State Transition Testing

To validate the EduTrack application using state transition testing, we will create a state transition table, identify the test requirements to cover every state and transition, and then execute the test cases. Here's the approach to follow:

Step-by-Step Process

1. **Identify the States and Transitions:**
   - Identify all the states in the system.
   - Identify all the possible transitions between the states.
2. **Create the State Transition Table:**
   - Represent the states and transitions in a table format.
   - Include both valid and invalid transitions.
3. **Design Test Cases:**
   - Create test cases to cover every state.
   - Create test cases to cover every transition.
   - Include test cases for invalid transitions.
4. **Execute Test Cases:**
   - Implement the test cases in Java.
   - Run the test cases to validate the application.

State Transition Table

Below is an example state transition table for the EduTrack application. This table includes various states and the corresponding transitions.

State	Event	Next State	Action
S1	Start	S2	Initialize
S2	Login	S3	Validate credentials
S2	Login	S4 (invalid)	Show error message
S3	Add Course	S5	Add course to user profile
S3	Add Course	S6 (invalid)	Show error message
S3	Logout	S2	Logout user
S5	Add Task	S7	Add task to course
S5	Add Task	S8 (invalid)	Show error message
S7	Mark Complete	S5	Update task status
S7	Logout	S2	Logout user

Test Cases for State Transitions
We will create test cases based on the state transition table to ensure all states and transitions are covered, including invalid transitions.
Test Cases:

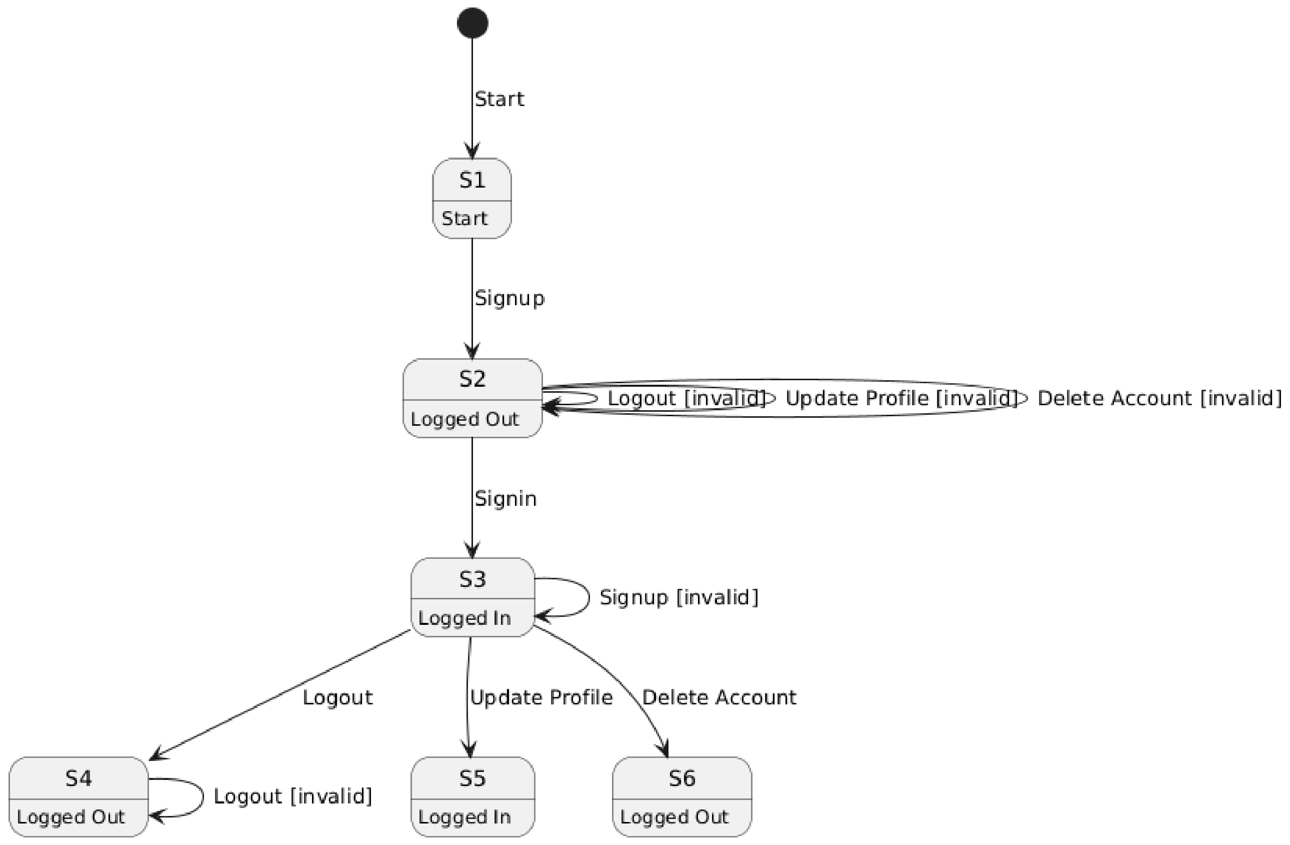
1. **Test Case 1: Valid Login**
    - Initial State: S2
    - Event: Login with valid credentials
    - Expected State: S3
    - Expected Action: Validate credentials
2. **Test Case 2: Invalid Login**
    - Initial State: S2
    - Event: Login with invalid credentials
    - Expected State: S4 (invalid)
    - Expected Action: Show error message
3. **Test Case 3: Add Course**
    - Initial State: S3
    - Event: Add Course
    - Expected State: S5
    - Expected Action: Add course to user profile
4. **Test Case 4: Invalid Add Course**
    - Initial State: S3
    - Event: Add Course with invalid data
    - Expected State: S6 (invalid)
    - Expected Action: Show error message
5. **Test Case 5: Logout**
    - Initial State: S3
    - Event: Logout
    - Expected State: S2
    - Expected Action: Logout user
6. **Test Case 6: Add Task**
    - Initial State: S5
    - Event: Add Task
    - Expected State: S7
    - Expected Action: Add task to course
7. **Test Case 7: Invalid Add Task**
    - Initial State: S5
    - Event: Add Task with invalid data
    - Expected State: S8 (invalid)
    - Expected Action: Show error message
8. **Test Case 8: Mark Task Complete**
    - Initial State: S7
    - Event: Mark Complete
    - Expected State: S5
    - Expected Action: Update task status

Implementing Test Cases in Java

Here is how you can implement the test cases using JUnit in Java:

```
package PathTesting;

import Proj_375_Classes.UserManager;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class StateTransitionTest {

  private UserManager userManager;

  @BeforeEach
  public void setUp() {
    userManager = new UserManager();
  }

  @Test
  public void testSignup_TransitionToLoggedIn() {
    System.out.println("Running test case: Sign Up Transition to Logged In...");
    assertTrue(userManager.signup("user1", "Password1!", "Password1!",
"user1@example.com", "User One"));
    assertFalse(userManager.isLoggedIn("user1")); // Initially, user is not logged in
  }

  @Test
  public void testLogin_TransitionToLoggedIn() {
    userManager.signup("user2", "Password1!", "Password1!", "user2@example.com",
"User Two");
    System.out.println("Running test case: Log In Transition to Logged In...");
    userManager.signin("user2", "Password1!");
    assertTrue(userManager.isLoggedIn("user2"));
  }

  @Test
  public void testLogout_TransitionToLoggedOut() {
    userManager.signup("user3", "Password1!", "Password1!", "user3@example.com",
"User Three");
    userManager.signin("user3", "Password1!");
    System.out.println("Running test case: Log Out Transition to Logged Out...");
    userManager.logout("user3");
    assertFalse(userManager.isLoggedIn("user3"));
  }

  @Test
  public void testUpdateProfile_StayLoggedIn() {
    userManager.signup("user4", "Password1!", "Password1!", "user4@example.com",
"User Four");
    userManager.signin("user4", "Password1!");
    System.out.println("Running test case: Update Profile, Stay Logged In...");
    assertTrue(userManager.updateProfile("user4", "newemail@example.com", "New
Name"));
    assertTrue(userManager.isLoggedIn("user4"));
  }

  @Test
  public void testDeleteAccount_TransitionToLoggedOut() {
    userManager.signup("user5", "Password1!", "Password1!", "user5@example.com",
"User Five");
    userManager.signin("user5", "Password1!");
    System.out.println("Running test case: Delete Account, Transition to Logged Out...");
    assertTrue(userManager.deleteAccount("user5"));
    assertFalse(userManager.isLoggedIn("user5"));
  }
}
```

Here's the PlantUML representation of the state transitions:



Conclusion
By following this approach, we ensure that every state and transition is tested, including invalid transitions. This comprehensive testing strategy helps in identifying issues and ensuring the robustness of the EduTrack application.