

Detailed Test Steps with Data:

Step Number	1
Description	Prepare a CSV file with 100 unique customer records, each with valid mandatory fields.

Test Data	{
	"file_name": "bulk_onboarding_100.csv",
	"record_count": "100",
	"sample_record": "{\"customer_name\": \"TestCorp01\", \"GCIF_level\": \"Egypt\", \"company_registration\": \"EG98765
	}

Expected Behavior	CSV file is ready for upload.

Step Number	2
Description	Upload the CSV file using the bulk onboarding feature.

Test Data	{
	"upload_file": "bulk_onboarding_100.csv"
	}

Expected Behavior	System processes file and displays progress indicator.

Step Number	3
Description	Monitor processing time and check for errors.

Test Data	{
	"expected_max_time_sec": "120"
	}

Expected Behavior	All records processed within 2 minutes; errors (if any) are reported.

Step Number	4
Description	Verify that all 100 customers appear in the customer list.

Test Data	{
	"expected_count": "100"
	}

Expected Behavior	Customer list shows 100 new entries with correct details.

Test Data Summary:

{ "overall_input": "CSV file with 100 valid customer records", "key_parameters": "file_name, record_count, processing time" }

Validation Criteria	All records processed within SLA; No data loss or corruption
Dependencies	Bulk upload feature; Customer list view
Notes	Validates scalability for onboarding.

Test Case 40: Negative Test: Duplicate Product Entitlement Assignment

Test ID	TC_040
Module	Corporate Module > Entitlement Management
Category	Functional
Priority	Medium

Test Type	Negative
Risk Level	Low
Estimated Time	3 minutes
Description	Ensures system prevents duplicate assignment of the same product entitlement to a customer.
Objective	Prevent redundancy and data integrity issues in entitlement mapping.
Preconditions	Admin user is logged in.; Customer already has 'Tax Collection' entitlement.
Expected Result	System blocks duplicate entitlement assignment and shows clear error.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the entitlement management section for the customer.
Test Data	{ "customer_id": "CUST2001" }
Expected Behavior	Customer's current entitlements are displayed.

Step Number	2
Description	Attempt to add 'Tax Collection' entitlement again.
Test Data	{ "entitlement_to_add": "Tax Collection" }
Expected Behavior	System prevents duplicate assignment and displays error: 'Entitlement already assigned.'

Test Data Summary:

{ "overall_input": "Customer with existing entitlement, attempt duplicate add", "key_parameters": "customer_id, entitlement_to_add" }

Validation Criteria	Duplicate assignments are blocked; Error message is clear
Dependencies	Entitlement management logic
Notes	Prevents data redundancy.

Test Case 41: Verify Mandatory Field Validation During Customer Onboarding

Test ID	TC_041
Module	Corporate Module > Customer Onboarding
Category	Functional
Priority	Critical
Test Type	Negative
Risk Level	High

Estimated Time	8 minutes
Description	Ensure that all mandatory fields are enforced during the customer onboarding process, and appropri
Objective	Validate that mandatory fields cannot be bypassed and the system provides clear feedback.
Preconditions	User is logged in as a business user; Onboarding form is accessible
Expected Result	Mandatory fields are enforced and clear error messages are shown for missing data.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the customer onboarding form.
Test Data	{ "url": "https://corp-portal.example.com/onboarding", "browser": "Chrome" }
Expected Behavior	Onboarding form loads successfully.

Step Number	2
Description	Leave the 'Company Name' mandatory field empty and fill in all other fields with valid data.
Test Data	{ "company_name": "", "contact_email": "contact@acme.com", "country": "Egypt", "tax_id": "EG1234567890" }
Expected Behavior	System highlights 'Company Name' as required and prevents submission.

Step Number	3
Description	Attempt to submit the form.
Test Data	{ "action": "Submit" }
Expected Behavior	Error message 'Company Name is required' is displayed.

Step Number	4
Description	Fill in the 'Company Name' with 'Acme Corp' and submit the form.
Test Data	{ "company_name": "Acme Corp" }
Expected Behavior	Form is submitted successfully and onboarding proceeds to the next step.

Test Data Summary:

{ "overall_input": "Company Name, Contact Email, Country, Tax ID", "key_parameters": "company_name (mandatory)" }

Validation Criteria	Mandatory fields enforced; Error messages displayed

Dependencies	Onboarding form availability
Notes	Test covers only one mandatory field; repeat for others as needed.

Test Case 42: Product Entitlement Assignment Based on Customer Profile

Test ID	TC_042
Module	Corporate Module > Product Entitlement
Category	Functional
Priority	High
Test Type	Positive
Risk Level	Medium
Estimated Time	10 minutes
Description	Verify that the correct payment modules and sub-products are assigned to a customer based on thei
Objective	Ensure entitlement logic is correctly applied for Egypt-based customers.
Preconditions	Admin user logged in; Customer profile with GCIF level 'Egypt' exists
Expected Result	Entitlements are assigned as per country-specific rules and cannot be removed if defaulted.

Detailed Test Steps with Data:

Step Number	1
Description	Access the product entitlement configuration for customer GCIF 'EGY001'.
Test Data	{ "gcif": "EGY001" }
Expected Behavior	Product entitlement page for EGY001 is displayed.

Step Number	2
Description	Check default entitlements for 'Governmental Payments' and its sub-products.
Test Data	{ "product": "Governmental Payments" }
Expected Behavior	'Tax Collection', 'Custom Collection', and 'Universal Collection' are pre-selected.

Step Number	3
Description	Attempt to deselect 'Tax Collection' sub-product.
Test Data	{ "sub_product": "Tax Collection", "action": "Deselect" }
Expected Behavior	System prevents deselection and displays message: 'Default entitlement cannot be removed for Egypt.'

Step Number	4
Description	Add a new sub-product 'Adhoc Bill' under 'Governmental Payments'.
Test Data	{ "sub_product": "Adhoc Bill", "action": "Add" }
Expected Behavior	'Adhoc Bill' is successfully added to the entitlement list.

Test Data Summary:

{ "overall_input": "GCIF: EGY001, Product: Governmental Payments, Sub-products: Tax Collection, Adhoc Bill", "key_parameters": "country, GCIF level" }

Validation Criteria	Default entitlements enforced; Cannot remove mandatory sub-products
Dependencies	Customer profile exists
Notes	Focuses on Egypt-specific entitlement logic.

Test Case 43: Role-Based Access Control for Admin Portal

Test ID	TC_043
Module	Admin Portal > User Management
Category	Security
Priority	Critical
Test Type	Negative
Risk Level	Critical
Estimated Time	12 minutes
Description	Validate that only users with 'Admin' role can add new beneficiaries in the admin portal.
Objective	Ensure RBAC is enforced for beneficiary management.
Preconditions	User accounts exist with 'Admin' and 'Business User' roles
Expected Result	Only admin users can add beneficiaries; business users are restricted.

Detailed Test Steps with Data:

Step Number	1
Description	Login as a business user and navigate to the beneficiary management section.
Test Data	{ "username": "business_user01", "password": "Passw0rd!", "role": "Business User" }
Expected Behavior	Beneficiary management section is visible but 'Add Beneficiary' button is disabled.

Step Number	2

Description	Attempt to access the 'Add Beneficiary' page via direct URL.

Test Data	{
	  "url": "https://admin-portal.example.com/beneficiaries/add"
	}

Expected Behavior	Access is denied with 'Insufficient permissions' message.

Step Number	3
Description	Logout and login as an admin user.

Test Data	{
	  "username": "admin_user01",
	  "password": "Adm1nPass@2024",
	  "role": "Admin"
	}

Expected Behavior	Admin dashboard loads successfully.

Step Number	4
Description	Navigate to the beneficiary management section and click 'Add Beneficiary'.

Test Data	{
	  "action": "Add Beneficiary"
	}

Expected Behavior	'Add Beneficiary' form is displayed and accessible.

Step Number	5
Description	Fill in beneficiary details and submit.

Test Data	{
	  "beneficiary_name": "Global Supplies Ltd",
	  "account_number": "123456789012",
	  "bank_code": "EG500001",
	  "country": "Egypt"
	}

Expected Behavior	Beneficiary is added successfully and confirmation message is shown.

Test Data Summary:

{ "overall_input": "User roles, Beneficiary details", "key_parameters": "role-based permissions" }

Validation Criteria	RBAC enforced; No unauthorized access
Dependencies	User roles configured
Notes	Covers both UI and direct URL access attempts.

Test Case 44: Auto-Rejection of Unapproved Transactions After 45 Days

Test ID	TC_044
Module	Corporate Module > Transaction Workflow

Category	Integration
Priority	High
Test Type	Positive
Risk Level	Medium
Estimated Time	10 minutes
Description	Verify that transactions not approved or released within 45 days are automatically rejected by the sys
Objective	Ensure workflow automation for stale transactions.
Preconditions	Pending transaction exists older than 45 days
Expected Result	Stale transactions are auto-rejected and initiators are notified.

Detailed Test Steps with Data:

Step Number	1
Description	Identify a transaction with status 'Pending Approval' created 46 days ago.
Test Data	{ "transaction_id": "TXN20240501", "creation_date": "2024-05-01", "current_date": "2024-06-16" }
Expected Behavior	Transaction is listed as pending and older than 45 days.

Step Number	2
Description	Run the auto-rejection batch job manually.
Test Data	{ "batch_job": "AutoRejectPendingTransactions" }
Expected Behavior	Batch job executes successfully.

Step Number	3
Description	Check the status of transaction 'TXN20240501' after batch job execution.
Test Data	{ "transaction_id": "TXN20240501" }
Expected Behavior	Transaction status is updated to 'Rejected'.

Step Number	4
Description	Verify that a notification email is sent to the transaction initiator.
Test Data	{ "initiator_email": "initiator@acme.com" }
Expected Behavior	Initiator receives an email with subject 'Transaction Auto-Rejected'.

Test Data Summary:

{ "overall_input": "Transaction ID, Dates, Batch job, Initiator email", "key_parameters": "transaction age, batch job" }

Validation Criteria	Transactions auto-rejected; Notification sent
Dependencies	Batch job scheduler
Notes	Uses a manual trigger for batch job to expedite test.

Test Case 45: Boundary Test for Amount Field (AMT) in Payment Module

Test ID	TC_045
Module	Corporate Module > Payment Processing
Category	Boundary
Priority	High
Test Type	Boundary
Risk Level	Medium
Estimated Time	9 minutes
Description	Test the lower and upper boundaries for the amount field (AMT) with two decimal places.
Objective	Ensure amount field enforces min/max limits and decimal precision.
Preconditions	User is logged in and has access to payment module
Expected Result	Amount field enforces boundaries and decimal precision.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the payment initiation form.
Test Data	{ "url": "https://corp-portal.example.com/payments/initiate" }
Expected Behavior	Payment form loads successfully.

Step Number	2
Description	Enter minimum allowed amount (0.01) and submit.
Test Data	{ "amount": "0.01" }
Expected Behavior	Form accepts input and proceeds to confirmation.

Step Number	3
Description	Enter maximum allowed amount (99999999.99) and submit.

Test Data	{ "amount": "99999999.99" }
Expected Behavior	Form accepts input and proceeds to confirmation.

Step Number	4
Description	Enter an amount with more than two decimal places (100.123) and submit.
Test Data	{ "amount": "100.123" }
Expected Behavior	System displays error: 'Amount must have at most two decimal places.'

Step Number	5
Description	Enter an amount below minimum (0.00) and submit.
Test Data	{ "amount": "0.0" }
Expected Behavior	System displays error: 'Amount must be greater than 0.00.'

Test Data Summary:

{ "overall_input": "Amounts: 0.01, 99999999.99, 100.123, 0.00", "key_parameters": "AMT field limits" }

Validation Criteria	Min/max enforced; Decimal precision enforced
Dependencies	Payment module configuration
Notes	Covers both lower and upper boundary conditions.

Test Case 46: Error Handling for Invalid SWIFT Characters in Alphanumeric Fields

Test ID	TC_046
Module	Corporate Module > Data Entry
Category	Error Handling
Priority	Medium
Test Type	Negative
Risk Level	Medium
Estimated Time	7 minutes
Description	Verify that only SWIFT-compliant characters are accepted in alphanumeric fields.
Objective	Ensure input validation for SWIFT compliance.
Preconditions	User is logged in and can access data entry forms
Expected Result	System enforces SWIFT character compliance in alphanumeric fields.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the beneficiary addition form.
Test Data	{ "url": "https://corp-portal.example.com/beneficiaries/add" }
Expected Behavior	Beneficiary form loads successfully.

Step Number	2
Description	Enter 'Beneficiary Name' with invalid character (e.g., 'Acme Corp!@#').
Test Data	{ "beneficiary_name": "Acme Corp!@#" }
Expected Behavior	System displays error: 'Invalid characters detected. Only SWIFT-compliant characters allowed.'

Step Number	3
Description	Enter 'Beneficiary Name' with valid SWIFT-compliant characters (e.g., 'Acme-Corp_2024').
Test Data	{ "beneficiary_name": "Acme-Corp_2024" }
Expected Behavior	Input is accepted and no error is displayed.

Step Number	4
Description	Submit the form with valid data.
Test Data	{ "beneficiary_name": "Acme-Corp_2024", "account_number": "9876543210", "bank_code": "EG500002" }
Expected Behavior	Beneficiary is added successfully.

Test Data Summary:

{ "overall_input": "Beneficiary Name: Acme Corp!@#, Acme-Corp_2024", "key_parameters": "SWIFT character set" }

Validation Criteria	Invalid characters rejected; Valid characters accepted
Dependencies	SWIFT validation rules implemented
Notes	Focuses on one field; similar logic applies to other alphanumeric fields.

Test Case 47: Performance Test for Bulk Beneficiary Upload

Test ID	TC_047
Module	Admin Portal > Beneficiary Management

Category	Performance
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	15 minutes
Description	Assess system performance when uploading a large beneficiary file (1000 records) via the admin po
Objective	Ensure system handles bulk uploads within acceptable time and without errors.
Preconditions	Admin user logged in; Bulk upload feature enabled
Expected Result	Bulk upload completes within 60 seconds and all records are processed successfully.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the bulk beneficiary upload page.
Test Data	{ "url": "https://admin-portal.example.com/beneficiaries/bulk-upload" }
Expected Behavior	Bulk upload page loads successfully.

Step Number	2
Description	Select and upload a CSV file containing 1000 valid beneficiary records.
Test Data	{ "file_path": "/testdata/beneficiaries_1000.csv", "record_count": "1000" }
Expected Behavior	File is accepted and upload process begins.

Step Number	3
Description	Monitor upload progress and measure completion time.
Test Data	{ "timer_start": "Upload initiated" }
Expected Behavior	Upload completes within 60 seconds.

Step Number	4
Description	Verify that all 1000 records are processed without errors.
Test Data	{ "expected_records": "1000" }
Expected Behavior	System confirms successful upload of 1000 beneficiaries.

Test Data Summary:

{ "overall_input": "CSV file with 1000 records", "key_parameters": "file size, record count" }

Validation Criteria	Upload time < 60s; No errors in processing
Dependencies	Bulk upload functionality
Notes	Use realistic beneficiary data in the CSV file.

Test Case 48: Usability Test for Adding New Payment Type Post Go-Live

Test ID	TC_048
Module	Admin Portal > Payment Type Management
Category	Usability
Priority	Medium
Test Type	Positive
Risk Level	Low
Estimated Time	8 minutes
Description	Evaluate the ease of adding a new governmental payment type through the admin UI after system g
Objective	Ensure the admin UI supports intuitive addition of new payment types.
Preconditions	Admin user logged in; System is in post go-live state
Expected Result	Admin can add new payment types easily and they become available for configuration.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the 'Payment Types' management section in the admin portal.

Test Data	{
	  "url": "https://admin-portal.example.com/payment-types"
	}

Expected Behavior	'Payment Types' management page loads successfully.

Step Number	2
Description	Click on 'Add New Payment Type'.

Test Data	{
	  "action": "Add New"
	}

Expected Behavior	Form for adding new payment type is displayed.

Step Number	3
Description	Enter payment type details: Name='Municipal Fees', Code='MUNFEE', Category='Governmental Payments'.

Test Data	{
	"name": "Municipal Fees",
	"code": "MUNFEE",
	"category": "Governmental Payments"
	}

Expected Behavior	System accepts input and enables the 'Save' button.

Step Number	4
Description	Save the new payment type.

Test Data	{
	"action": "Save"
	}

Expected Behavior	New payment type 'Municipal Fees' is added and visible in the payment types list.

Step Number	5
Description	Verify that the new payment type is available for entitlement assignment.

Test Data	{
	"payment_type": "Municipal Fees"
	}

Expected Behavior	'Municipal Fees' appears as an option in entitlement configuration.

Test Data Summary:

{ "overall_input": "Payment type details: Municipal Fees, MUNFEE, Governmental Payments", "key_parameters": "name, code, category" }

Validation Criteria	New type visible in list; Available for entitlement
Dependencies	Admin UI for payment types
Notes	Test focuses on UI intuitiveness and discoverability.

Test Case 49: Integration Test for Entitlement Assignment and Transaction Processing

Test ID	TC_049
Module	Corporate Module > Entitlement & Transactions
Category	Integration
Priority	Critical
Test Type	Positive
Risk Level	High
Estimated Time	14 minutes
Description	Verify that a user with assigned entitlements can initiate and complete a transaction for an entitled pr
Objective	Ensure integration between entitlement assignment and transaction processing.
Preconditions	User is assigned entitlement for 'Tax Collection'

Expected Result	User can initiate and complete a transaction for an entitled product.

Detailed Test Steps with Data:

Step Number	1
Description	Login as user 'entitled_user01' with entitlement for 'Tax Collection'.
Test Data	{ "username": "entitled_user01", "password": "UserPass@2024", "entitlement": "Tax Collection" }
Expected Behavior	User dashboard displays 'Tax Collection' as available product.

Step Number	2
Description	Initiate a new 'Tax Collection' transaction.
Test Data	{ "product": "Tax Collection" }
Expected Behavior	Transaction initiation form for 'Tax Collection' is displayed.

Step Number	3
Description	Enter transaction details: Amount=5000.00, Tax ID='EGTAX2024', Description='Quarterly tax payment'.
Test Data	{ "amount": "5000.0", "tax_id": "EGTAX2024", "description": "Quarterly tax payment" }
Expected Behavior	Form accepts input and enables 'Submit' button.

Step Number	4
Description	Submit the transaction.
Test Data	{ "action": "Submit" }
Expected Behavior	Transaction is created and status is set to 'Pending Approval'.

Step Number	5
Description	Logout and login as an approver for the same customer.
Test Data	{ "username": "approver01", "password": "Approve@2024" }
Expected Behavior	Approver dashboard loads successfully.

Step Number	6
Description	Approve the pending 'Tax Collection' transaction.

Test Data	{ "transaction_id": "Auto-generated", "action": "Approve" }
Expected Behavior	Transaction status changes to 'Approved'.

Test Data Summary:

{ "overall_input": "User credentials, Entitlement, Transaction details", "key_parameters": "entitlement, product, transaction" }

Validation Criteria	Entitled product visible; Transaction processed end-to-end
Dependencies	Entitlement assignment; Approver user setup
Notes	Covers both entitlement and transaction workflow.

Test Case 50: Security Test: Attempt to Add Beneficiary with SQL Injection

Test ID	TC_050
Module	Admin Portal > Beneficiary Management
Category	Security
Priority	Critical
Test Type	Negative
Risk Level	Critical
Estimated Time	10 minutes
Description	Test system security by attempting to add a beneficiary with SQL injection payload in the name field.
Objective	Ensure the system is protected against SQL injection attacks.
Preconditions	Admin user logged in
Expected Result	SQL injection payload is rejected and system remains secure.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to 'Add Beneficiary' form.
Test Data	{ "url": "https://admin-portal.example.com/beneficiaries/add" }
Expected Behavior	Add Beneficiary form loads successfully.

Step Number	2
Description	Enter beneficiary name with SQL injection payload: 'Robert'); DROP TABLE Beneficiaries;--'.

Test Data	{ "beneficiary_name": "Robert'); DROP TABLE Beneficiaries;--", "account_number": "1122334455", "bank_code": "EG500003" }
Expected Behavior	System rejects input and displays error: 'Invalid characters detected.'

Step Number	3
Description	Check system logs for any SQL errors or exceptions.
Test Data	{ "log_type": "Application/Error" }
Expected Behavior	No SQL errors or stack traces are present in logs.

Step Number	4
Description	Attempt to submit the form with sanitized input: 'Robert Smith'.
Test Data	{ "beneficiary_name": "Robert Smith" }
Expected Behavior	Beneficiary is added successfully.

Test Data Summary:

{ "overall_input": "Beneficiary Name with SQL injection, Account Number, Bank Code", "key_parameters": "input sanitization" }

Validation Criteria	SQL injection blocked; No system errors
Dependencies	Input validation implementation
Notes	Test should be repeated for other input fields.

Test Case 51: Cross-Platform Test: Entitlement Assignment on Different Browsers

Test ID	TC_051
Module	Admin Portal > Product Entitlement
Category	Integration
Priority	Medium
Test Type	Positive
Risk Level	Low
Estimated Time	13 minutes
Description	Verify that product entitlement assignment works consistently across Chrome, Firefox, and Edge.
Objective	Ensure cross-platform compatibility for entitlement configuration.
Preconditions	Admin user credentials available

Expected Result	Entitlement assignment works consistently across Chrome, Firefox, and Edge.

Detailed Test Steps with Data:

Step Number	1
Description	Login to the admin portal using Chrome.
Test Data	{ "browser": "Chrome", "username": "admin_user01", "password": "Adm1nPass@2024" }
Expected Behavior	Admin dashboard loads without UI issues.

Step Number	2
Description	Assign 'Universal Collection' entitlement to customer GCIF 'EGY002'.
Test Data	{ "gcif": "EGY002", "entitlement": "Universal Collection" }
Expected Behavior	Entitlement assignment is successful and confirmation is displayed.

Step Number	3
Description	Repeat entitlement assignment on Firefox.
Test Data	{ "browser": "Firefox", "gcif": "EGY002", "entitlement": "Universal Collection" }
Expected Behavior	Process completes successfully with no browser-specific issues.

Step Number	4
Description	Repeat entitlement assignment on Edge.
Test Data	{ "browser": "Edge", "gcif": "EGY002", "entitlement": "Universal Collection" }
Expected Behavior	Process completes successfully with no browser-specific issues.

Step Number	5
Description	Verify that the entitlement is reflected correctly in all browsers.
Test Data	{ "gcif": "EGY002" }
Expected Behavior	Entitlement status is consistent across all browsers.

Test Data Summary:

{ "overall_input": "Browsers: Chrome, Firefox, Edge; GCIF: EGY002; Entitlement: Universal Collection", "key_parameters": "browser compatibility" }

Validation Criteria	No browser-specific issues; Consistent entitlement status
Dependencies	Admin portal accessible on all browsers
Notes	Focuses on UI and functional consistency.

Test Case 52: Successful Admin Login with Valid Credentials

Test ID	TC_051
Module	Authentication > Login
Category	Functional
Priority	Critical
Test Type	Positive
Risk Level	Critical
Estimated Time	3 minutes
Description	Verify that an admin user can successfully log in using valid credentials and is redirected to the dash
Objective	Ensure authentication works for admin users with correct credentials.
Preconditions	Admin user account exists with username 'admin_corp' and password 'AdminPass!2024'; User is log
Expected Result	Admin user is logged in and lands on the dashboard.

Detailed Test Steps with Data:

Step Number	1
Description	Open Chrome browser and navigate to the login page.
Test Data	{ "browser": "Chrome", "url": "https://corpportal.example.com/login" }
Expected Behavior	Login page loads successfully with username and password fields visible.

Step Number	2
Description	Enter valid admin username and password.
Test Data	{ "username": "admin_corp", "password": "AdminPass!2024" }
Expected Behavior	Fields accept input and no validation errors are shown.

Step Number	3
Description	Click the 'Login' button.

Test Data	{ "button": "Login" }
Expected Behavior	System authenticates credentials and displays a loading spinner.

Step Number	4
Description	Wait for redirection after successful authentication.
Test Data	{ "timeout_seconds": "5" }
Expected Behavior	User is redirected to the admin dashboard with a welcome message.

Test Data Summary:

{ "overall_input": "admin_corp/AdminPass!2024", "key_parameters": "username, password, browser" }

Validation Criteria	Successful authentication; Dashboard access
Dependencies	User account exists; Login service operational
Notes	Test with actual admin credentials only.

Test Case 53: Form Validation for Mandatory Numeric and Amount Fields

Test ID	TC_052
Module	Corporate Module > Customer Onboarding
Category	Functional
Priority	High
Test Type	Negative
Risk Level	High
Estimated Time	4 minutes
Description	Verify that the onboarding form enforces mandatory numeric and amount fields with correct validatio
Objective	Ensure form validation for mandatory fields is enforced.
Preconditions	User is logged in as admin; Onboarding form is accessible
Expected Result	Form is not submitted and mandatory field errors are displayed.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the onboarding form.

Test Data	{ "menu": "Customer Onboarding" }
Expected Behavior	Onboarding form loads with all fields visible.

Step Number	2
Description	Leave 'Customer ID' (NUM, mandatory) and 'Initial Deposit' (AMT, mandatory) fields empty.
Test Data	{ "Customer ID": "", "Initial Deposit": "" }
Expected Behavior	Fields remain empty.

Step Number	3
Description	Fill all other required fields with valid data.
Test Data	{ "Company Name": "Acme Corp", "Country": "Egypt", "Contact Email": "contact@acme.com" }
Expected Behavior	Other fields accept input.

Step Number	4
Description	Click 'Submit' to attempt onboarding.
Test Data	{ "button": "Submit" }
Expected Behavior	Form displays validation errors for missing 'Customer ID' and 'Initial Deposit'.

Test Data Summary:

{ "overall_input": "Missing Customer ID and Initial Deposit, valid other fields", "key_parameters": "Customer ID, Initial Deposit" }

Validation Criteria	Mandatory field validation; Error messages displayed
Dependencies	Onboarding form available
Notes	Test for both numeric and amount field types.

Test Case 54: Boundary Test for Fixed Length Alphanumeric Field

Test ID	TC_053
Module	Corporate Module > Product Entitlement
Category	Boundary
Priority	Medium
Test Type	Boundary

Risk Level	Medium
Estimated Time	3 minutes
Description	Verify that the system accepts input up to the maximum allowed length for a fixed length alphanumer
Objective	Ensure boundary validation for fixed length fields.
Preconditions	User is logged in as admin; Product entitlement configuration form is accessible
Expected Result	System enforces fixed length and accepts only up to 10 characters.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the product entitlement configuration form.
Test Data	{ "menu": "Product Entitlement" }
Expected Behavior	Form loads with all entitlement fields visible.

Step Number	2
Description	Enter exactly 10 alphanumeric characters in the 'Entitlement Code' field (fixed length 10).
Test Data	{ "Entitlement Code": "AB12CD34EF" }
Expected Behavior	Field accepts the 10 characters without error.

Step Number	3
Description	Attempt to enter an 11th character in the 'Entitlement Code' field.
Test Data	{ "Entitlement Code": "AB12CD34EFA" }
Expected Behavior	System prevents entry of the 11th character or displays an error.

Step Number	4
Description	Submit the form with the valid 10-character code.
Test Data	{ "button": "Submit" }
Expected Behavior	Form is submitted successfully.

Test Data Summary:

{ "overall_input": "AB12CD34EF (10 chars), AB12CD34EFA (11 chars)", "key_parameters": "Entitlement Code" }

Validation Criteria	Fixed length enforcement; Error on overflow
Dependencies	Entitlement form available
Notes	Test both acceptance and rejection at boundary.

Test Case 55: Role-Based Access Control for Product Entitlement

Test ID	TC_054
Module	Corporate Module > Product Entitlement
Category	Security
Priority	Critical
Test Type	Negative
Risk Level	Critical
Estimated Time	3 minutes
Description	Verify that only admin users can access and modify product entitlement configurations.
Objective	Ensure RBAC is enforced for sensitive configuration screens.
Preconditions	User account 'user_basic' exists with no admin privileges; User is logged out
Expected Result	Non-admin users cannot access or modify product entitlements.

Detailed Test Steps with Data:

Step Number	1
Description	Login as non-admin user.
Test Data	{ "username": "user_basic", "password": "UserPass!2024" }
Expected Behavior	User is authenticated and redirected to user dashboard.

Step Number	2
Description	Attempt to navigate to the 'Product Entitlement' configuration page via URL.
Test Data	{ "url": "https://corpportal.example.com/admin/entitlement" }
Expected Behavior	Access is denied and user receives a '403 Forbidden' or equivalent error.

Step Number	3
Description	Attempt to access the configuration page via the UI menu.
Test Data	{ "menu": "Product Entitlement" }
Expected Behavior	Menu option is not visible or is disabled.

Test Data Summary:

{ "overall_input": "user_basic/UserPass!2024", "key_parameters": "user role" }

Validation Criteria	Access denied for non-admins; No data leakage
Dependencies	RBAC implemented
Notes	Test both direct URL and UI navigation.

Test Case 56: Auto-Rejection of Stale Transactions after 45 Days

Test ID	TC_055
Module	Corporate Module > Transaction Workflow
Category	Functional
Priority	High
Test Type	Positive
Risk Level	High
Estimated Time	5 minutes
Description	Ensure that transactions pending approval or release for more than 45 days are automatically rejecte
Objective	Validate automated workflow handling for stale transactions.
Preconditions	Transaction exists with status 'Pending Approval' and creation date 46 days ago
Expected Result	Transactions older than 45 days are auto-rejected.

Detailed Test Steps with Data:

Step Number	1
Description	Login as admin user.

Test Data	{
	 "username": "admin_corp",
	 "password": "AdminPass!2024"
	 }

Expected Behavior	Admin dashboard loads.

Step Number	2
Description	Navigate to the 'Pending Transactions' list.

Test Data	{
	 "menu": "Pending Transactions"
	 }

Expected Behavior	List of pending transactions is displayed.

Step Number	3
Description	Search for transaction with ID 'TXN123456' and creation date 46 days ago.

Test Data	{
	 "transaction_id": "TXN123456",
	 "creation_date": "2024-05-15"
	 }

Expected Behavior	Transaction is listed with status 'Pending Approval'.

Step Number	4
Description	Trigger the auto-rejection batch job or wait for scheduled execution.

Test Data	{
	  "batch_job": "AutoRejectStaleTransactions"
	}

Expected Behavior	Transaction status changes to 'Rejected' automatically.

Test Data Summary:

{ "overall_input": "TXN123456, creation date 2024-05-15", "key_parameters": "transaction age" }

Validation Criteria	Auto-rejection after 45 days; Status update
Dependencies	Batch job scheduled; Transaction data seeded
Notes	Test with transactions just over and just under 45 days.

Test Case 57: Add New Governmental Payment Type via Admin Portal

Test ID	TC_056
Module	Admin Portal > Governmental Payments Management
Category	Integration
Priority	Medium
Test Type	Positive
Risk Level	Medium
Estimated Time	6 minutes
Description	Verify that admin can add a new governmental payment type and it becomes available for customer
Objective	Ensure admin portal supports dynamic addition of payment types.
Preconditions	User is logged in as admin; No payment type named 'Municipal Fees' exists
Expected Result	New payment type is added and available for entitlement.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to 'Governmental Payments Management' in the admin portal.

Test Data	{
	  "menu": "Governmental Payments Management"
	}

Expected Behavior	Management page loads with list of payment types.

Step Number	2

Description	Click 'Add New Payment Type'.
Test Data	{ "button": "Add New Payment Type" }
Expected Behavior	Form to add new payment type appears.

Step Number	3
Description	Enter details for the new payment type.
Test Data	{ "Payment Type Name": "Municipal Fees", "Description": "Payments for municipal services", "Default Country": "Egypt" }
Expected Behavior	Fields accept input with no errors.

Step Number	4
Description	Submit the new payment type.
Test Data	{ "button": "Save" }
Expected Behavior	Payment type is saved and appears in the list.

Step Number	5
Description	Verify that 'Municipal Fees' is available in the customer entitlement configuration.
Test Data	{ "entitlement_menu": "Product Entitlement" }
Expected Behavior	'Municipal Fees' is selectable as a payment type.

Test Data Summary:

{ "overall_input": "Municipal Fees, Egypt", "key_parameters": "Payment Type Name" }

Validation Criteria	Payment type addition; Availability for entitlement
Dependencies	Admin portal operational
Notes	Test with unique payment type names.

Test Case 58: SQL Injection Attempt on Beneficiary Addition

Test ID	TC_057
Module	Admin Portal > Beneficiary Management
Category	Security
Priority	Critical
Test Type	Negative

Risk Level	Critical
Estimated Time	4 minutes
Description	Verify that the system is protected against SQL injection attacks when adding a new beneficiary.
Objective	Ensure input sanitization and security for beneficiary management.
Preconditions	User is logged in as admin; Beneficiary addition form is accessible
Expected Result	System prevents SQL injection and no malicious action is performed.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to 'Beneficiary Management' in the admin portal.

Test Data	{
	 "menu": "Beneficiary Management"
	 }

Expected Behavior	Beneficiary management page loads.

Step Number	2
Description	Click 'Add New Beneficiary'.

Test Data	{
	 "button": "Add New Beneficiary"
	 }

Expected Behavior	Beneficiary addition form appears.

Step Number	3
Description	Enter SQL injection string in the 'Beneficiary Name' field.

Test Data	{
	 "Beneficiary Name": "'; DROP TABLE beneficiaries; --",
	 "Account Number": "1234567890",
	 "Bank": "Cairo Bank"
	 }

Expected Behavior	System rejects input or sanitizes it; error message is displayed.

Step Number	4
Description	Submit the form.

Test Data	{
	 "button": "Save"
	 }

Expected Behavior	Beneficiary is not added and system logs the attempt.

Test Data Summary:

{ "overall_input": "'; DROP TABLE beneficiaries; --", "key_parameters": "Beneficiary Name" }

Validation Criteria	No SQL injection possible; Proper error handling
Dependencies	Input sanitization implemented

Notes	Test with various SQL payloads.

Test Case 59: Performance Test: Bulk Entitlement Assignment

Test ID	TC_058
Module	Corporate Module > Product Entitlement
Category	Performance
Priority	Medium
Test Type	Positive
Risk Level	Medium
Estimated Time	8 minutes
Description	Measure system performance when assigning product entitlements to a large number of customers s
Objective	Ensure system can handle bulk entitlement operations efficiently.
Preconditions	User is logged in as admin; CSV file with 1000 customer records is prepared
Expected Result	All 1000 customers are assigned entitlements within 60 seconds.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to 'Bulk Entitlement Assignment' in the admin portal.
Test Data	{ "menu": "Bulk Entitlement Assignment" }
Expected Behavior	Bulk assignment page loads.

Step Number	2
Description	Upload CSV file with 1000 customer records.
Test Data	{ "file_name": "bulk_entitlement_1000.csv", "file_size": "350KB" }
Expected Behavior	File is uploaded and parsed successfully.

Step Number	3
Description	Select 'Universal Collection' as the product entitlement.
Test Data	{ "product": "Universal Collection" }
Expected Behavior	'Universal Collection' is selected for all customers.

Step Number	4
Description	Click 'Assign Entitlements' and measure processing time.

Test Data	{ "button": "Assign Entitlements" }
Expected Behavior	System processes all records within 60 seconds and displays success message.

Test Data Summary:

{ "overall_input": "bulk_entitlement_1000.csv, Universal Collection", "key_parameters": "file size, record count" }

Validation Criteria	Performance within SLA; No errors for valid data
Dependencies	Bulk assignment feature enabled
Notes	Monitor CPU and memory usage during test.

Test Case 60: Usability: Field Labels and Help Text for Onboarding Form

Test ID	TC_059
Module	Corporate Module > Customer Onboarding
Category	Usability
Priority	Low
Test Type	Positive
Risk Level	Low
Estimated Time	4 minutes
Description	Ensure that all fields on the onboarding form have clear labels and context-sensitive help text.
Objective	Validate user experience for form completion.
Preconditions	User is logged in as admin; Onboarding form is accessible
Expected Result	All fields have clear labels and help text; error messages are user-friendly.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the onboarding form.
Test Data	{ "menu": "Customer Onboarding" }
Expected Behavior	Form loads with all fields visible.

Step Number	2
Description	Hover over the 'Initial Deposit' field label.

Test Data	{ "field": "Initial Deposit" }
Expected Behavior	Tooltip or help text appears explaining the required format (e.g., 'Enter amount in EGP, two decimal places').

Step Number	3
Description	Check that all mandatory fields are marked with a red asterisk.
Test Data	{ "fields": "Customer ID, Initial Deposit, Company Name" }
Expected Behavior	Mandatory fields are visually indicated.

Step Number	4
Description	Attempt to submit the form with a missing mandatory field and observe the error message.
Test Data	{ "Customer ID": "", "Initial Deposit": "1000.00", "Company Name": "Acme Corp" }
Expected Behavior	Clear, user-friendly error message is displayed for the missing field.

Test Data Summary:

{ "overall_input": "Hover and submit with missing Customer ID", "key_parameters": "field labels, help text" }

Validation Criteria	Label clarity; Help text presence
Dependencies	Help text implemented
Notes	Check for accessibility compliance.

Test Case 61: Error Handling: Invalid Amount Field Format

Test ID	TC_060
Module	Corporate Module > Customer Onboarding
Category	Error Handling
Priority	Medium
Test Type	Negative
Risk Level	Medium
Estimated Time	3 minutes
Description	Verify that the system handles invalid amount field formats gracefully and displays appropriate error
Objective	Ensure robust error handling for amount fields.
Preconditions	User is logged in as admin; Onboarding form is accessible
Expected Result	Form is not submitted and clear error message is shown for invalid amount format.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the onboarding form.
Test Data	{ "menu": "Customer Onboarding" }
Expected Behavior	Form loads with all fields visible.

Step Number	2
Description	Enter invalid value in the 'Initial Deposit' field (e.g., '1000,00' using comma).
Test Data	{ "Initial Deposit": "1000,00" }
Expected Behavior	Field accepts input but highlights it as invalid.

Step Number	3
Description	Fill all other required fields with valid data.
Test Data	{ "Customer ID": "20001", "Company Name": "Beta Corp", "Country": "Egypt" }
Expected Behavior	Other fields accept input.

Step Number	4
Description	Click 'Submit' to attempt onboarding.
Test Data	{ "button": "Submit" }
Expected Behavior	Form displays error message: 'Amount must be in format 0.00'.

Test Data Summary:

{ "overall_input": "Initial Deposit: 1000,00", "key_parameters": "amount field format" }

Validation Criteria	Error message for invalid format; No data saved
Dependencies	Amount field validation implemented
Notes	Test with various invalid formats.

Test Case 62: Verify Product Entitlement Assignment During Customer Onboarding (Egypt GCIF)

Test ID	TC_061
Module	Corporate Module > Customer Onboarding
Category	Functional
Priority	Critical
Test Type	Positive
Risk Level	Critical
Estimated Time	10 minutes
Description	Test that a new customer with GCIF level for Egypt is automatically assigned the correct default prod
Objective	Ensure entitlement logic for Egypt is correctly applied and defaults are set as per documentation.
Preconditions	Admin user is logged into the onboarding portal; Product entitlement configuration for Egypt is availal
Expected Result	Customer with Egypt GCIF is onboarded with correct default product entitlements, and mandatory su

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the customer onboarding page as an admin
Test Data	{ "admin_username": "admin_egypt", "admin_password": "AdminPass!2024", "url": "https://corp-portal.example.com/onboarding" }
Expected Behavior	Onboarding page loads successfully with admin controls visible

Step Number	2
Description	Enter new customer details with GCIF level set to Egypt
Test Data	{ "customer_name": "Cairo Holdings LLC", "gcif": "EGY1234567", "country": "Egypt", "contact_email": "contact@cairoholdings.com" }
Expected Behavior	Customer details are accepted and country-specific fields appear

Step Number	3
Description	Proceed to product entitlement section and review default selections
Test Data	{ "entitlement_section": "True" }
Expected Behavior	Default products (Governmental Payments, Tax Collection, Custom Collection, Universal Collection) and their sub-prod

Step Number	4
Description	Attempt to deselect a mandatory sub-product (e.g., Adhoc Bill under Tax Collection)

Test Data	{ "sub_product": "Adhoc Bill", "action": "deselect" }
Expected Behavior	System prevents deselection and displays a message: 'Adhoc Bill is mandatory for Egypt GCIF customers'

Step Number	5
Description	Submit the onboarding form
Test Data	{ "confirmation": "True" }
Expected Behavior	Customer is onboarded successfully and entitlement mapping is stored

Test Data Summary:

{ "overall_input": "New customer with Egypt GCIF; default product/sub-product selections", "key_parameters": "gcif: EGY1234567, country: Egypt" }

Validation Criteria	Mandatory products/sub-products are pre-selected and enforced; Entitlement mapping is correctly stored
Dependencies	Product entitlement configuration for Egypt; Admin portal access
Notes	Focuses on Egypt-specific entitlement logic and default enforcement.

Test Case 63: File Upload Validation for Beneficiary Addition (Admin Portal)

Test ID	TC_062
Module	Admin Portal > Beneficiary Management
Category	Functional
Priority	High
Test Type	Boundary/Error
Risk Level	High
Estimated Time	12 minutes
Description	Test the file upload functionality for adding beneficiaries, including file type and size validation.
Objective	Ensure only allowed file types and sizes are accepted, and invalid files are rejected with proper mess
Preconditions	Admin user is logged into the admin portal; Beneficiary upload feature is enabled
Expected Result	Only allowed file types and sizes are accepted; invalid files are rejected with clear error messages.

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the 'Add Beneficiary' section and select 'Upload File'

Test Data	{ "admin_username": "admin_beneficiary", "admin_password": "Beneficiary#2024", "url": "https://admin.example.com/beneficiaries/upload" }
Expected Behavior	'Upload File' dialog appears

Step Number	2
Description	Upload a valid CSV file with 10 beneficiary records
Test Data	{ "file_name": "beneficiaries_valid.csv", "file_type": "text/csv", "file_size_kb": "45", "record_count": "10" }
Expected Behavior	File is accepted, records are previewed, and no errors are shown

Step Number	3
Description	Attempt to upload an unsupported file type (e.g., .exe file)
Test Data	{ "file_name": "malware.exe", "file_type": "application/x-msdownload", "file_size_kb": "120" }
Expected Behavior	System rejects the file and displays: 'Unsupported file type. Only CSV and XLSX are allowed.'

Step Number	4
Description	Upload a valid XLSX file exceeding the maximum allowed size (e.g., 6 MB when limit is 5 MB)
Test Data	{ "file_name": "large_beneficiaries.xlsx", "file_type": "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", "file_size_kb": "6144" }
Expected Behavior	System rejects the file and displays: 'File size exceeds the 5 MB limit.'

Step Number	5
Description	Upload a valid XLSX file with 250 beneficiary records (boundary test)
Test Data	{ "file_name": "beneficiaries_250.xlsx", "file_type": "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", "file_size_kb": "4900", "record_count": "250" }
Expected Behavior	File is accepted, records are previewed, and no errors are shown

Test Data Summary:

{ "overall_input": "CSV/XLSX files of various sizes and types", "key_parameters": "file_type, file_size_kb, record_count" }

Validation Criteria	File type and size validation enforced; Clear error messages for invalid uploads
Dependencies	File upload component; Beneficiary management module
Notes	Includes both positive and negative scenarios, including boundary test for record count.

Test Case 64: Security Test: SWIFT Compliance Character Validation in Amount Field

Test ID	TC_063
Module	Corporate Module > Transaction Processing
Category	Security
Priority	High
Test Type	Negative/Boundary
Risk Level	High
Estimated Time	8 minutes
Description	Test that the amount field only accepts SWIFT-compliant characters and rejects invalid input.
Objective	Ensure that only valid characters are accepted in amount fields as per SWIFT compliance.
Preconditions	User is logged into the transaction entry page; Amount field is visible and editable
Expected Result	Amount field only accepts SWIFT-compliant characters (digits and decimal point), and all invalid cha

Detailed Test Steps with Data:

Step Number	1
Description	Navigate to the transaction entry form
Test Data	{ "username": "user_swift", "password": "SwiftTest@2024", "url": "https://corp-portal.example.com/transactions/new" }
Expected Behavior	Transaction entry form loads successfully

Step Number	2
Description	Enter a valid amount using digits and a decimal point (e.g., 12345.67)
Test Data	{ "amount": "12345.67" }
Expected Behavior	Amount is accepted without error

Step Number	3
Description	Attempt to enter an amount with an invalid character (e.g., 1234@56)

Test Data	{ "amount": "1234@56" }
Expected Behavior	System rejects input and displays: 'Invalid character detected. Only digits and decimal points are allowed.'

Step Number	4
Description	Attempt to enter an amount with a comma (e.g., 1,234.56)
Test Data	{ "amount": "1,234.56" }
Expected Behavior	System rejects input and displays: 'Commas are not allowed in amount fields.'

Step Number	5
Description	Enter an amount with two decimal places (boundary test, e.g., 100.99)
Test Data	{ "amount": "100.99" }
Expected Behavior	Amount is accepted without error

Test Data Summary:

{ "overall_input": "Amounts with valid and invalid characters", "key_parameters": "amount field input" }

Validation Criteria	Only allowed characters accepted; Clear error messages for invalid input
Dependencies	Transaction entry form; SWIFT compliance validation
Notes	Focuses on input validation for security and compliance.

Test Case 65: API Integration: Add New Governmental Payment Type Post Go-Live

Test ID	TC_064
Module	Admin Portal > Governmental Payments Management
Category	Integration
Priority	Medium
Test Type	Positive/Negative
Risk Level	Medium
Estimated Time	10 minutes
Description	Test the API endpoint for adding a new governmental payment type after system go-live, and verify U
Objective	Ensure new payment types can be added via API and are immediately available in the admin portal.
Preconditions	Admin API credentials are available; System is in post go-live state
Expected Result	New governmental payment type is added via API and reflected in the admin portal; duplicate entries

Detailed Test Steps with Data:

Step Number	1
Description	Authenticate with the admin API using valid credentials
Test Data	{ "api_endpoint": "https://api.example.com/v1/auth/login", "username": "api_admin", "password": "ApiAdmin#2024" }
Expected Behavior	Authentication token is returned

Step Number	2
Description	Invoke the 'Add Payment Type' API with new type details
Test Data	{ "api_endpoint": "https://api.example.com/v1/gov-payments/types", "auth_token": "Bearer <token>", "payload": "{\"payment_type_name\": \"Municipal Fees\", \"description\": \"Payments for municipal services\", \"active\": }
Expected Behavior	API returns 201 Created with new payment type ID

Step Number	3
Description	Navigate to the admin portal's governmental payments management section
Test Data	{ "admin_username": "admin_portal", "admin_password": "PortalAdmin2024", "url": "https://admin.example.com/gov-payments" }
Expected Behavior	New payment type 'Municipal Fees' appears in the list

Step Number	4
Description	Attempt to add a duplicate payment type via API
Test Data	{ "api_endpoint": "https://api.example.com/v1/gov-payments/types", "auth_token": "Bearer <token>", "payload": "{\"payment_type_name\": \"Municipal Fees\", \"description\": \"Duplicate entry test\", \"active\": true}" }
Expected Behavior	API returns 409 Conflict with message: 'Payment type already exists.'

Test Data Summary:

{ "overall_input": "API payloads for new and duplicate payment types", "key_parameters": "payment_type_name, description, active" }

Validation Criteria	API adds new payment type and prevents duplicates; UI reflects changes immediately
Dependencies	API endpoint for payment type management; Admin portal UI
Notes	Covers both positive (add) and negative (duplicate) API scenarios.