**Test Strategy and Design**

**I. Test Plan & Test Cases**

**Overall Test Strategy**

The test strategy for the given user stories will encompass both functional and non-functional testing. The focus will be on ensuring that the application meets user requirements, is user-friendly, performs well under load, and is secure. The testing will be divided into three main areas based on the user stories: Web UI, API, and Mobile Native App.

**Test Objectives**

1. Validate the functionality of creating and viewing challenges in the CTFLearner web application.

2. Ensure the Reqres API endpoints for user management function correctly.

3. Verify the mobile trading app's login and portfolio viewing features.

**Test Scope**

- Functional Testing: Verify that all user stories are implemented correctly.

- Non-Functional Testing: Assess performance, security, and usability.

**Test Cases**

**User Story 1 (Web UI - CTFLearner)**

**1. Test Case 1: Create Challenge (Positive Scenario)**

- Precondition: User is logged in.

- Steps:

1. Navigate to the "Create Challenge" page.

2. Fill in the challenge details.

3. Submit the challenge.

- Expected Result: Challenge is created successfully and displayed in "My Challenge".

**2. Test Case 2: Create Challenge (Negative Scenario)**

- Precondition: User is logged in.

- Steps:

1. Navigate to the "Create Challenge" page.

2. Leave required fields empty.

3. Submit the challenge.

- Expected Result: Error message is displayed indicating required fields.

**3. Test Case 3: View My Challenges**

- Precondition: User has created challenges.

- Steps:

1. Navigate to "My Challenge".

- Expected Result: All created challenges are displayed.

User Story 2 (API - Reqres)

**1. Test Case 1: Login (Positive Scenario)**

- Steps:

1. Send POST request to `/login` with valid credentials.

- Expected Result: Successful login response with token.

**2. Test Case 2: Get User (Positive Scenario)**

- Steps:

1. Send GET request to `/users/2` with Bearer Token.

- Expected Result: User details are returned successfully.

**3. Test Case 3: Update User (Negative Scenario)**

- Steps:

1. Send PUT request to `/users/2` with invalid data.

- Expected Result: Error response indicating invalid data.

**User Story 3 (Mobile Native App)**

**1. Test Case 1: Login (Valid Credentials)**

- Steps:

1. Open the mobile app.

2. Enter valid username and password.

3. Click on login.

- Expected Result: User is logged in and redirected to the portfolio view.

**2. Test Case 2: Login (Invalid Credentials)**

- Steps:

1. Open the mobile app.

2. Enter invalid username and password.

3. Click on login.

- Expected Result: Error message indicating invalid credentials.

**3. Test Case 3: View Portfolio**

- Precondition: User is logged in.

- Steps:

1. Navigate to the portfolio section.

- Expected Result: Portfolio details are displayed.

**II. Test Automation Framework Development**

**1. User Story 1**: Web UI Automation

- **Framework Setup**: use Cypress integrate with Browserstack

- **Reusable Login Test Case**:

- Implement a login method in the Page Object Model that can be reused across test cases.

**- Test Cases:**

- **Positive Scenario**: Create a challenge successfully.

- **Negative Scenario**: Attempt to create a challenge with missing fields.

**2. User Story 2:** API Automation

- **Framework Setup**: Use Cypress for API testing.

- **Endpoints:**

- **Login**: Automate login to retrieve a token.

- **Get User**: Automate retrieval of user details.

- **Update User**: Automate updating user information.

- **Test Cases:**

- **Positive Scenario**: Successful login, successful user retrieval, successful user update.

- **Negative Scenario**: Invalid login, invalid user retrieval, invalid user update.

**3. User Story 3**: Mobile Native App Automation

- **Framework Setup**: Use Appium for mobile automation.

- **Test Cases**:

- **Login**: Automate login with valid and invalid credentials.

- **View Portfolio**: Automate the navigation to the portfolio section.

- **UI Verification**: Verify UI elements and navigation.

- **Parallel Execution**: Ensure the framework supports parallel execution on real devices, emulators, and simulators.