**UNIT TESTING**

In this section, a detailed account of the testing strategy used to assert functionality of the software produced for this project will be discussed.

To run tests:

* open a terminal inside C:\xampp\htdocs\YHROCU-CLONE\Team6-YHROCU-workflow\SOFTWARE\_AUTH.
* Make sure composer is installed on your computer and run “composer install”
* Run “vendor\bin\phpunit” for all tests or append with UNITTESTING/[TEST FILE] to test a specific file.

See Table Below:

|  |  |  |  |
| --- | --- | --- | --- |
| Test File | Test Name | Description | Output |
| DatabaseConnectionTest.php | testConnectionIsValid() | Test that the database connection is valid by using inc\_connect.php and by asserting the connection is a instance of mysqli class. |  |
|  | testInsertAndFetch() | Test inserting a dummy row into the test\_table and then fetching it. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| Auth0LoginSystemTest.php | testAuth0FactoryReturnsAuth0Instance() | Test that the Auth0Factory returns a valid Auth0 instance. |  |
|  | testLoginDoesNotCrash() | Test that calling the login() method on the Auth0 instance does not crash. |  |
|  | testLogoutUrlGenerated() | Test that the logout() method generates a valid logout URL. |  |
|  | testLogoutClearsSession() | Test that the auth0\_logout.php file clears the user session. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| Auth0CallbackTest.php | testCallbackFlowSetsSessionData() | Tests the Auth0 callback flow. Uses a fake Auth0 instance (FakeAuth0) to simulate the Auth0 SDK behavior. Mocks Auth0UserManager to simulate fetching user data. Verifies that after the callback, the session contains the expected user data. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| Auth0LogoutSystemTest.php | testLogoutClearsSession() | Sets a dummy user in the session. Tests that logout clears session. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| Auth0SessionPersistenceTest.php | testSessionContainsExpectedKeysAfterLogin() | Tests that after simulating a login, the session contains the expected keys and values. Simulates a login by faking an Auth0 user with specific data. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| AdminRoleTest.php | testAdminHasAllPermissions() | Tests that an admin user has all the expected permissions. |  |
|  | testUserHasLimitedPermissions() | Tests that a regular user has only limited permissions. |  |
|  | testManagerCanEditTasks() | Tests that a manager user can edit tasks but cannot manage users. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| AdminUserTest.php | testAccessDeniedForGuest() | Verifies that a guest (with no session) is denied access. Expected output should include "Access Denied". |  |
|  | testAccessDeniedForNonAdmin() | Verifies that a user with role "user" cannot access the admin page. |  |
|  | testAdminGetsAccess() | Verifies that an admin sees the welcome message when accessing the admin page. |  |
|  | testAdminCanCreateUserWithMockedManager() | Mocks Auth0UserManager to simulate successful user creation. Injects the mock into the global space and verifies that a success message appears. |  |
|  | testAccessDeniedForManager() | Verifies that a user with role "manager" is denied admin access. |  |
|  | testAccessDeniedForUnknownRole() | Verifies that a user with an unknown or invalid role is denied access. |  |
|  | testAccessDeniedWhenRoleIsMissing() | Verifies that a user with no role set is denied access. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| CreateTaskPageTest.php | testAccessDeniedForGuest() | If there's no user in session, the page should return JSON error 'Not authorized' in test mode. |  |
|  | testAccessGrantedForAdmin() | If the user is admin, they can view the form, so presumably we get no JSON error and standard HTML (meaning JSON is null). |  |
|  | testInvalidFormSubmissionShowsError() | If we omit required fields (e.g., 'subject' is empty), we expect "Please fill in all required fields." in JSON error. |  |
|  | testTaskCreationWorks() | Submitting valid data should yield "Task created successfully." in JSON and the new task should be in the DB. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| CreateProjectPageTest.php | testAccessDeniedForGuest() | This scenario checks if a user who isn't logged in tries to load the page. We expect a JSON error of 'Not authorized' from test mode. |  |
|  | testAccessGrantedForAdmin() | Here, I'm making sure an admin can view the form. If the user is admin, we expect no JSON error, meaning it outputs normal HTML. |  |
|  | testInvalidFormSubmissionShowsError() | This checks if we pass incomplete data. We expect a JSON error "All fields are required." in test mode. |  |
|  | testProjectCreationWorks() | Here, I submit valid data as an admin and expect "Project created successfully." in the JSON, then confirm the project actually exists in the database. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ViewTaskPageTest.php | testInvalidTaskIdShowsError() | Test that an empty task ID returns a JSON error "Invalid task ID". |  |
|  | testNonexistentTaskShowsError() | Test that a nonexistent task ID (e.g. 99999) returns "Task not found" error. |  |
|  | testViewTaskPageDisplaysTask() | Test that a valid task ID returns correct task details in JSON. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ViewProjectPageTest.php | testInvalidProjectIdShowsError() | Test that an empty or invalid project ID returns an error. |  |
|  | testNonexistentProjectShowsError() | Test that a nonexistent project ID returns a "Project not found" error. |  |
|  | testViewProjectPageDisplaysProject() | Test that a valid project ID returns the correct project details. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| EditTaskPageTest.php | testInvalidTaskId() | Test that "Invalid task ID" JSON error returns when Task ID is invalid. |  |
|  | testNonexistentTask() | Test that "Task not found" JSON error returns when Task not found. |  |
|  | testUnauthorizedUser() | Test that when user is not authorized to see page expect JSON error. |  |
|  | testAllFieldsRequired() | Test that all fields are filled in before submitting form. |  |
|  | testEditTaskSuccessNoAssign() | Test that when no Assigned user email is not sent. |  |
|  | testEditTaskSuccessWithAssign() | Test that when user is Assigned email is sent. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| EditProjectPageTest.php | testInvalidProjectId() | Test that "Invalid project ID" JSON error returns when Project ID is invalid. |  |
|  | testNonexistentProjectShowsError() | Test that "Project not found" JSON error returns when Project not found. |  |
|  | testUnauthorizedUser() | Test that when user is not authorized to see page expect JSON error. |  |
|  | testEditProjectAllFieldsRequired() | Test that all fields are filled in before submitting form. |  |
|  | testEditProjectSuccessAsManager() | Test that when editing as Manager, return JSON "Project updated successfully". |  |
|  | testEditProjectSuccessAsAdmin() | Test that when editing as Admin, return JSON "Project updated successfully". |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| TaskLogsTest.php | testUnauthorizedUser() | Verifies that if a user with a non-staff role (e.g., "user") accesses the page, the JSON response contains "You are not authorized". |  |
|  | testInvalidId() | Verifies that providing an empty task ID returns a JSON error "Invalid task ID". |  |
|  | testNonexistentTask() | Verifies that a task ID of "99999" returns a JSON error "Task not found". |  |
|  | testNoLogsFound() | Verifies that when there are no logs in the system for a given task, the JSON response returns an "info" key with "No logs found". |  |
|  | testMockLogsReturned() | Verifies that when using the mock\_logs flag, the JSON response returns a logs array containing one entry with expected values. |  |
|  | testCsvExportProductionMode() | Forces production mode by setting force\_prod=1 and export=1, then verifies that the CSV export output contains the expected BOM and header fields. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ProjectLogsTest.php | testUnauthorizedUser() | Verifies that if a user with a non-staff role (e.g., "user") accesses the page, the JSON response contains "You are not authorized". |  |
|  | testInvalidProjectId() | Verifies that providing an empty project ID returns a JSON error "Invalid project ID". |  |
|  | testNonexistentProject() | Verifies that a project ID of "99999" returns a JSON error "Project not found". |  |
|  | testNoLogsFound() | Verifies that when there are no logs in the system for a given project, the JSON response returns an info message "No logs found". |  |
|  | testMockLogsReturned() | Verifies that when the mock\_logs flag is provided, the JSON response includes both projectLogs and taskLogs arrays with the expected dummy data. |  |
|  | testCsvExportProductionMode() | Forces production mode using force\_prod=1 and triggers CSV export via export=1. Verifies that the CSV output includes the Excel BOM and key column headers. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ToggleTheme.test.js | Test: Switching from light to dark mode | Initially, the document body is set to "light-mode". After calling toggleTheme(), the body should switch to "dark-mode", the "light-mode" class should be removed, and localStorage should store "dark". |  |
|  | Test: Switching from dark to light mode | Initially, the document body is set to "dark-mode". After calling toggleTheme(), the body should switch to "light-mode", the "dark-mode" class should be removed, and localStorage should store "light". |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ShowHidePassword.test.js | Test: Reveals the Password and Updates the Icon | Simulates clicking the toggle when the password is hidden. Expects the input field's type to change to "text" and the icon source to update to show "eye.svg". |  |
|  | Test: Hides the Password and Updates the Icon | Simulates clicking the toggle when the password is already revealed. Expects the input field's type to revert to "password" and the icon source to update to show "eye-crossed.png". |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| PasswordStrength.test.js | Test: Very Weak Password | Checks that a very short password ("a") is identified as "Very Weak", with a red fill at 20% and the submit button disabled. |  |
|  | Test: Weak Password | Checks that a weak password ("abc123") is identified as "Weak", with an orange fill at 40% and the submit button disabled. |  |
|  | Test: Medium Password | Checks that a medium-strength password ("Abcdef1") is identified as "Medium", with a gold fill at 60% and the submit button disabled. |  |
|  | Test: Good Password | Checks that a good password ("Abcd1234") is identified as "Good", with a lightgreen fill at 80% and the submit button enabled. |  |
|  | Test: Strong Password | Checks that a strong password ("Abcd1234!") is identified as "Strong", with a green fill at 100% and the submit button enabled. |  |