PayHab Test Plan Report

Team Name: Dream Team
Project Title: PayHab

Table of Contents

---

1. Scope

Product Overview:
PayHab is a digital solution designed to help Habib University students track credit-based transactions with on-campus vendors. The system enables users to register, log in securely, view a dashboard that aggregates outstanding dues with various vendors, record new loan entries, mark payments, and receive notifications for pending dues. The app leverages real-time data synchronization through Firebase and communicates with a Flask backend via RESTful APIs.

Areas to be Tested:

- **User Management:** Registration, authentication (login), and profile retrieval.
- **Loan & Transaction Management:** Loan entry, tracking outstanding dues, and clearing loans taken.
- **Dashboard Functionality:** Aggregation and display of user data and vendor loan summaries.
- **Notifications:** Sending and receiving reminders for pending dues.
- **API Communication:** Ensuring proper RESTful interaction between the app and the backend.
- **Real-Time Data Sync:** Verification of immediate updates across users and vendors using Firebase.

2. Test Approach

Testing Processes:

- **Test Levels:**
  - **Unit Testing:** Verify individual modules (e.g., API endpoints, UI components).
  - **Integration Testing:** Confirm communication between the frontend (Flutter) and backend (Flask & Firebase).
  - **System Testing:** End-to-end testing of user flows (registration, loan entry, payments, notifications).
  - **Performance Testing:** Evaluate response times, especially for real-time data sync and dashboard aggregation.
  - **Usability Testing:** Verify intuitive UI/UX for smooth navigation.

Roles and Responsibilities:

- **Project Manager:** Sameer Kamani
- **Developers:** Aamaina Mukarram Tahir Ali ,Ahmad Hamayun Hanif, Saif Nazir ,Sameer Kamani.
- **Testers:** Aamaina Mukarram tahir Ali ,Ahmad Hamayun Hanif, Saif Nazir ,Sameer Kamani

Types of Testing:

- Manual testing for exploratory and user acceptance tests.

Bug Tracking:

- All defects will be tracked using either Jira or GitHub Issues, based on the team's preference, and prioritized by severity and impact.

3. Test Environment

Platforms and Devices:

- **Platforms:** Android (Emulator), iOS(Emulator)
- **Backend:** Flask (Python) API hosted on Firebase
- **Database:** Firebase Firestore
- **Testing Tools:** Postman for API testing, Firebase Emulator for local testing

4. Release Control

Versioning:

- Semantic versioning will be used (e.g., v1.0.0 for the initial release).

Deployment Frequency:

- Releases will be scheduled weekly for major updates.
- A clear rollback plan (using Git for version control) is established in case of critical issues in the live environment.

5. Risk Analysis

Identified Risks and Mitigation Plans:

User Registration Failures:

- **Risk:** Users may not be able to register correctly due to input errors or unexpected system behavior.
- **Mitigation:** Manually test registration with both valid and invalid data to ensure proper error messages and account creation.

Login Issues:

- **Risk:** Registered users might face difficulties logging in (e.g., incorrect error handling for wrong credentials).
- **Mitigation:** Test login scenarios thoroughly with correct and incorrect credentials to verify that access is granted only when appropriate.

Loan Entry Errors:

- **Risk:** Loan transactions might not be recorded accurately or may display incorrect status (e.g., "outstanding" not updating).
- **Mitigation:** Manually enter a variety of loan transactions and verify that the dashboard displays the correct details.

Payment Processing Problems:

- **Risk:** Payments may not update the loan status correctly, causing confusion in outstanding balances.
- **Mitigation:** Simulate payments for existing loans and ensure that the system updates the status (e.g., from "outstanding" to "paid" or "partial") as expected.

Notification Failures:

- **Risk:** Users may not receive notifications when dues are pending, leading to missed reminders.
- **Mitigation:** Manually trigger scenarios where notifications should be sent and confirm that the correct message appears on the test device.

API Communication Issues:

- **Risk:** Inconsistent data may result from communication issues between the Flutter app and the Flask backend.
- **Mitigation:** Manually test API endpoints using tools such as Postman, simulate network issues, and verify that error handling and fallback mechanisms work correctly.

Environment Differences:

- **Risk:** The application may behave inconsistently across different platforms (Android vs. iOS).
- **Mitigation:** Conduct thorough cross-platform testing using both emulators and real devices to ensure consistent functionality and user experience.

---

6. Test Plan (Test Cases)

Below are sample test cases covering the critical functionalities of PayHab:

Test Case TC001: User Registration Test

- **Test Case ID:** TC001
- **Test Case Name:** Validate New User Registration
- **Test Case Objective:** To ensure a new student can register successfully using a valid student ID, password, name, and email.
- **Prerequisites:**
  - Test environment setup with backend and Firebase ready.
  - Valid test data (unique student ID, valid email format).
- **Test Steps:**
  - Navigate to the registration page on the mobile app.
  - Enter valid student ID, password, name, and email.
  - Click on the "Sign Up" button.
- **Expected Results:**
  - The system creates a new account and displays a confirmation message.
  - The user is redirected to the login page, or automatically logged in if applicable.

Test Case TC002: User Login Test

- **Test Case ID:** TC002
- **Test Case Name:** Validate User Login Functionality
- **Test Case Objective:** To verify that a registered user can log in with the correct credentials.
- **Prerequisites:**
  - A registered user account exists in the system.
- **Test Steps:**
  - Navigate to the login screen.
  - Enter valid student ID and password.
  - Click on the "Login" button.
- **Expected Results:**
  - User is successfully authenticated, and an access token is received.
  - The dashboard screen loads with user details and outstanding dues.

Test Case TC003: Dashboard Functionality Test

- **Test Case ID:** TC003
- **Test Case Name:** Verify Dashboard Data Aggregation
- **Test Case Objective:** To ensure the dashboard correctly aggregates and displays user information and vendor loan summaries.
- **Prerequisites:**
  - User is logged in.
  - Existing loan entries in the system.
- **Test Steps:**
  - Log in as a registered user.
  - Navigate to the dashboard.
- **Expected Results:**
  - The dashboard displays accurate user profile details(his pending loans) .
  - Vendor list shows the correct total outstanding amounts and associated loan details.

Test Case TC004: Loan Entry & Tracking Test

- **Test Case ID:** TC004
- **Test Case Name:** Record and Track a New Loan
- **Test Case Objective:** To verify that a new loan entry is recorded and tracked properly in the system.
- **Prerequisites:**
  - User must be logged in.
  - A list of vendors is available (from a hard-coded set).
- **Test Steps:**
  - Navigate to the "Loan Entry" screen.
  - Select a vendor and enter a valid loan amount along with the current date and an optional description.
  - Submit the loan entry.
- **Expected Results:**
  - The new loan is recorded and appears in the dashboard under the selected vendor.
  - Loan status is set as "outstanding."

Test Case TC005: Loan Payment Test

- **Test Case ID:** TC005
- **Test Case Name:** Process Loan Payment
- **Test Case Objective:** To ensure that processing a payment updates the loan status appropriately.
- **Prerequisites:**
  - A loan entry exists with the status "outstanding."
- **Test Steps:**
  - Select an outstanding loan from the dashboard.
  - Initiate the payment process by entering a valid payment amount and date.
  - Confirm the payment submission.
- **Expected Results:**
  - The system updates the loan status to "paid" or "partial" based on the payment amount.
  - The dashboard reflects the updated loan status.

Test Case TC006: Notification Test

- **Test Case ID:** TC006
- **Test Case Name:** Verify Notification for Pending Dues
- **Test Case Objective:** To check that users receive notifications for pending dues.
- **Prerequisites:**
  - A user with at least one outstanding loan.
- **Test Steps:**
  - Trigger a condition where a notification should be sent (e.g., overdue loan).
  - Monitor the user’s device for a push/SMS notification.
- **Expected Results:**
  - A notification is sent and received with the correct title and message prompting the user to clear dues.
