# Test Plan for Para Bank Application

## 1. Introduction

This Test Plan document outlines the testing strategy and approach for the Para Bank application, detailing the objectives, scope, types of testing, resources, schedule, and deliverables to ensure a high-quality application.

## 2. Objectives

The primary objectives of the test plan are:

* To verify that all functional and non-functional requirements of the Para Bank application are met.
* To identify and resolve any defects or issues in the application.
* To ensure the application is stable, secure, and performs efficiently under expected workloads.

## 3. Scope

The scope of testing includes the following modules of the Para Bank application:

* User Registration and Login
* Account Management
* Fund Transfers
* Bill Payments
* Loan Applications
* Viewing and Downloading Statements

## 4. Test Items

The following items will be tested:

* User Interface (UI)
* Application Programming Interfaces (APIs)
* Functionalities as per the requirement specification
* Performance under various loads
* Security aspects to protect user data

## 5. Types of Testing

The following types of testing will be conducted:

* **Functional Testing**: Validate each function of the software application operates in conformance with the requirement specification.
* **Regression Testing**: Ensure recent code changes have not adversely affected existing functionalities.
* **Integration Testing**: Validate the interfaces between different modules.
* **Performance Testing**: Assess the speed, responsiveness, and stability of the application under a particular workload.
* **Security Testing**: Identify vulnerabilities and ensure data protection mechanisms are robust.
* **User Acceptance Testing (UAT)**: Verify the application meets user needs and requirements.

## 6. Testing Approach

* **Manual Testing**: All test cases will initially be executed manually to ensure functionality and usability.
* **Automation Testing**: Regression and performance testing will be automated using suitable tools.

## 7. Test Environment

The test environment will include:

* **Test Server**: Configured to mimic the production environment.
* **Browsers**: Latest versions of popular browsers (Chrome, Firefox, Safari, Edge).
* **Devices**: Desktop, tablet, and mobile devices for responsive testing.

## 8. Deliverables

* **Test Strategy Document**
* **Test Plan Document**
* **Test Scenarios List**
* **Detailed Test Cases**
* **Requirements Traceability Matrix (RTM)**
* **Test Execution Reports**
* **Defect Reports**
* **Final Test Summary Report**

## 9. Test Schedule

The testing schedule is as follows:

* **Test Planning**: 1 week
* **Test Case Development**: 2 weeks
* **Test Environment Setup**: 1 week
* **Test Execution**: 3 weeks
* **Defect Reporting and Fixing**: Continuous during test execution
* **Final Regression Testing**: 1 week
* **User Acceptance Testing (UAT)**: 1 week

## 10. Resources

* **Test Manager**: Responsible for overall test planning and execution.
* **Test Lead**: Coordinates the testing activities and team.
* **Testers**: Execute test cases and report defects.
* **Automation Engineers**: Develop and maintain automated test scripts.
* **Developers**: Fix defects identified during testing.
* **Business Analysts**: Verify requirements and assist with UAT.

## 11. Entry and Exit Criteria

### Entry Criteria

* Requirement documents are complete and signed off.
* Test environment is set up and configured.
* Test cases are reviewed and approved.

### Exit Criteria

* All critical and major defects are resolved.
* Test execution is complete with no open high-priority defects.
* UAT is signed off by stakeholders.

## 12. Risk Management

### Risks

* Delays in environment setup.
* Unavailability of key resources.
* High number of defects found during testing.

### Mitigation

* Plan for backup resources.
* Schedule buffer time for environment setup.
* Continuous communication with development teams for quick resolution of defects.

## 13. Approval

The Test Plan document must be approved by the following stakeholders:

* Project Manager
* Test Manager
* Business Analyst

Development Lead

# Test Scenarios for Para Bank Application

**1. User Registration and Login**

1.1 Verify that a new user can register with valid details.

1.2 Verify that the system displays an error when attempting to register with an already existing email.

1.3 Verify that the user receives a confirmation email upon successful registration.

1.4 Verify that a registered user can log in with valid credentials.

1.5 Verify that the system displays an error for incorrect username or password.

1.6 Verify that the "Forgot Password" functionality sends a reset link to the user's email.

1.7 Verify that the user can reset their password using the link sent to their email.

**2. Account Management**

2.1 Verify that the user can create a new account (e.g., savings, checking).

2.2 Verify that the user can view a list of their accounts.

2.3 Verify that the user can view account details (e.g., balance, transactions).

2.4 Verify that the user can edit account information.

2.5 Verify that the user can delete an account.

2.6 Verify that the system displays an error when trying to delete an account with a non-zero balance.

**3. Fund Transfers**

3.1 Verify that the user can transfer funds between their own accounts.

3.2 Verify that the user can transfer funds to another user's account.

3.3 Verify that the system displays an error when trying to transfer funds exceeding the available balance.

3.4 Verify that the user receives a confirmation message upon successful transfer.

3.5 Verify that the transaction history is updated after a fund transfer.

**4. Bill Payments**

4.1 Verify that the user can add a new bill payee.

4.2 Verify that the user can schedule a bill payment.

4.3 Verify that the user can view scheduled bill payments.

4.4 Verify that the user can edit a scheduled bill payment.

4.5 Verify that the user can delete a scheduled bill payment.

4.6 Verify that the user receives a confirmation message upon successful bill payment.

**5. Loan Applications**

5.1 Verify that the user can apply for a loan with valid details.

5.2 Verify that the system displays an error when mandatory fields are left blank during loan application.

5.3 Verify that the user can view the status of their loan application.

5.4 Verify that the user receives a notification upon loan approval or rejection.

**6. Viewing and Downloading Statements**

6.1 Verify that the user can view their account statements.

6.2 Verify that the user can download their account statements in PDF format.

6.3 Verify that the system displays an error when trying to view statements for a non-existent account.

**7. User Interface (UI)**

7.1 Verify that all UI elements are displayed correctly on various screen sizes (responsive design).

7.2 Verify that the application is usable and accessible (e.g., screen reader compatibility, keyboard navigation).

7.3 Verify that error messages are displayed in a user-friendly manner.

**8. Performance Testing**

8.1 Verify that the application loads within acceptable time limits under normal load conditions.

8.2 Verify that the application maintains acceptable performance levels under peak load conditions.

**9. Security Testing**

9.1 Verify that the user passwords are stored securely (e.g., hashed, encrypted).

9.2 Verify that the application is protected against common security threats (e.g., SQL injection, XSS).

9.3 Verify that user sessions expire after a period of inactivity.

9.4 Verify that the application enforces strong password policies.

**10. Integration Testing**

10.1 Verify that the application correctly integrates with external payment gateways.

10.2 Verify that the application correctly integrates with email services for notifications.

10.3 Verify that data synchronization between different modules (e.g., account management, transactions) is accurate.

**11. Regression Testing**

11.1 Verify that all critical functionalities work as expected after code changes.

11.2 Verify that no new defects are introduced into the existing functionalities after updates.

**12. User Acceptance Testing (UAT)**

12.1 Verify that the application meets all business requirements and user needs.

12.2 Verify that end-users can perform their tasks without issues.

These test scenarios cover the critical functionalities and aspects of the Para Bank application, ensuring comprehensive testing and validation of the application’s features and performance.

# Test Cases for each Scenarios for Para Bank Application

### 1. User Registration and Login

#### 1.1 Verify that a new user can register with valid details

* **Test Case ID**: TC-UR-001
* **Test Scenario**: User Registration
* **Preconditions**: None
* **Test Steps**:
  1. Navigate to the registration page.
  2. Enter valid user details (name, email, password, etc.).
  3. Click on the "Register" button.
* **Expected Results**: User should be registered successfully and a confirmation message should be displayed.

#### 1.2 Verify that the system displays an error when attempting to register with an already existing email

* **Test Case ID**: TC-UR-002
* **Test Scenario**: Duplicate Email Registration
* **Preconditions**: An account with the same email already exists.
* **Test Steps**:
  1. Navigate to the registration page.
  2. Enter details with an email that already exists in the system.
  3. Click on the "Register" button.
* **Expected Results**: An error message indicating that the email is already registered should be displayed.

#### 1.3 Verify that the user receives a confirmation email upon successful registration

* **Test Case ID**: TC-UR-003
* **Test Scenario**: Registration Confirmation Email
* **Preconditions**: None
* **Test Steps**:
  1. Complete the registration process with valid details.
* **Expected Results**: A confirmation email should be sent to the user's email address.

#### 1.4 Verify that a registered user can log in with valid credentials

* **Test Case ID**: TC-UL-001
* **Test Scenario**: User Login
* **Preconditions**: User is registered.
* **Test Steps**:
  1. Navigate to the login page.
  2. Enter valid username and password.
  3. Click on the "Login" button.
* **Expected Results**: User should be logged in successfully and redirected to the dashboard.

#### 1.5 Verify that the system displays an error for incorrect username or password

* **Test Case ID**: TC-UL-002
* **Test Scenario**: Invalid Login
* **Preconditions**: None
* **Test Steps**:
  1. Navigate to the login page.
  2. Enter an incorrect username or password.
  3. Click on the "Login" button.
* **Expected Results**: An error message indicating incorrect username or password should be displayed.

#### 1.6 Verify that the "Forgot Password" functionality sends a reset link to the user's email

* **Test Case ID**: TC-UL-003
* **Test Scenario**: Forgot Password
* **Preconditions**: User is registered.
* **Test Steps**:
  1. Navigate to the login page.
  2. Click on the "Forgot Password" link.
  3. Enter the registered email address.
  4. Click on the "Submit" button.
* **Expected Results**: A password reset link should be sent to the user's email address.

#### 1.7 Verify that the user can reset their password using the link sent to their email

* **Test Case ID**: TC-UL-004
* **Test Scenario**: Password Reset
* **Preconditions**: User has requested a password reset.
* **Test Steps**:
  1. Open the email with the reset link.
  2. Click on the reset link.
  3. Enter a new password.
  4. Confirm the new password.
  5. Click on the "Reset Password" button.
* **Expected Results**: The password should be reset successfully, and the user should be able to log in with the new password.

### 2. Account Management

#### 2.1 Verify that the user can create a new account (e.g., savings, checking)

* **Test Case ID**: TC-AM-001
* **Test Scenario**: Create Account
* **Preconditions**: User is logged in.
* **Test Steps**:
  1. Navigate to the account creation page.
  2. Select the type of account (savings, checking).
  3. Enter the required details.
  4. Click on the "Create Account" button.
* **Expected Results**: The new account should be created successfully and listed under the user's accounts.

#### 2.2 Verify that the user can view a list of their accounts

* **Test Case ID**: TC-AM-002
* **Test Scenario**: View Accounts
* **Preconditions**: User has one or more accounts.
* **Test Steps**:
  1. Navigate to the accounts page.
* **Expected Results**: A list of all the user's accounts should be displayed.

#### 2.3 Verify that the user can view account details (e.g., balance, transactions)

* **Test Case ID**: TC-AM-003
* **Test Scenario**: View Account Details
* **Preconditions**: User has one or more accounts.
* **Test Steps**:
  1. Navigate to the accounts page.
  2. Click on an account to view its details.
* **Expected Results**: Account details such as balance and recent transactions should be displayed.

#### 2.4 Verify that the user can edit account information

* **Test Case ID**: TC-AM-004
* **Test Scenario**: Edit Account Information
* **Preconditions**: User has an account.
* **Test Steps**:
  1. Navigate to the account details page.
  2. Click on the "Edit" button.
  3. Modify the account information.
  4. Click on the "Save" button.
* **Expected Results**: The account information should be updated successfully.

#### 2.5 Verify that the user can delete an account

* **Test Case ID**: TC-AM-005
* **Test Scenario**: Delete Account
* **Preconditions**: User has an account with a zero balance.
* **Test Steps**:
  1. Navigate to the account details page.
  2. Click on the "Delete" button.
  3. Confirm the deletion.
* **Expected Results**: The account should be deleted successfully.

#### 2.6 Verify that the system displays an error when trying to delete an account with a non-zero balance

* **Test Case ID**: TC-AM-006
* **Test Scenario**: Delete Account with Balance
* **Preconditions**: User has an account with a non-zero balance.
* **Test Steps**:
  1. Navigate to the account details page.
  2. Click on the "Delete" button.
* **Expected Results**: An error message indicating that the account cannot be deleted due to a non-zero balance should be displayed.

### 3. Fund Transfers

#### 3.1 Verify that the user can transfer funds between their own accounts

* **Test Case ID**: TC-FT-001
* **Test Scenario**: Internal Fund Transfer
* **Preconditions**: User has at least two accounts.
* **Test Steps**:
  1. Navigate to the fund transfer page.
  2. Select the source and destination accounts.
  3. Enter the transfer amount.
  4. Click on the "Transfer" button.
* **Expected Results**: The funds should be transferred successfully, and the balances of the source and destination accounts should be updated.

#### 3.2 Verify that the user can transfer funds to another user's account

* **Test Case ID**: TC-FT-002
* **Test Scenario**: External Fund Transfer
* **Preconditions**: User knows the recipient's account details.
* **Test Steps**:
  1. Navigate to the fund transfer page.
  2. Select the source account and enter the recipient's account details.
  3. Enter the transfer amount.
  4. Click on the "Transfer" button.
* **Expected Results**: The funds should be transferred successfully, and the balance of the source account should be updated.

#### 3.3 Verify that the system displays an error when trying to transfer funds exceeding the available balance

* **Test Case ID**: TC-FT-003
* **Test Scenario**: Insufficient Funds Transfer
* **Preconditions**: User's account balance is less than the transfer amount.
* **Test Steps**:
  1. Navigate to the fund transfer page.
  2. Select the source and destination accounts.
  3. Enter an amount greater than the available balance.
  4. Click on the "Transfer" button.
* **Expected Results**: An error message indicating insufficient funds should be displayed.

#### 3.4 Verify that the user receives a confirmation message upon successful transfer

* **Test Case ID**: TC-FT-004
* **Test Scenario**: Fund Transfer Confirmation
* **Preconditions**: None
* **Test Steps**:
  1. Perform a fund transfer.
* **Expected Results**: A confirmation message indicating the transfer was successful should be displayed.

#### 3.5 Verify that the transaction history is updated after a fund transfer

* **Test Case ID**: TC-FT-005
* **Test Scenario**: Transaction History Update
* **Preconditions**: None
* **Test Steps**:
  1. Perform a fund transfer.
  2. Navigate to the transaction history page.
* **Expected Results**: The transaction should be listed in the transaction history.

### 4. Bill Payments

#### 4.1 Verify that the user can add a new bill payee

* **Test Case ID**: TC-BP-001
* **Test Scenario**: Add Bill Payee
* **Preconditions**: None
* **Test Steps**:
  1. Navigate to the bill payment page.
  2. Click on "Add Payee".
  3. Enter payee details.
  4. Click on "Save".
* **Expected Results**: The new bill payee should be added successfully.

#### 4.2 Verify that the user can schedule a bill payment

* **Test Case ID**: TC-BP-002
* **Test Scenario**: Schedule Bill Payment
* **Preconditions**: User has added a payee.
* **Test Steps**:
  1. Navigate to the bill payment page.
  2. Select a payee.
  3. Enter payment details and schedule date.
  4. Click on "Schedule Payment".
* **Expected Results**: The bill payment should be scheduled successfully.

#### 4.3 Verify that the user can view scheduled bill payments

* **Test Case ID**: TC-BP-003
* **Test Scenario**: View Scheduled Payments
* **Preconditions**: User has scheduled bill payments.
* **Test Steps**:
  1. Navigate to the bill payment page.
  2. Click on "View Scheduled Payments".
* **Expected Results**: A list of scheduled bill payments should be displayed.

#### 4.4 Verify that the user can edit a scheduled bill payment

* **Test Case ID**: TC-BP-004
* **Test Scenario**: Edit Scheduled Payment
* **Preconditions**: User has scheduled bill payments.
* **Test Steps**:
  1. Navigate to the scheduled payments page.
  2. Click on "Edit" next to a scheduled payment.
  3. Modify the payment details.
  4. Click on "Save".
* **Expected Results**: The scheduled payment should be updated successfully.

#### 4.5 Verify that the user can delete a scheduled bill payment

* **Test Case ID**: TC-BP-005
* **Test Scenario**: Delete Scheduled Payment
* **Preconditions**: User has scheduled bill payments.
* **Test Steps**:
  1. Navigate to the scheduled payments page.
  2. Click on "Delete" next to a scheduled payment.
  3. Confirm the deletion.
* **Expected Results**: The scheduled payment should be deleted successfully.

#### 4.6 Verify that the user receives a confirmation message upon successful bill payment

* **Test Case ID**: TC-BP-006
* **Test Scenario**: Bill Payment Confirmation
* **Preconditions**: None
* **Test Steps**:
  1. Perform a bill payment.
* **Expected Results**: A confirmation message indicating the payment was successful should be displayed.

### 5. Loan Applications

#### 5.1 Verify that the user can apply for a loan with valid details

* **Test Case ID**: TC-LA-001
* **Test Scenario**: Apply for Loan
* **Preconditions**: User is logged in.
* **Test Steps**:
  1. Navigate to the loan application page.
  2. Enter loan details.
  3. Click on "Submit Application".
* **Expected Results**: The loan application should be submitted successfully.

#### 5.2 Verify that the system displays an error when mandatory fields are left blank during loan application

* **Test Case ID**: TC-LA-002
* **Test Scenario**: Mandatory Fields Validation
* **Preconditions**: None
* **Test Steps**:
  1. Navigate to the loan application page.
  2. Leave mandatory fields blank.
  3. Click on "Submit Application".
* **Expected Results**: An error message indicating that mandatory fields are required should be displayed.

#### 5.3 Verify that the user can view the status of their loan application

* **Test Case ID**: TC-LA-003
* **Test Scenario**: View Loan Application Status
* **Preconditions**: User has applied for a loan.
* **Test Steps**:
  1. Navigate to the loan application status page.
* **Expected Results**: The status of the loan application should be displayed.

#### 5.4 Verify that the user receives a notification upon loan approval or rejection

* **Test Case ID**: TC-LA-004
* **Test Scenario**: Loan Application Notification
* **Preconditions**: User has applied for a loan.
* **Test Steps**:
  1. Approve or reject the loan application.
* **Expected Results**: The user should receive a notification about the loan approval or rejection.

### 6. Viewing and Downloading Statements

#### 6.1 Verify that the user can view their account statements

* **Test Case ID**: TC-VS-001
* **Test Scenario**: View Account Statements
* **Preconditions**: User has an account with transactions.
* **Test Steps**:
  1. Navigate to the account statements page.
* **Expected Results**: The account statements should be displayed.

#### 6.2 Verify that the user can download their account statements in PDF format

* **Test Case ID**: TC-VS-002
* **Test Scenario**: Download Account Statements
* **Preconditions**: User has an account with transactions.
* **Test Steps**:
  1. Navigate to the account statements page.
  2. Click on "Download PDF".
* **Expected Results**: The account statements should be downloaded in PDF format.

#### 6.3 Verify that the system displays an error when trying to view statements for a non-existent account

* **Test Case ID**: TC-VS-003
* **Test Scenario**: View Statements for Non-existent Account
* **Preconditions**: None
* **Test Steps**:
  1. Navigate to the account statements page.
  2. Select a non-existent account.
* **Expected Results**: An error message indicating that the account does not exist should be displayed.

### 7. User Interface (UI)

#### 7.1 Verify that all UI elements are displayed correctly on various screen sizes (responsive design)

* **Test Case ID**: TC-UI-001
* **Test Scenario**: Responsive Design
* **Preconditions**: None
* **Test Steps**:
  1. Open the application on different devices (desktop, tablet, mobile).
* **Expected Results**: All UI elements should be displayed correctly and adjust to the screen size.

#### 7.2 Verify that the application is usable and accessible (e.g., screen reader compatibility, keyboard navigation)

* **Test Case ID**: TC-UI-002
* **Test Scenario**: Usability and Accessibility
* **Preconditions**: None
* **Test Steps**:
  1. Use the application with a screen reader.
  2. Navigate through the application using keyboard only.
* **Expected Results**: The application should be usable and accessible with a screen reader and keyboard navigation.

#### 7.3 Verify that error messages are displayed in a user-friendly manner

* **Test Case ID**: TC-UI-003
* **Test Scenario**: User-friendly Error Messages
* **Preconditions**: None
* **Test Steps**:
  1. Trigger an error (e.g., invalid login, form validation error).
* **Expected Results**: Error messages should be clear, informative, and user-friendly.

### 8. Performance Testing

#### 8.1 Verify that the application loads within acceptable time limits under normal load conditions

* **Test Case ID**: TC-PT-001
* **Test Scenario**: Normal Load Performance
* **Preconditions**: None
* **Test Steps**:
  1. Measure the page load time under normal user load.
* **Expected Results**: The application should load within acceptable time limits (e.g., less than 3 seconds).

#### 8.2 Verify that the application maintains acceptable performance levels under peak load conditions

* **Test Case ID**: TC-PT-002
* **Test Scenario**: Peak Load Performance
* **Preconditions**: None
* **Test Steps**:
  1. Measure the application performance under peak user load.
* **Expected Results**: The application should maintain acceptable performance levels without crashes or significant slowdowns.

### 9. Security Testing

#### 9.1 Verify that the user passwords are stored securely (e.g., hashed, encrypted)

* **Test Case ID**: TC-ST-001
* **Test Scenario**: Secure Password Storage
* **Preconditions**: None
* **Test Steps**:
  1. Check the database for user password storage method.
* **Expected Results**: User passwords should be stored securely using hashing or encryption.

#### 9.2 Verify that the application is protected against common security threats (e.g., SQL injection, XSS)

* **Test Case ID**: TC-ST-002
* **Test Scenario**: Protection Against Security Threats
* **Preconditions**: None
* **Test Steps**:
  1. Attempt SQL injection and XSS attacks on input fields.
* **Expected Results**: The application should not be vulnerable to these attacks.

#### 9.3 Verify that user sessions expire after a period of inactivity

* **Test Case ID**: TC-ST-003
* **Test Scenario**: Session Expiration
* **Preconditions**: User is logged in.
* **Test Steps**:
  1. Stay inactive for a defined period (e.g., 15 minutes).
* **Expected Results**: The user session should expire, and the user should be logged out automatically.

#### 9.4 Verify that the application enforces strong password policies

* **Test Case ID**: TC-ST-004
* **Test Scenario**: Password Policy Enforcement
* **Preconditions**: None
* **Test Steps**:
  1. Attempt to set a weak password during registration or password change.
* **Expected Results**: The application should enforce strong password policies (e.g., minimum length, complexity requirements).

### 10. Integration Testing

#### 10.1 Verify that the application correctly integrates with external payment gateways

* **Test Case ID**: TC-IT-001
* **Test Scenario**: Payment Gateway Integration
* **Preconditions**: None
* **Test Steps**:
  1. Perform a transaction using an external payment gateway.
* **Expected Results**: The transaction should be processed successfully, and the payment status should be updated in the application.

#### 10.2 Verify that the application correctly integrates with email services for notifications

* **Test Case ID**: TC-IT-002
* **Test Scenario**: Email Service Integration
* **Preconditions**: None
* **Test Steps**:
  1. Trigger an event that sends an email notification (e.g., registration, password reset).
* **Expected Results**: The email notification should be sent and received successfully.

#### 10.3 Verify that data synchronization between different modules (e.g., account management, transactions) is accurate

* **Test Case ID**: TC-IT-003
* **Test Scenario**: Data Synchronization
* **Preconditions**: None
* **Test Steps**:
  1. Perform actions in different modules (e.g., create account, transfer funds).
* **Expected Results**: Data should be synchronized accurately between modules (e.g., account balances should update correctly after a fund transfer).

### 11. Regression Testing

#### 11.1 Verify that all critical functionalities work as expected after code changes

* **Test Case ID**: TC-RT-001
* **Test Scenario**: Critical Functionality Regression
* **Preconditions**: Code changes have been made.
* **Test Steps**:
  1. Execute test cases for critical functionalities.
* **Expected Results**: All critical functionalities should work as expected without any defects.

#### 11.2 Verify that no new defects are introduced into the existing functionalities after updates

* **Test Case ID**: TC-RT-002
* **Test Scenario**: New Defects Regression
* **Preconditions**: Code changes have been made.
* **Test Steps**:
  1. Execute test cases for existing functionalities.
* **Expected Results**: No new defects should be introduced, and existing functionalities should work as expected.

### 12. User Acceptance Testing (UAT)

#### 12.1 Verify that the application meets all business requirements and user needs

* **Test Case ID**: TC-UAT-001
* **Test Scenario**: Business Requirements Verification
* **Preconditions**: None
* **Test Steps**:
  1. Review business requirements.
  2. Execute test cases to verify functionalities against requirements.
* **Expected Results**: The application should meet all business requirements and user needs.

#### 12.2 Verify that end-users can perform their tasks without issues

* **Test Case ID**: TC-UAT-002
* **Test Scenario**: End-User Task Verification
* **Preconditions**: None
* **Test Steps**:
  1. Have end-users perform common tasks (e.g., register, login, transfer funds).
* **Expected Results**: End-users should be able to perform their tasks without issues, and the application should be intuitive and user-friendly.