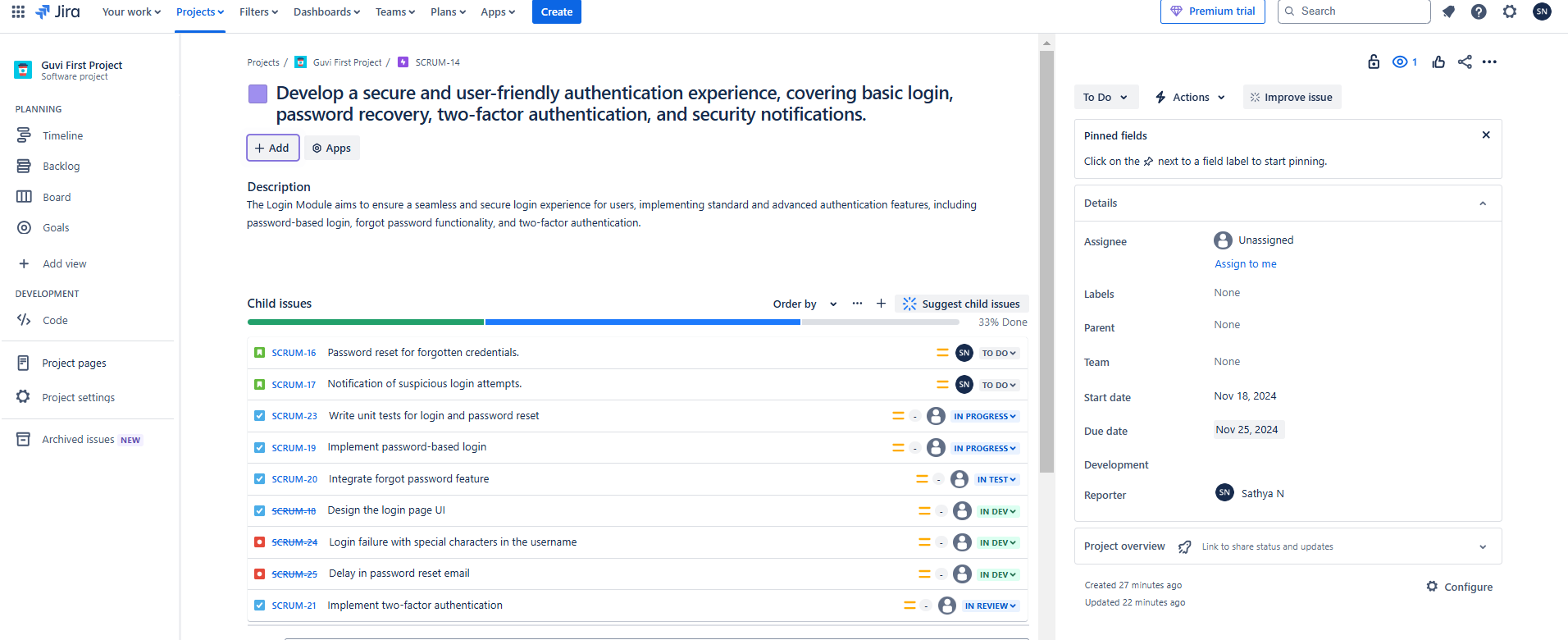
**Development of Netbanking App**

(Project Management Tool Jira)

### **Epic 1: Login Module**

**Description**: The Login Module is responsible for providing a secure and seamless authentication experience for users. It includes functionalities such as password-based login, forgot password, and two-factor authentication.



#### 

#### 

#### 

#### 

#### 

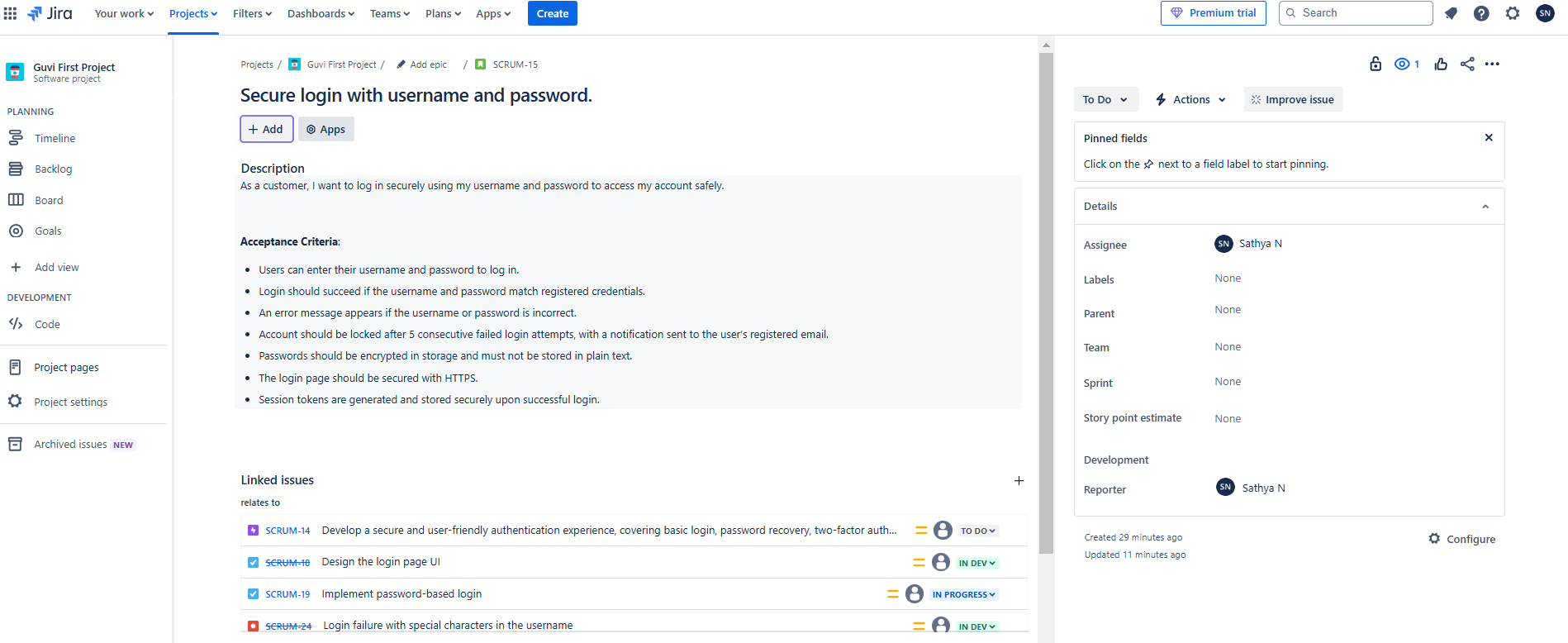
#### 

#### **User Stories for Login Module**

**User Story 1:** As a customer, I want to log in securely using my username and password.

**Acceptance Criteria:**

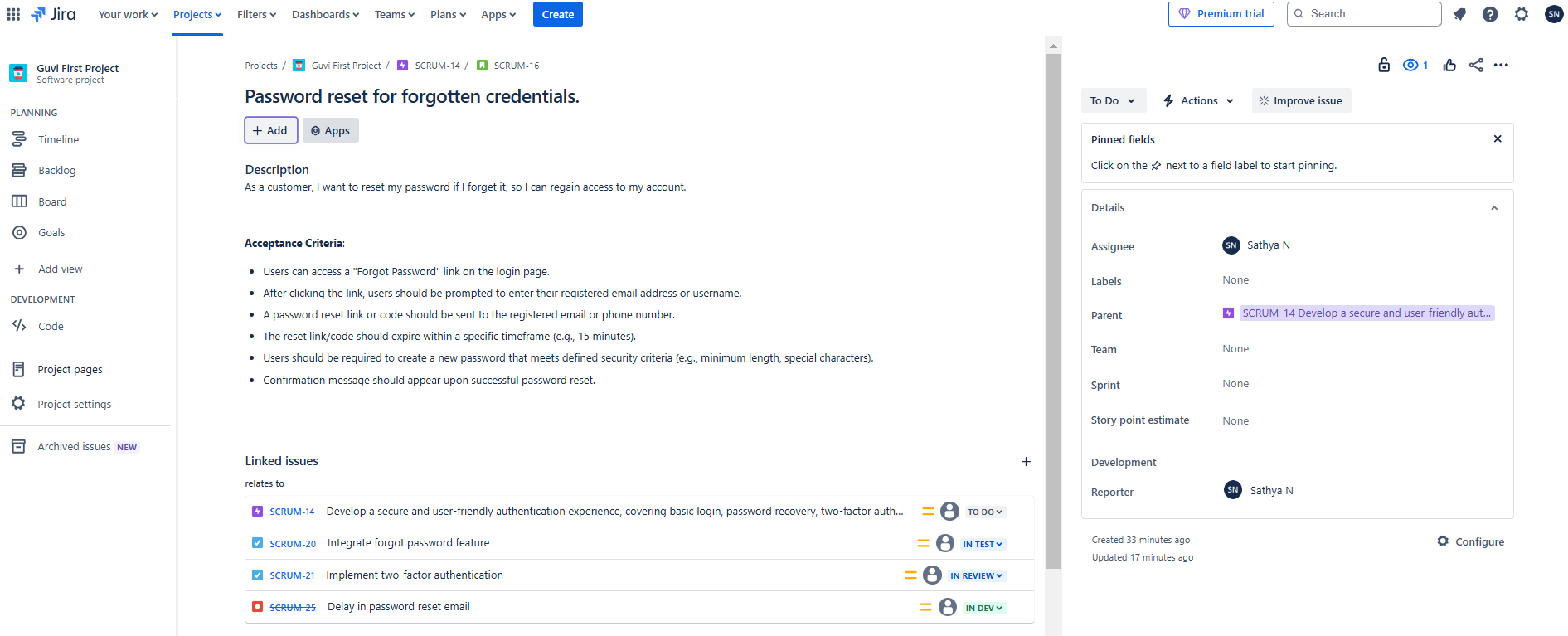
* Users can enter their username and password to log in.
* Login should succeed if the username and password match registered credentials.
* An error message appears if the username or password is incorrect.
* Account should be locked after 5 consecutive failed login attempts, with a notification sent to the user’s registered email.
* Passwords should be encrypted in storage and must not be stored in plain text.
* The login page should be secured with HTTPS.
* Session tokens are generated and stored securely upon successful login.



**User Story 2:** As a customer, I want to reset my password if I forget it.

**Acceptance Criteria:**

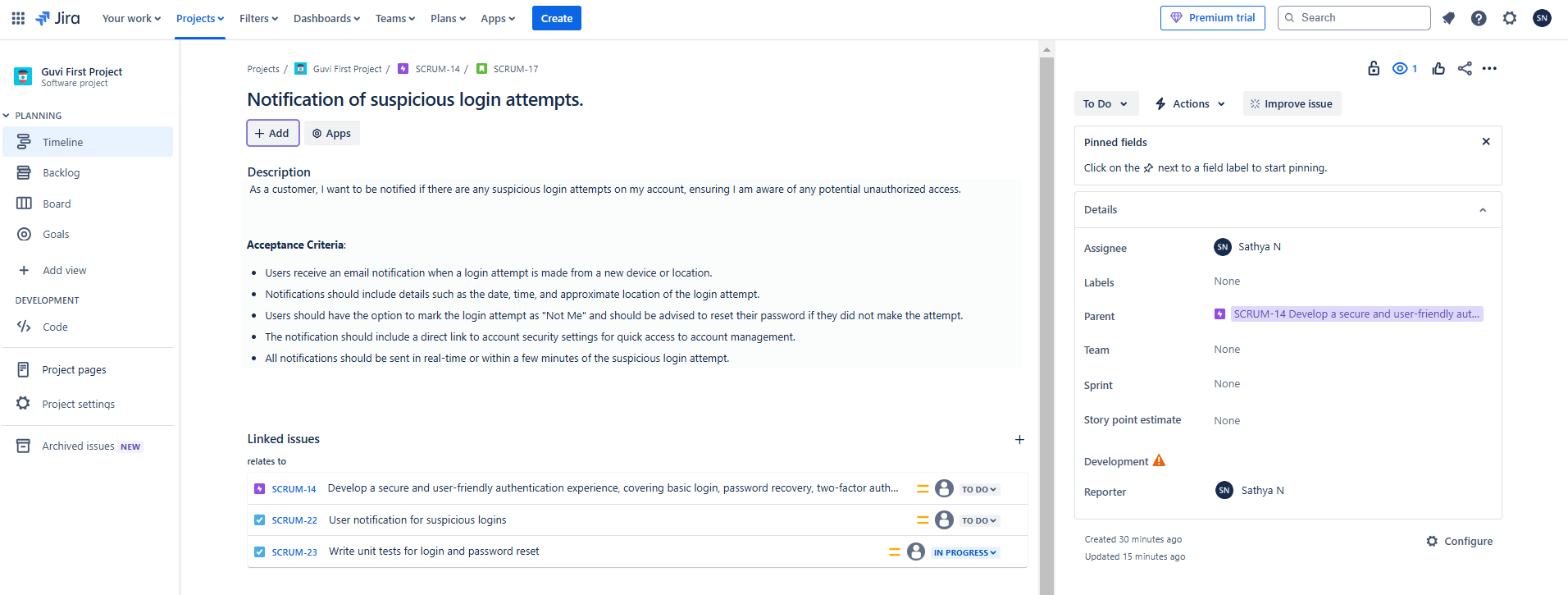
* Users can access a "Forgot Password" link on the login page.
* After clicking the link, users should be prompted to enter their registered email address or username.
* A password reset link or code should be sent to the registered email or phone number.
* The reset link/code should expire within a specific timeframe (e.g., 15 minutes).
* Users should be required to create a new password that meets defined security criteria (e.g., minimum length, special characters).
* Confirmation message should appear upon successful password reset.



**User Story 3:** As a customer, I want to be notified of suspicious login attempts to secure my account.

**Acceptance Criteria:**

* Users receive an email notification when a login attempt is made from a new device or location.
* Notifications should include details such as the date, time, and approximate location of the login attempt.
* Users should have the option to mark the login attempt as "Not Me" and should be advised to reset their password if they did not make the attempt.
* The notification should include a direct link to account security settings for quick access to account management.
* All notifications should be sent in real-time or within a few minutes of the suspicious login attempt.



#### 

#### 

#### 

#### **Tasks for Login Module**

### **Tasks for Login Module**

1. **Task 1**:
   * **Summary**: Design the login page UI.
   * **Description**: Create a user-friendly and secure interface for the login page, ensuring it meets design and usability standards.
2. **Task 2**:
   * **Summary**: Implement password-based login.
   * **Description**: Develop the backend functionality that enables users to log in using their username and password, ensuring data validation and security.
3. **Task 3**:
   * **Summary**: Integrate forgot password feature.
   * **Description**: Add the forgot password feature, allowing users to reset their password via email or SMS if they have forgotten it.
4. **Task 4**:
   * **Summary**: Implement two-factor authentication.
   * **Description**: Set up an additional layer of security requiring users to enter a code (sent to their phone or email) after entering their password.
5. **Task 5**:
   * **Summary**: User notification for suspicious logins.
   * **Description**: Create notifications to alert users of suspicious login attempts, enabling them to secure their accounts if needed.
6. **Task 6**:
   * **Summary**: Write unit tests for login and password reset.
   * **Description**: Develop tests to verify the functionality and reliability of the login and password reset features, ensuring the module performs as expected.



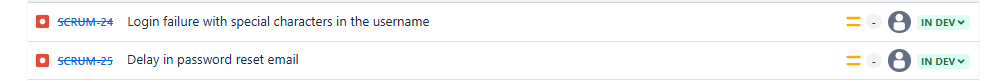
### **Bugs for Login Module**

**Bug 1**:

* + **Summary**: Login failure with special characters in the username.
  + **Description**: The login functionality currently fails when the username includes special characters, resulting in unsuccessful login attempts.

**Bug 2**:

* + **Summary**: Delay in password reset email.
  + **Description**: The password reset email is not delivered within the expected timeframe, causing delays in account recovery for users.



#### 

#### **Test Cases for Login Module**

* Test Case 1: Verify that a registered user can log in successfully with correct credentials.
* Test Case 2: Verify that an error message appears if the user enters incorrect credentials.
* Test Case 3: Verify that a password reset email is sent when a user selects the "Forgot Password" option.
* Test Case 4: Verify that the two-factor authentication screen appears after entering correct credentials.

### 

### 

### 

### **Epic 2: Account Management Module**

**Description**: This module enables customers to manage their accounts, view balances, check account details, and update profile information.

#### 

#### **User Stories for Account Management Module**

### **User Story 1:** As a customer, I want to view my account balance in real-time.

**Summary**: This user story focuses on providing customers with an up-to-date view of their account balance. Real-time updates are critical for a good user experience, especially for customers making frequent transactions.

**Description**: The account balance should always reflect the most recent transactions, ensuring that customers are fully aware of their available funds. The system should push updates as soon as there is any change to the account, eliminating the need for manual refreshes or delayed balance updates.

* **Acceptance Criteria**:
  + The account balance should be displayed prominently on the account overview page.
  + The displayed balance should reflect real-time updates (changes due to transactions, deposits, etc.).
  + The page should auto-refresh or update the balance without the need for a manual page reload.
  + The balance should be clearly labeled with a currency symbol and accurate value.

### **User Story 2:** As a customer, I want to view and edit my profile details.

**Summary:** This user story allows customers to manage their personal information by viewing and editing their profile details, ensuring that they can keep their information up to date.

**Description:** Customers should have the ability to edit and save changes to their personal information, ensuring that contact details or preferences are current. Proper validation needs to be in place to avoid errors like invalid email addresses or incomplete fields.

* **Acceptance Criteria**:
  + The profile page should display existing user information (name, email, phone number, etc.).
  + Customers should be able to edit their details and save the changes.
  + Profile edits should be saved successfully and reflect in the account immediately after submission.
  + Input validation should be implemented for all fields (e.g., correct email format, phone number length).
  + A confirmation message should be shown after successful profile update.

**User Story 3:** As a customer, I want to see a summary of my accounts.

**Summary**: This user story ensures that customers can quickly get an overview of all their accounts in one place, which helps them track and manage their finances.

**Description**: A consolidated view of all accounts should be provided, with essential information for each account displayed clearly. This will help users easily navigate and access further account-specific details with minimal effort.

* **Acceptance Criteria**:
  + The summary should display a list of all accounts linked to the customer (e.g., checking, savings, etc.).
  + Each account should display key information, such as the account type, balance, and recent activity.
  + The summary should be easy to navigate and view without requiring additional clicks.
  + The customer should be able to access details of each account from the summary page.

#### 

#### **Tasks for Account Management Module**

#### 

### **Task 1:** Design account overview page.

**Summary**: Designing the account overview page is the foundational step in presenting users with a clear and intuitive dashboard where they can view their account balance, profile, and summary.

**Description**: This task involves wireframing and creating UI mockups that will represent the account management information. The design should ensure clarity, accessibility, and easy navigation between different account sections.

### **Task 2:** Implement real-time balance display.

**Summary**: This task involves implementing the logic to fetch and display the account balance in real-time, ensuring that the user sees up-to-date information.

**Description**: Develop the functionality that retrieves the current balance from the backend and updates the frontend in real-time. This could include integrating with APIs that push updates or using WebSocket connections for live data.

### **Task 3:** Implement profile editing functionality.

**Summary**: This task focuses on providing the ability for customers to modify their profile information (like name, contact info, etc.) and save those changes in the system.

**Description**: The functionality will include form validation, error handling, and a mechanism to update the user profile data in the database when changes are made. It should ensure that changes are reflected immediately.

### **Task 4:** Create account summary view.

**Summary**: This task involves designing and implementing a view that displays an overview of all accounts linked to the customer, providing the essential details for each.

**Description**: Implement a dashboard-like view that lists the customer’s accounts with essential information such as balance, account type, and recent transactions. It should also support navigation to individual account details.

### **Task 5:** Write unit tests for account summary and profile update functionalities.

**Summary**: This task involves writing automated tests to ensure that both the account summary and profile update functionalities work as expected.

**Description**: Unit tests should be created for each critical function, such as retrieving account data, updating the profile, and handling error scenarios. These tests will ensure the reliability and correctness of the functionalities.

### 

### **Bugs for Account Management Module**

### 

### **Bug 1:** Profile changes do not reflect immediately on the account summary page.

**Summary:** This bug occurs when changes made to the user profile do not immediately appear on the account summary page.

**Description:** After a user successfully updates their profile, the changes should be visible immediately across the system, including the account summary. This issue requires debugging the profile update mechanism and ensuring that the UI refreshes properly.

### **Bug 2:** Account balance display does not update on refresh.

**Summary:** The account balance display fails to automatically update even after a user refreshes the page or after a transaction.

**Description:** The system should be able to fetch and display the latest account balance after a refresh. This issue could be related to the real-time data update logic or caching mechanism, which needs to be resolved for accurate display.

#### **Test Cases for Account Management Module**

**Test Case 1**: Verify that the account balance is displayed accurately and updates in real-time.

**Test Case 2**: Verify that users can update their profile information successfully.

**Test Case 3**: Verify that the account summary shows all accounts linked to the user.

### 

### 

### **Epic 3: Transaction Module**

**Description**: The **Transaction Module** aims to provide customers with a secure and user-friendly way to handle their financial transactions, including transferring funds, scheduling future transactions, and receiving confirmations for successful transactions. The module will allow customers to initiate transfers to other accounts, plan ahead for scheduled transactions, and ensure they receive confirmation notifications once their transactions are processed.

#### **User Stories for Transaction Module**

#### **User Story 1:** As a customer,I want to transfer funds to another account securely and easily,So that I can manage my finances effectively.

**Acceptance Criteria:**

* The system should authenticate the user before initiating any transfer (via login or multi-factor authentication).
* The user can enter the recipient's account details (e.g., account number, routing number) and transfer amount.
* The system should validate the recipient’s account details before confirming the transaction.
* The user can review the details before confirming the transaction.
* The system should display a success or failure message with transaction details.
* The transfer should be completed securely with encryption to protect sensitive data.

#### 

#### 

#### **User Story 2:** As a customer,I want to schedule future transactions, So that I can automate payments and avoid missing deadlines.

#### 

#### **Acceptance Criteria:**

* The user should be able to select a date and time for a future transaction.
* The user can set up recurring transactions with options for frequency (e.g., weekly, monthly).
* The system should notify the user of the scheduled transaction with a reminder.
* The scheduled transaction will be automatically processed on the specified date and time.
* The system should allow the user to edit or cancel scheduled transactions before execution.
* The transaction is processed according to the details set by the user (amount, recipient, etc.).

**User Story 3:**As a customer,I want to receive confirmation after a successful transaction,So that I have proof of the completed transaction for my records.

**Acceptance Criteria:**

1. The user should receive an immediate confirmation notification once the transaction is successfully processed.
2. The confirmation should include the transaction ID, recipient details, amount transferred, and date/time.
3. The confirmation should be sent via email, SMS, or in-app notification, as per user preference.
4. The system should log the transaction for the user’s reference in their transaction history.
5. The confirmation should be clear, accurate, and easy to understand.
6. If the transaction fails, the system should notify the user with the reason for failure.

#### **Tasks for Transaction Module**

**Task 1:** Develop Fund Transfer Functionality

**Description**: Implement the core functionality that enables customers to transfer funds to other accounts securely. This involves ensuring proper account validation, amount entry, transaction processing, and successful execution.

**Objective**: To allow users to initiate and complete transactions to transfer money easily and securely.

**Task 2:** Implement Scheduling Feature for Future Transactions

**Description**: Build a feature that allows customers to schedule transactions for future dates, either once-off or recurring. This includes setting up user inputs for date, time, and frequency, as well as logic to execute the scheduled transactions automatically.

**Objective**: To provide customers the ability to automate transactions for future execution, ensuring flexibility in payment management.

**Task 3:** Design Transaction Confirmation Message

**Description**: Create a standardized confirmation message that will be displayed after a transaction is completed successfully. This message should clearly outline transaction details (e.g., transaction ID, recipient, amount, and time).

**Objective**: To provide customers with confirmation of successful transactions for their records and peace of mind.

**Task 4:** Create Notification for Successful Transactions

**Description**: Develop a notification system that alerts the customer when a transaction has been successfully completed. This could be via email, SMS, or app notification.

**Objective**: To ensure customers are notified of successful transactions and can verify the completion of their transfers.

**Task 5:** Write Unit Tests for Transaction and Scheduling Functionalities

**Description**: Write and execute unit tests to verify the correctness and reliability of the transaction processing and scheduling features. This ensures that edge cases and failure scenarios are handled properly.

**Objective**: To ensure robust functionality of the transaction module through thorough testing, reducing the risk of defects in production.

**Bugs for Transaction Module**

**Bug 1:** Scheduled Transactions Do Not Execute at the Designated Time

**Description**: Scheduled transactions are failing to execute at the designated time, potentially due to issues in the scheduling mechanism or timing logic.

**Impact**: Customers may miss important scheduled payments, leading to missed deadlines or penalties.

**Bug 2:** Transaction Confirmation Is Not Received for Cross-Bank Transfers

**Description**: Users are not receiving confirmation notifications after completing cross-bank transfers, likely due to integration or system communication issues.

**Impact**: Lack of confirmation may leave customers uncertain about the success of their transactions, which could lead to confusion or customer dissatisfaction.

#### 

#### **Test Cases for Transaction Module**

* Test Case 1: Verify that the user can successfully transfer funds to another account.
* Test Case 2: Verify that the scheduling feature accurately sets transactions for future dates.
* Test Case 3: Verify that a transaction confirmation message is received after a successful transaction.

**Epic 4: Payment Module**

**Description**: This module provides bill payment functionality, allowing users to pay for utilities, credit cards, and other services directly through the app.

#### 

#### **User Stories for Payment Module**

#### **User Story 1:** As a customer, I want to pay my utility bills through the app.

**Acceptance Criteria:**

* The app should allow customers to select a utility biller from a list or search for one.
* The customer should be able to enter the payment amount and confirm payment.
* The payment should be processed securely via an integrated payment gateway.
* A success or failure message should be displayed to the customer once the payment is complete.
* A receipt or confirmation of the payment should be sent to the customer’s email or displayed within the app.
* The system should allow payment for multiple bill types (e.g., electricity, water, gas, etc.).

#### **User Story 2:** As a customer, I want to save frequently used billers for easy access.

**Acceptance Criteria:**

* Customers should be able to mark a biller as "frequent" after making a payment or manually adding it.
* Saved billers should appear in a designated section called "Frequent Billers" on the bill payment screen.
* The app should display saved billers in a list that allows customers to select and pay with a single click.
* Customers should have the option to remove or edit saved billers.
* The app should save biller information securely, ensuring it is accessible only to the user who saved it.

#### **User Story 3:** As a customer, I want to view a history of my payments.

**Acceptance Criteria:**

* A "Payment History" section should be available in the app where customers can view all past payments.
* The history should display transaction details including the biller, payment date, amount, and status (success/failure).
* The payment history should be updated in real time after each successful transaction.
* Customers should have the option to filter or search payment history by date, biller, or amount.
* The payment history should display both successful and failed payment attempts.

#### **Tasks for Payment Module**

**Task 1:** Implement Bill Payment Functionality

**Summary**: Develop and integrate the functionality that allows users to pay their utility bills directly through the app.

**Description**: This task involves designing and coding the payment process, which includes selecting a biller, entering payment details, and securely processing the payment through a payment gateway. The task also includes displaying success or failure messages, and optionally sending confirmation emails or receipts.

**Task 2:** Develop Feature to Save Frequently Used Billers

**Summary**: Create a feature that allows users to save frequently used billers for quick and easy access in future payments.

**Description**: This task includes implementing a system where users can save their billers after a transaction or manually mark a biller as a favorite. The saved billers will appear in a dedicated section, and users can easily select them for future payments without re-entering the biller’s information. Additionally, the task involves managing functionality to remove or edit saved billers.

**Task 3:** Create Payment History View

**Summary**: Build a feature that enables users to view their payment history, providing transparency and easy access to previous transaction details.

**Description**: This task focuses on creating a payment history section where users can view past transactions, including biller name, amount, payment status, and transaction date. It will also include features to filter and search through the history, as well as ensure that it updates in real-time after each payment is made.

**Task 4:** Write Unit Tests for Bill Payment and Biller Saving Functionalities

**Summary**: Write unit tests to ensure the accuracy and functionality of the bill payment and biller saving features.

**Description**: This task involves creating unit tests to verify that the bill payment and biller saving functionalities work as intended. Tests will cover scenarios like valid/invalid payments, correct biller saving and retrieval, and edge cases for both features. The goal is to ensure that any potential issues are identified and fixed early through automated testing.

#### **Bugs for Payment Module**

**Bug 1:** Saved Billers Are Not Displaying in the Bill Payment Section

**Summary**: There is an issue where saved billers are not appearing in the bill payment section, preventing users from quickly selecting them for payments.

**Description**: This bug occurs when users save a biller for future use, but the saved biller does not show up in the bill payment interface as expected. The issue may be related to data syncing, display logic, or backend integration. Fixing this bug will ensure that the list of saved billers is properly retrieved and displayed for user convenience.

**Bug 2:** Payment History Does Not Update with Recent Transactions

**Summary**: The payment history section is not reflecting the most recent transactions, causing users to see outdated information.

**Description**: This bug occurs when a user makes a payment, but the payment history view does not update with the new transaction details (such as amount, date, and status). The issue could be related to how the transaction data is being recorded, stored, or retrieved. Addressing this bug will ensure that the payment history is always up-to-date with the latest transaction information.

#### **Test Cases for Payment Module**

* Test Case 1: Verify that users can successfully pay a bill through the app.
* Test Case 2: Verify that users can save billers and see them in the payment section.
* Test Case 3: Verify that payment history displays all recent transactions accurately.

**Epic 5: Report Module**

**Description**: This module allows users to generate and view transaction history, account statements, and other reports.

#### **User Stories for Report Module**

### **User Story 1:** View Transaction History for the Last 6 Months

**Summary:**As a customer, I want to view my transaction history for the last 6 months so that I can keep track of my financial activities over this period.

**Description:**This feature will allow customers to access a detailed view of all their transactions that occurred in the last six months. The transaction history will include the date, type, amount, and status of each transaction. The interface should present this data in an easy-to-read format with the ability to scroll through and navigate.

**Acceptance Criteria:**

* The system must display the transaction history for the last 6 months when the user navigates to the transaction history page.
* The list should include transaction details like date, type (e.g., debit, credit), amount, and transaction status (e.g., completed, pending).
* The system should only show transactions from the past 6 months and hide older transactions.
* The data should load without errors and display clearly formatted details for each transaction.
* A "Load More" option should be available to load older transactions within the 6-month window, if applicable.

### 

### **User Story 2:** Download Account Statement as a PDF

**Summary:**As a customer, I want to download my account statement as a PDF so that I can have a portable, printable record of my transactions.

**Description:**Customers will have the option to download their transaction history or account statement in PDF format for a specified time range. This feature should allow customers to select the time frame (e.g., monthly, quarterly) and provide an easy option to generate and download the document.

**Acceptance Criteria:**

* The system should provide a “Download PDF” button on the transaction history page.
* The user should be able to specify the date range (e.g., last 30 days, custom range, or last 6 months) for the account statement.
* The downloaded PDF must include the transaction details (e.g., date, transaction type, amount, status) in a clear, organized format.
* The PDF download should be quick and error-free, ensuring that all requested data is included.
* The PDF should be formatted to be easy to read and suitable for printing.

**User Story 3: Filter Transactions by Date, Type, and Amount**

**Summary:**As a customer, I want to filter my transactions by date, type, and amount so that I can easily find specific transactions that meet my criteria.

**Description:**This feature allows customers to apply filters to their transaction history to narrow down the list based on specific criteria: date range, transaction type (e.g., credit, debit), and transaction amount. This will help customers quickly locate and review relevant transactions.

**Acceptance Criteria:**

* The user should be able to filter transactions by date range (e.g., specific dates, last 30 days).
* The user should be able to filter by transaction type (e.g., credit, debit).
* The user should be able to filter by transaction amount (e.g., greater than, less than, or equal to a specified amount).
* The filters should be easily accessible and clearly labeled on the transaction history page.
* The system should display only the transactions that meet the filter criteria, and it should update the transaction list dynamically after applying the filter.
* The filters should be resettable to show all transactions when the user clears the applied filters.

#### **Tasks for Report Module**

### **Task 1:** Create Transaction History View

**Summary:**Develop a user interface to display a customer’s transaction history.

**Description:**This task involves creating a front-end view where customers can see a list of their transactions for the past six months. The transaction history view should present each transaction with essential details like date, transaction type (e.g., credit or debit), amount, and status. Additionally, the view should be user-friendly, allowing customers to navigate and load more transactions as needed. This view will also support other features like filtering and PDF download.

**Task 2:** Develop PDF Download Feature for Account Statements

**Summary:**Build a feature allowing customers to download their account statements in PDF format.

**Description:**The task includes creating the backend and frontend components necessary for customers to download their transaction history as a PDF. Customers should be able to specify a date range for the account statement, and the system should generate a well-formatted PDF document containing all transaction details within the selected period. This functionality should handle various time frames, ensuring accuracy and readability.

**Task 3:** Implement Transaction Filter Functionality

**Summary:**Enable customers to filter their transaction history by date, type, and amount.

**Description:**This task involves developing a filtering functionality within the transaction history view. Customers will have the ability to apply filters based on date range, transaction type, and transaction amount. This feature will allow customers to narrow down their transaction history to locate specific records easily. The filters should dynamically update the transaction list and be designed for ease of use with a clear interface.

**Task 4:** Write Unit Tests for Transaction History and Filter Features

**Summary:**Write unit tests to verify the functionality of the transaction history view and filter features.

**Description:**This task involves creating automated tests to ensure that the transaction history view and filtering functionality work as expected. The tests should cover scenarios for displaying transactions, handling various filter criteria, and verifying that the correct data is shown based on the applied filters. These tests will help catch bugs early and confirm that the features work reliably under different conditions.

#### **Bugs for Report Module**

### **Bug 1:** PDF Download Feature Generates an Empty File

**Summary:**The PDF download feature creates an empty file instead of a populated account statement.

**Description:**This bug occurs when a customer attempts to download their account statement as a PDF. Instead of receiving a populated document, the file generated is empty, lacking the expected transaction data. This issue could be related to data retrieval, formatting, or the PDF generation process and requires investigation to ensure that the document includes the correct information for the specified date range.

**Bug 2:** Filters Do Not Apply Correctly When Multiple Criteria Are Selected

**Summary:**Filters do not work correctly when multiple filtering criteria are selected simultaneously.

**Description:**This bug affects the transaction history filter functionality. When customers apply more than one filter criterion (e.g., date range and transaction type), the system fails to filter transactions accurately, often displaying an incorrect or unfiltered list. This issue likely stems from the backend query logic or the filter component itself and needs to be resolved to ensure the correct filtering behavior for multiple criteria.

#### **Test Cases for Report Module**

* Test Case 1: Verify that users can view transaction history for the last 6 months.
* Test Case 2: Verify that users can download account statements as PDFs.
* Test Case 3: Verify that the filter applies correctly to transaction history.

### **Sprint Planning and Execution**

#### **Sprint 1:** Login Module and Account Management Module

Focus: Implement and test Login Module and Account Management Module functionality.

#### **Sprint 2:** Transaction Module

Focus: Develop and test the fund transfer and scheduling functionalities in the Transaction Module.

#### **Sprint 3:** Payment Module and Report Module

Focus: Complete the Payment Module and Report Module, conduct end-to-end testing, and integrate all functionalities.