**Test Plan Document**

**Test Suite: API Testing for Backend API**

**Test Cases**

**Test Case 1: Create Records**

* Goal: Insert a new record into the backend API
* Expected Result: The API returns a 201 status code and a unique ID for the new record
* Test Method Name: **testCreateRecord**
* Priority: High

|  |
| --- |
| **Automation Script:**  // Method to create a user and return the user ID  **private** String createUser(SoftAssert softAssert) {  // Test data for creating a user  HashMap<String, String> userData = generateUserData();  String userId = **null**;  **try** {  // Send POST request to create user and extract the user ID  userId = given()  .contentType("application/json")  .body(userData)  .when()  .post(baseUrl)  .then()  .statusCode(201)  .extract().path("\_id");  // Log success message  logger.log(Level.INFO, "User created successfully with ID: " + userId);  // Assert that user ID is not null  softAssert.assertNotNull(userId, "User ID should not be null");  } **catch** (Exception e) {  // Log error message  logger.log(Level.SEVERE, "Error occurred while creating user: " + e.getMessage());    // Fail the test with a descriptive message  softAssert.fail("Failed to create user: " + e.getMessage());  }  **return** userId;  } |

**Test Case 2: Retrieve Records**

* Goal: Retrieve a list of records from the backend API
* Expected Result: The API returns a 200 status code and a list of records
* Test Method Name: **testRetrieveRecords**
* Priority: Medium

|  |
| --- |
| **Automation Script:**  // Method to read a user using the provided user ID  **private** **void** readUser(SoftAssert softAssert, String userId) {  **try** {  // Send GET request to read user details  given()  .when()  .get(baseUrl + userId)  .then()  .statusCode(200)  .log().all();  // Log success message  logger.log(Level.INFO, "User read successfully with ID: " + userId);  } **catch** (Exception e) {  // Log error message  logger.log(Level.SEVERE, "Error occurred while reading user: " + e.getMessage());  softAssert.fail("Failed to read user: " + e.getMessage());  }  } |

**Test Case 3: Update Record**

* Goal: Update a specific record in the backend API
* Expected Result: The API returns a 200 status code and the updated record
* Test Method Name: **testUpdateRecord**
* Priority: Medium

|  |
| --- |
| **Automation Script:**  // Method to update a user using the provided user ID  **private** **void** updateUser(SoftAssert softAssert, String userId) {  // Test data for updating a user  HashMap<String, String> updatedUserData = generateUserData();  updatedUserData.put("username", "UpdatedUser");  updatedUserData.put("firstname", "UpdatedFirstName");  **try** {  // Send PUT request to update user details  given()  .contentType("application/json")  .body(updatedUserData)  .when()  .put(baseUrl + userId)  .then()  .statusCode(200)  .log().all();  // Log success message  logger.log(Level.INFO, "User updated successfully with ID: " + userId);  } **catch** (Exception e) {  // Log error message  logger.log(Level.SEVERE, "Error occurred while updating user: " + e.getMessage());  softAssert.fail("Failed to update user: " + e.getMessage());  }  } |

**Test Case 4: Delete Record**

* Goal: Delete a specific record from the backend API
* Expected Result: The API returns a 200 status code and the deleted record is no longer available.

|  |
| --- |
| **Automation Script:**  // Method to delete a user using the provided user ID  **private** **void** deleteUser(SoftAssert softAssert, String userId) {  **try** {  // Send DELETE request to delete user  given()  .when()  .delete(baseUrl + userId)  .then()  .statusCode(204)  .log().all();  // Log success message  logger.log(Level.INFO, "User deleted successfully with ID: " + userId);  } **catch** (Exception e) {  // Log error message  logger.log(Level.SEVERE, "Error occurred while deleting user: " + e.getMessage());  softAssert.fail("Failed to delete user: " + e.getMessage());  }  } |