**Test Cases for AuthController (Login & Sign-up Controller)**

**Test Case 1: Sign-up with Valid Data**

* **Test Objective:** Verify that a new user can successfully sign up with valid credentials.
* **Test Steps:**
  1. Send a POST request to api/auth/signup with valid email and password in the body:

json

Copy code

{

"email": "testuser@example.com",

"password": "ValidPassword123"

}

* 1. Verify that the response status is 200 OK.
  2. Verify the response message is {"message": "User successfully created."}.
* **Expected Result:** User is successfully created, and the response contains a success message.

**Test Case 2: Sign-up with Existing Email**

* **Test Objective:** Verify that the sign-up process fails if the email is already taken.
* **Test Steps:**
  1. Send a POST request to api/auth/signup with an already used email and a new password:

json

Copy code

{

"email": "existinguser@example.com",

"password": "ValidPassword123"

}

* 1. Verify that the response status is 400 BadRequest.
  2. Verify the response message is {"message": "Email is already taken."}.
* **Expected Result:** The request should fail with an appropriate error message.

**Test Case 3: Login with Correct Credentials**

* **Test Objective:** Verify that the user can log in successfully with correct email and password.
* **Test Steps:**
  1. Send a POST request to api/auth/login with a valid email and password:

json

Copy code

{

"email": "testuser@example.com",

"password": "ValidPassword123"

}

* 1. Verify that the response status is 200 OK.
  2. Verify the response message is {"message": "Login successful."}.
* **Expected Result:** The user is logged in successfully with the correct credentials.

**Test Case 4: Login with Incorrect Password**

* **Test Objective:** Verify that the login fails with an incorrect password.
* **Test Steps:**
  1. Send a POST request to api/auth/login with a valid email and incorrect password:

json

Copy code

{

"email": "testuser@example.com",

"password": "InvalidPassword123"

}

* 1. Verify that the response status is 401 Unauthorized.
  2. Verify the response message is {"message": "Invalid password."}.
* **Expected Result:** The login attempt should fail with a 401 Unauthorized response.

**Test Case 5: Get All Credentials**

* **Test Objective:** Verify that the system returns all credentials stored in the database.
* **Test Steps:**
  1. Send a GET request to api/settings/credentials.
  2. Verify that the response status is 200 OK.
  3. Verify that the response contains a list of credentials.
* **Expected Result:** The credentials should be returned as a list.

**Test Case 6: Delete Credential by ID**

* **Test Objective:** Verify that a credential can be deleted by its ID.
* **Test Steps:**
  1. Send a DELETE request to api/settings/credential/delete/{id}, where {id} is the ID of an existing credential.
  2. Verify that the response status is 200 OK.
  3. Verify the response message is {"message": "Credential deleted successfully."}.
* **Expected Result:** The specified credential is deleted successfully, and the correct success message is returned.

**Test Cases for UserStoryDescriptionController (User Story Controller)**

**Test Case 1: Generate Test Cases for Valid User Story**

* **Test Objective:** Verify that the system generates test cases from a valid user story.
* **Test Steps:**
  1. Send a POST request to api/userstorydescription/GenerateTestCases with a valid user story in the request body:

json

Copy code

{

"userStory": "As a user, I want to be able to log in so that I can access my account."

}

* 1. Verify that the response status is 200 OK.
  2. Verify that the response contains generated test cases for the user story.
* **Expected Result:** The system successfully generates and returns the test cases based on the user story.

**Test Case 2: Generate Test Cases for Invalid User Story Format**

* **Test Objective:** Verify that the system returns an error for an invalid user story format.
* **Test Steps:**
  1. Send a POST request to api/userstorydescription/GenerateTestCases with an invalid user story:

json

Copy code

{

"userStory": "This is not a valid user story format"

}

* 1. Verify that the response status is 400 BadRequest.
  2. Verify the response message is "Invalid user story format: This is not a valid user story format".
* **Expected Result:** The system should return a 400 BadRequest with the appropriate error message.

**Test Case 3: Generate Test Cases with Empty User Story**

* **Test Objective:** Verify that the system returns an error if the user story is empty.
* **Test Steps:**
  1. Send a POST request to api/userstorydescription/GenerateTestCases with an empty user story:

json

Copy code

{

"userStory": ""

}

* 1. Verify that the response status is 400 BadRequest.
  2. Verify the response message is "User story cannot be empty."
* **Expected Result:** The system should return a 400 BadRequest indicating that the user story is empty.

**Test Case 4: Export All User Stories from Jira with Missing Parameters**

* **Test Objective:** Verify that the system returns an error if required Jira parameters are missing.
* **Test Steps:**
  1. Send a GET request to api/userstorydescription/ExportAllUserStories without providing the required parameters (domain, project, issueType, username, and apiToken).
  2. Verify that the response status is 400 BadRequest.
  3. Verify the response message is "Please provide domain, project, issue type, username, and API token.".
* **Expected Result:** The system should return a 400 BadRequest with a specific error message about missing parameters.

**Test Case 5: Export All User Stories from Jira with Valid Parameters**

* **Test Objective:** Verify that the system correctly exports user stories from Jira when valid parameters are provided.
* **Test Steps:**
  1. Send a GET request to api/userstorydescription/ExportAllUserStories with valid query parameters:

json

Copy code

{

"domain": "your-jira-instance.atlassian.net",

"project": "PROJECT\_KEY",

"issueType": "User Story",

"username": "your\_username",

"apiToken": "your\_api\_token"

}

* 1. Verify that the response status is 200 OK.
  2. Verify that the response contains a list of Jira issues (user stories) with details such as summary, description, and reporter.
* **Expected Result:** The system should return a list of exported user stories from Jira.

**Test Case Execution Considerations:**

* **Input Validation:** Ensure that all required fields are properly validated before processing any request.
* **Edge Cases:** Consider edge cases such as:
  + Empty or null user stories.
  + Invalid characters in email fields.
  + Invalid Jira API credentials or domain.
* **Concurrency:** Ensure that the system can handle multiple requests (e.g., multiple sign-ups or test case generations) simultaneously without issues.

**Conclusion:**

These test cases cover the major functionalities of the AuthController and UserStoryDescriptionController. The tests ensure that the controllers handle common scenarios like user sign-up, login, generating test cases, and exporting Jira issues effectively, while also handling errors and edge cases gracefully.