**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 php 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.

To run node.js tests:

* Install Node.js & NPM with commands: “node -v” and “npm -v”
* Install jest with commands: “npm init -y” and “npm install --save-dev jest”
* Run “npm test” or “npx jest path/to/[test file].js”

See Table Below:

|  |  |  |  |
| --- | --- | --- | --- |
| Test File | Test Name | Description | Output |
| DatabaseConnectionTest.php | testConnectionIsValid() | Verifies that the database connection established via inc\_connect.php is valid by asserting it is an instance of the mysqli class. |  |
|  | testInsertAndFetch() | Tests basic database functionality by inserting a dummy row into a test table and retrieving it to confirm data integrity. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| Auth0LoginSystemTest.php | testAuth0FactoryReturnsAuth0Instance() | Ensures that the Auth0Factory correctly returns a valid Auth0 instance. |  |
|  | testLoginDoesNotCrash() | Confirms that calling the login() method on the Auth0 instance does not cause errors or exceptions. |  |
|  | testLogoutUrlGenerated() | Validates that the logout() method successfully generates a proper logout URL. |  |
|  | testLogoutClearsSession() | Confirms that auth0\_logout.php correctly clears the user session data upon logout. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| Auth0CallbackTest.php | testCallbackFlowSetsSessionData() | Simulates the Auth0 callback process using a mock Auth0 instance and Auth0UserManager. Verifies that valid session data is set after authentication. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| Auth0LogoutSystemTest.php | testLogoutClearsSession() | Sets a dummy user in the session and verifies that the logout process successfully clears all session data. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| Auth0SessionPersistenceTest.php | testSessionContainsExpectedKeysAfterLogin() | After simulating a login, checks that the session includes all expected Auth0 keys and values. Uses a fake Auth0 user to validate session persistence. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| AdminRoleTest.php | testAdminHasAllPermissions() | Verifies that a user with the admin role has access to all assigned permissions and capabilities within the system. |  |
|  | testUserHasLimitedPermissions() | Checks that a regular user account has restricted access and cannot perform admin-level actions. |  |
|  | testManagerCanEditTasks() | Confirms that a manager can edit tasks but is restricted from performing admin-related functions such as user management |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| AdminUserTest.php | testAccessDeniedForGuest() | Ensures that an unauthenticated guest (no session) is denied access to protected areas. Output should include an "Access Denied" message. |  |
|  | testAccessDeniedForNonAdmin() | Verifies that users with the "user" role cannot access the admin panel or perform admin-level tasks. |  |
|  | testAdminGetsAccess() | Confirms that an admin successfully sees the welcome message upon accessing the admin dashboard. |  |
|  | testAdminCanCreateUserWithMockedManager() | Mocks Auth0UserManager to simulate user creation. Injects the mock and verifies that a success message appears after the operation. |  |
|  | testAccessDeniedForManager() | Checks that users with the "manager" role are denied access to the admin page or any admin-level permissions. |  |
|  | testAccessDeniedForUnknownRole() | Verifies that users with invalid or unrecognized roles are denied access and redirected or shown an appropriate error. |  |
|  | testAccessDeniedWhenRoleIsMissing() | Ensures that users without a role set in their session are denied access to protected pages.. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ListTaskPageTest.php | testGuestUserRedirectedToLogin() | Confirms that unauthenticated users (guests) are redirected to the login page or shown a "Please log in first" error when attempting to access the dashboard. |  |
|  | testAdminCanViewDashboard() | Verifies that an admin can access the full dashboard view, including all tasks and admin-level options. |  |
|  | testUserSeesAssignedTasksOnly() | Ensures that regular users only see tasks assigned to them and do not have access to task creation features. |  |
|  | testAdminSeesAllTasks() | Confirms that admins can view all tasks, including those assigned to other users. |  |
|  | testUserCannotSeeUnassignedTasks() | Verifies that standard users do not see tasks that are unassigned or not specifically assigned to them. |  |
|  | testUserCanViewDashboard() | Ensures that users can access their dashboard and only see features relevant to their role, such as viewing assigned tasks and hiding admin controls. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| CreateTaskPageTest.php | testAccessDeniedForGuest() | Verifies that unauthenticated users (no session) attempting to access the page receive a JSON error message: “Not authorized” (when in test mode). |  |
|  | testAccessGrantedForAdmin() | Confirms that admins can access the form without triggering JSON errors. The test passes if the response returns standard HTML output. |  |
|  | testInvalidFormSubmissionShowsError() | Submits a form with missing required fields (e.g., empty subject) and checks for a JSON response indicating a validation error. |  |
|  | testTaskCreationWorks() | Submits valid task data and checks for a success message in the JSON response. Also verifies that the task is successfully added to the database. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| CreateProjectPageTest.php | testAccessDeniedForGuest() | Ensures that guests (unauthenticated users) attempting to access the project creation page receive a JSON error: “Not authorized.” |  |
|  | testAccessGrantedForAdmin() | Confirms that admins can access the project creation form, and that the response returns regular HTML without any JSON error when in test mode. |  |
|  | testInvalidFormSubmissionShowsError() | Submits incomplete form data (e.g., missing required fields) and verifies that a JSON error message such as “All fields are required” is returned. |  |
|  | testProjectCreationWorks() | Submits valid project data as an admin and checks for a success message in the JSON response. Also verifies that the project is correctly added to the database. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ViewTaskPageTest.php | testInvalidTaskIdShowsError() | Confirms that submitting an empty or invalid task ID returns a JSON error message like “Invalid task ID.” |  |
|  | testNonexistentTaskShowsError() | Verifies that a request for a task with a non-existent ID (e.g., 99999) returns a JSON error stating “Task not found.” |  |
|  | testViewTaskPageDisplaysTask() | Ensures that providing a valid task ID returns the correct task details in JSON format. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ViewProjectPageTest.php | testInvalidProjectIdShowsError() | Confirms that an empty or invalid project ID triggers an error response, preventing page load or data retrieval. |  |
|  | testNonexistentProjectShowsError() | Validates that accessing a project with a non-existent ID returns an appropriate error message such as “Project not found.” |  |
|  | testViewProjectPageDisplaysProject() | Test that a valid project ID returns the correct project details. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| EditTaskPageTest.php | testInvalidTaskId() | Verifies that submitting an invalid task ID returns a JSON error message such as “Invalid task ID.” |  |
|  | testNonexistentTask() | Confirms that requesting a non-existent task ID (e.g., 99999) returns a JSON error: “Task not found.” |  |
|  | testUnauthorizedUser() | Ensures that unauthorized users (not logged in or lacking permission) receive a JSON access error when trying to access the edit task page. |  |
|  | testAllFieldsRequired() | Validates that all required form fields must be filled in before submission. Missing data results in a JSON validation error. |  |
|  | testEditTaskSuccessNoAssign() | Confirms that if a task is edited without assigning it to a user, no assignment email is triggered |  |
|  | testEditTaskSuccessWithAssign() | Verifies that editing a task and assigning it to a user triggers an assignment email to be sent. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| EditProjectPageTest.php | testInvalidProjectId() | Verifies that submitting an invalid or empty project ID returns a JSON error message: “Invalid project ID.” |  |
|  | testNonexistentProjectShowsError() | Confirms that a request for a non-existent project ID returns a JSON error such as “Project not found.” |  |
|  | testUnauthorizedUser() | Ensures that users without access permissions receive a JSON error when attempting to access the edit project page. |  |
|  | testEditProjectAllFieldsRequired() | Validates that all required project form fields must be filled in before submission. Missing data should trigger a JSON validation error |  |
|  | testEditProjectSuccessAsManager() | Confirms that when a user with the manager role edits a project with valid data, the system returns a JSON success message: “Project updated successfully.” |  |
|  | testEditProjectSuccessAsAdmin() | Verifies that an admin can successfully edit a project and receives the expected JSON message: “Project updated successfully.” |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| TaskLogsTest.php | testUnauthorizedUser() | Verifies that if a user without the appropriate role (e.g., a regular user) accesses the task logs page, a JSON error message is returned: “You are not authorized.” |  |
|  | testInvalidId() | Checks that providing an empty or invalid task ID results in a JSON error: “Invalid task ID.” |  |
|  | testNonexistentTask() | Confirms that using a non-existent task ID (e.g., 99999) returns a JSON error message: “Task not found.” |  |
|  | testNoLogsFound() | Verifies that if no logs exist for a valid task, the JSON response contains an “info” message stating: “No logs found.” |  |
|  | testMockLogsReturned() | When the mock\_logs flag is enabled, confirms that the JSON response returns a logs array containing one entry with expected mock 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 users without appropriate access rights (e.g., role "user") are blocked from accessing the project logs page, returning a JSON error: “You are not authorized.” |  |
|  | testInvalidProjectId() | Checks that submitting an empty or invalid project ID results in a JSON error message: “Invalid project ID.” |  |
|  | testNonexistentProject() | Confirms that using a non-existent project ID (e.g., 99999) returns a JSON error: “Project not found.” |  |
|  | testNoLogsFound() | Verifies that if no logs exist for a valid project, the system returns a JSON response with an informational message: “No logs found.” |  |
|  | testMockLogsReturned() | Ensures that enabling the mock\_logs flag returns a JSON response that includes both projectLogs and taskLogs arrays filled with expected dummy entries |  |
|  | 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. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ToggleDropdown.test.JS | Test: open the dropdown when toggled | When the user clicks on the .ACTION-DROPDOWN-TOGGLE button, the corresponding dropdown becomes active. |  |
|  | Test: Closes other dropdowns when one is opened | When a second dropdown is toggled open, any other currently open dropdowns (with the active class) should be closed. |  |
|  | Test: Closes dropdowns when clicking outside | If a dropdown is open and the user clicks **anywhere else on the page**, it should be closed. |  |
| NEXT FILE | ------------------ | ------------- | --------------------- |
| ResetLinkCopy.test.js | Test: copies reset link and shows alert | When copyResetLink() is called, it copies the value of the #reset-link input and shows an alert with "Link copied to clipboard!". |  |