Here's a detailed automated testing suite for the login and fund transfer functionalities of a banking application. This suite includes test cases for scenarios such as multiple concurrent user logins, incorrect PIN entries, and fund transfer validations.

1. Login Functionality Testing

Test Case 1: Valid Login

Objective: Verify a user can log in with correct credentials.

Pre-condition: User account exists in the system.

Test Steps:

Navigate to the login page.

Enter valid username and PIN.

Click on the "Login" button.

Expected Result: User successfully logs in and is redirected to the dashboard.

Test Case 2: Invalid Login (Incorrect PIN)

Objective: Verify the system prevents login with an incorrect PIN.

Pre-condition: User account exists in the system.

Test Steps:

Navigate to the login page.

Enter valid username but an incorrect PIN.

Click on the "Login" button.

Expected Result: Error message displayed: "Incorrect PIN. Please try again."

Test Case 3: Account Lockout After Multiple Incorrect Attempts

Objective: Verify the account is locked after a defined number of incorrect login attempts.

Pre-condition: User account exists with a lockout threshold (e.g., 3 failed attempts).

Test Steps:

Navigate to the login page.

Enter valid username and incorrect PIN 3 times.

Attempt a 4th login.

Expected Result: Error message: "Account locked. Please contact support."

Test Case 4: Concurrent Logins

Objective: Verify the application handles multiple concurrent logins for the same user.

Pre-condition: User account exists, and two devices are available.

Test Steps:

Log in on Device A with valid credentials.

Simultaneously log in on Device B with the same credentials.

Expected Result:

If allowed, both sessions are active.

If not allowed, the second login invalidates the first or denies access.

Test Case 5: Session Timeout

Objective: Verify that an idle session times out after a defined period.

Pre-condition: User account exists.

Test Steps:

Log in with valid credentials.

Leave the session idle for the timeout period (e.g., 10 minutes).

Attempt to perform an action after the timeout.

Expected Result: User is logged out and redirected to the login page.

2. Fund Transfer Functionality Testing

Test Case 6: Insufficient Funds

Objective: Verify that a transfer is denied if the source account has insufficient funds.

Pre-condition: Source account balance is less than the transfer amount.

Test Steps:

Log in to the application.

Navigate to the fund transfer page.

Enter a transfer amount greater than the account balance.

Confirm the transfer.

Expected Result: Error message: "Insufficient funds."

Test Case 7: Transfer to Invalid Account

Objective: Verify that a transfer is denied to an invalid or non-existent account.

Pre-condition: Destination account does not exist.

Test Steps:

Log in to the application.

Navigate to the fund transfer page.

Enter valid source account details and an invalid destination account number.

Confirm the transfer.

Expected Result: Error message: "Invalid account. Please check the details."

Test Case 8: Daily Transfer Limit Exceeded

Objective: Verify the system enforces daily transfer limits.

Pre-condition: Daily transfer limit is defined (e.g., $10,000).

Test Steps:

Log in to the application.

Perform transfers until the daily limit is reached.

Attempt another transfer exceeding the limit.

Expected Result: Error message: "Daily transfer limit exceeded."

Tools: Use tools like Selenium, Appium, or Cypress for front-end testing, and Postman or JMeter for API testing.

Test Data Management: Use a separate database for test accounts with configurable balances and limits.

Test Execution: Schedule test runs via CI/CD pipelines (e.g., Jenkins or GitHub Actions).

Reporting: Generate detailed reports using tools like Allure or TestNG.