### ***UseCases:***

#### **Use Case 1: Sending Money to Friends/Family**

**Goal**: To allow users to send money to their friends or family quickly and securely.

**Actors**: User (Customer), Payment Gateway (Stripe/PayPal), Recipient (Friend/Family)

##### **Preconditions:**

* The user is logged into their account.
* The user has a verified payment method linked to their account.
* The recipient has a valid email/phone number registered with the app (or can be invited to join).

##### **Basic Flow:**

1. **User opens the app and navigates to the "Send Money" page**:
   1. The user selects the **Send Money** option from the home page.
2. **User enters the recipient’s details**:
   1. The user selects a contact from their phone’s contact list, or they can manually input the recipient’s **email** or **phone number**.
3. **User specifies the payment amount**:
   1. The user enters the amount of money they wish to send.
4. **User selects the payment method**:
   1. The user selects one of their linked payment methods (e.g., credit card, bank account).
5. **User reviews transaction details**:
   1. The app displays a summary of the transaction, including recipient details, amount, and payment method.
6. **User confirms the transaction**:
   1. The user clicks the **Confirm** button to complete the payment.
7. **Payment is processed**:
   1. The payment gateway processes the transaction, and the funds are transferred to the recipient.
8. **User receives confirmation**:
   1. The user is notified that the payment was successful.
   2. The recipient receives a notification about the incoming payment.

##### **Alternative Flow:**

1. **Recipient not found**:
   1. If the recipient is not registered with the app, the system prompts the user to send an invitation to the recipient to join the platform.

##### **Postconditions:**

* The money is transferred to the recipient, and both the sender and the recipient receive notifications.
* The transaction details are logged in the user’s transaction history.

#### **Use Case 2: Admin Panel for Transaction Management**

**Goal**: To allow administrators to view, manage, and resolve issues related to user transactions, including refunds, disputes, and fraud detection.

**Actors**: Admin (System Admin), User (Customer), Support Team

##### **Preconditions:**

* The admin is logged into the **Admin Panel**.
* The admin has appropriate permissions to manage transactions and resolve disputes.

##### **Basic Flow:**

1. **Admin logs into the Admin Panel**:
   1. The admin uses their credentials to access the back-end system.
2. **Admin views transaction list**:
   1. The admin is presented with a list of recent transactions from all users, including transaction ID, user, amount, recipient, and status.
3. **Admin filters transactions**:
   1. The admin can filter transactions by date, user, status (pending, completed, disputed), etc.
4. **Admin reviews disputed transactions**:
   1. If a transaction is flagged or disputed by a user, the admin reviews the details of the dispute, including the reason provided by the user.
5. **Admin resolves disputes**:
   1. Based on the investigation, the admin can:
      1. Approve a refund to the user.
      2. Approve or deny the dispute.
      3. Suspend the user’s account if fraudulent activity is detected.
6. **Admin processes refunds**:
   1. If a refund is required, the admin processes the refund using the payment gateway.
   2. The system sends a refund confirmation to the user.
7. **Admin generates reports**:
   1. The admin can generate financial and transactional reports for auditing purposes.

##### **Alternative Flow:**

1. **Fraud detection**:
   1. If the system detects suspicious activity (e.g., multiple failed transactions, unusual payment amounts), the admin is alerted, and the transaction is flagged for review.

##### **Postconditions:**

* The admin successfully manages and resolves disputes, processes refunds, and flags fraudulent transactions.
* The app logs all actions performed by the admin for auditing purposes.

### **Use Case 3: Payment Setup**

**Goal**: To enable users to securely link a payment method (credit/debit card, bank account) to their account for future transactions.

**Actors**: User (Customer), Payment Gateway (Stripe/PayPal), Bank (for bank accounts)

##### **Preconditions:**

* The user is logged into the app.
* The user has access to valid payment information (e.g., credit card or bank account details).

##### **Basic Flow:**

1. **User navigates to the Payment Setup page**:
   1. The user clicks on the “Payment Methods” section in the app's menu to link a new payment method.
2. **User selects the payment method type**:
   1. The user chooses either **Credit/Debit Card** or **Bank Account** as the type of payment method they wish to link.
3. **User enters payment information**:
   1. For a **Credit/Debit Card**:
      1. The user enters their **card number**, **expiry date**, and **CVV**.
      2. The system securely stores the details using encryption.
   2. For a **Bank Account**:
      1. The user provides their **bank account number** and **routing number**.
      2. The system verifies these details by making a small test transfer.
4. **User verifies the payment method**:
   1. The system prompts the user to confirm the details entered for accuracy.
   2. If all details are correct, the user proceeds to save the information.
5. **User receives confirmation**:
   1. The system displays a confirmation that the payment method has been successfully linked.
   2. The linked payment method is now available for use in making transactions.
6. **Payment gateway verifies and processes the data**:
   1. The app securely sends the payment details to the payment gateway (e.g., Stripe or PayPal) for verification and storage.
   2. The gateway returns a success or failure message.

##### **Alternative Flow:**

1. **Invalid payment details**:
   1. If the card number is invalid, or the bank account information is incorrect, the system will display an error and prompt the user to re-enter their details.

##### **Postconditions:**

* The user's payment method is securely linked to their account, enabling them to make future payments without re-entering the information.
* The system ensures that the payment information is securely encrypted and complies with industry standards like PCI-DSS.

### **Use Case 4: Refund Request**

**Goal**: To allow users to request refunds for transactions that they believe were processed incorrectly or were unauthorized.

**Actors**: User (Customer), Admin (System Admin), Payment Gateway (Stripe/PayPal)

##### **Preconditions:**

* The user is logged into the app.
* The user has a completed transaction that is eligible for a refund.
* The transaction must meet specific criteria for eligibility (e.g., within 30 days, not a fraud attempt).

##### **Basic Flow:**

1. **User navigates to the Transaction History**:
   1. The user selects the “Transaction History” option from the app’s menu to view all past transactions.
2. **User selects the transaction to dispute**:
   1. The user identifies the transaction they believe is incorrect and selects it for more details.
3. **User clicks on "Request Refund"**:
   1. On the transaction details page, the user clicks the "Request Refund" button to initiate the refund process.
4. **User provides a reason for the refund**:
   1. The user must provide a reason for the refund request, such as:
      1. Duplicate payment.
      2. Product/service not received.
      3. Payment error.
5. **User submits refund request**:
   1. Once the user has entered the reason, they click the "Submit Request" button to send the refund request to the admin.
6. **Admin reviews the refund request**:
   1. The admin is notified of the refund request and logs into the admin panel to review the request.
   2. The admin checks the transaction history, verifies the payment method, and investigates the reason provided by the user.
7. **Admin processes the refund**:
   1. If the refund request is legitimate, the admin approves the refund and initiates the payment reversal through the payment gateway.
   2. The system automatically processes the refund and updates the transaction status.
8. **User receives confirmation**:
   1. The user receives a notification stating that the refund has been processed.
   2. The refund is credited to the original payment method.

##### **Alternative Flow:**

1. **Refund not eligible**:
   1. If the refund request does not meet the required conditions (e.g., outside of the refund window), the system notifies the user that the request cannot be processed.

##### **Postconditions:**

* The user receives the refund for the valid transaction.
* The refund is reflected in the user’s transaction history.
* The system logs all refund requests and actions taken by the admin for auditing purposes.

### **Use Case 5: Multi-currency Support**

**Goal**: To allow users to send money in different currencies while the system automatically converts the amount to CAD (Canadian Dollar).

**Actors**: User (Customer), Payment Gateway (Stripe/PayPal), Currency Exchange API (e.g., XE, Open Exchange Rates)

##### **Preconditions:**

* The user is logged into the app.
* The user has a linked payment method and sufficient balance for the transaction.
* The system has access to a currency exchange API for real-time rates.

##### **Basic Flow:**

1. **User initiates a money transfer**:
   1. The user clicks on the “Send Money” option in the app.
2. **User selects the recipient and payment amount**:
   1. The user selects the recipient, enters the amount, and chooses the currency they want to send (e.g., USD, EUR, GBP).
3. **System checks exchange rates**:
   1. The system queries the Currency Exchange API for the latest exchange rate from the selected currency to CAD.
4. **User reviews transaction details**:
   1. The system displays the conversion rate, the amount in the chosen currency, and the equivalent amount in CAD, including any transaction fees.
5. **User confirms the transaction**:
   1. The user clicks “Confirm” to approve the transaction, accepting the exchange rate and the fees.
6. **Payment is processed**:
   1. The payment gateway processes the transaction, converting the selected currency to CAD at the provided exchange rate.
   2. The system transfers the equivalent amount in CAD to the recipient’s account.
7. **User and recipient receive confirmation**:
   1. The user and the recipient are notified that the transaction has been completed successfully.

##### **Alternative Flow:**

1. **Error in currency conversion**:
   1. If the system fails to fetch the latest exchange rate, it notifies the user and requests them to try again later.

##### **Postconditions:**

* The system ensures that the transfer is successfully processed with the correct currency conversion to CAD.
* The transaction history reflects the original currency, exchange rate, and converted CAD amount.

### **Use Case 6: Transaction History**

**Goal**: To allow users to view and manage their past transactions.

**Actors**: User (Customer), Admin (for support), System (App)

##### **Preconditions:**

* The user is logged into the app.
* The system has a record of past transactions.

##### **Basic Flow:**

1. **User navigates to Transaction History**:
   1. The user clicks on the "Transaction History" section in the app’s menu.
2. **User views transaction list**:
   1. The app displays a list of past transactions, including the transaction ID, recipient, date, amount, and status.
3. **User filters transactions**:
   1. The user can filter transactions by date range, status (pending, completed, refunded), or transaction type (send, receive).
4. **User selects a transaction for details**:
   1. The user clicks on a specific transaction to view more detailed information, such as payment method, recipient’s info, and any associated notes.
5. **User exports transaction summary**:
   1. The user can export the transaction history into formats like PDF or CSV for financial tracking or record-keeping.

##### **Alternative Flow:**

1. **Error in displaying transaction history**:
   1. If there is an issue retrieving transactions, the system will notify the user and prompt them to try again later.

##### **Postconditions:**

* The system ensures that the transaction history is accurately displayed and can be filtered or exported for the user’s convenience.

### **Use Case 7: Add Funds to Account**

**Goal**: To allow users to add funds to their account from a linked payment method.

**Actors**: User (Customer), Payment Gateway (Stripe/PayPal), Bank (for bank transfers)

##### **Preconditions:**

* The user is logged in.
* The user has a linked payment method (credit/debit card, bank account).
* The user has sufficient balance to proceed with the transaction.

##### **Basic Flow:**

1. **User navigates to Add Funds**:
   1. The user clicks on the "Add Funds" button in the app.
2. **User selects the payment method**:
   1. The user selects the payment method (credit/debit card or bank account) from the available linked methods.
3. **User enters the amount to add**:
   1. The user enters the amount they wish to add to their account.
4. **System validates the entered amount**:
   1. The system checks if the entered amount is within the allowable limits (e.g., no minimum or maximum restrictions).
5. **User confirms the payment**:
   1. The user reviews the amount to be added and the associated fees (if any).
   2. The user clicks the "Confirm" button to proceed with the transaction.
6. **Payment gateway processes the transaction**:
   1. The payment gateway (e.g., Stripe or PayPal) processes the transaction and transfers the funds to the user's account.
7. **User receives confirmation**:
   1. The user receives a confirmation that the funds have been successfully added to their account, and their new balance is displayed.

##### **Alternative Flow:**

1. **Payment method validation failure**:
   1. If the payment method is invalid or the transaction fails, the system will notify the user and prompt them to try another method.

##### **Postconditions:**

* The user’s account balance is updated with the newly added funds.
* The transaction is recorded in the user’s transaction history.

### **Use Case 8: Transaction Notification**

**Goal**: To send notifications to users regarding their transactions.

**Actors**: User (Customer), Notification Service (Push Notification, Email Service)

##### **Preconditions:**

* The user is logged into the app.
* The user has provided a valid email address and enabled push notifications.

##### **Basic Flow:**

1. **Transaction is completed**:
   1. A transaction (payment, refund, fund addition, etc.) is completed on the user’s account.
2. **System triggers a notification**:
   1. The system sends a push notification and/or email to the user to inform them about the transaction's status.
3. **User receives the notification**:
   1. The user receives a real-time notification via their chosen method (e.g., app push notification, email).
4. **User opens notification for details**:
   1. The user clicks on the notification to view the transaction details, including amount, recipient, and status.

##### **Alternative Flow:**

1. **Email or push notification failure**:
   1. If the notification cannot be delivered (e.g., no internet connection), the system queues the notification for later delivery.

##### **Postconditions:**

* The user is kept informed about their transactions and any necessary updates.
* The system logs the notification delivery status.

### **Use Case 9: Transaction Cancellation**

**Goal**: To allow users to cancel an in-progress transaction before it is finalized.

**Actors**: User (Customer), Payment Gateway (Stripe/PayPal), Admin (Support)

##### **Preconditions:**

* The user is logged into the app.
* The transaction is still pending and has not yet been processed or completed.

##### **Basic Flow:**

1. **User navigates to Pending Transactions**:
   1. The user clicks on the “Pending Transactions” section to see all uncompleted transactions.
2. **User selects the transaction to cancel**:
   1. The user selects the specific pending transaction they wish to cancel.
3. **User clicks “Cancel Transaction”**:
   1. The user is prompted to confirm the cancellation of the transaction.
4. **System verifies transaction status**:
   1. The system checks the status of the transaction (must be pending or in-progress).
   2. If the transaction can be canceled, the system proceeds to cancel the transaction.
5. **Payment gateway processes cancellation**:
   1. The system contacts the payment gateway (Stripe/PayPal) to reverse the payment process if applicable.
6. **User receives cancellation confirmation**:
   1. The user is notified that the transaction has been successfully canceled, and the transaction status is updated to “Cancelled”.

##### **Alternative Flow:**

1. **Transaction already completed**:
   1. If the transaction has already been processed, the system informs the user that the cancellation is no longer possible and provides next steps (e.g., requesting a refund).

##### **Postconditions:**

* The transaction is marked as canceled.
* If the payment was processed, the system initiates a refund process (if applicable).
* The user’s account balance is updated accordingly.

### **Use Case 10: User Profile Management**

**Goal**: To enable users to manage their profile, including updating personal details and contact information.

**Actors**: User (Customer)

##### **Preconditions:**

* The user is logged into the app.

##### **Basic Flow:**

1. **User navigates to Profile Settings**:
   1. The user selects the "Profile" section in the app menu.
2. **User views current profile details**:
   1. The app displays the user's current profile information, such as name, email address, phone number, and linked payment methods.
3. **User updates profile details**:
   1. The user can modify their contact information (e.g., email, phone number), password, or linked payment methods.
4. **User confirms changes**:
   1. The user clicks the "Save Changes" button after reviewing the updates.
5. **System validates updates**:
   1. The system verifies that the changes are correct (e.g., valid email format, phone number format, etc.).
6. **User receives confirmation**:
   1. The system notifies the user that their profile has been successfully updated.

##### **Alternative Flow:**

1. **Invalid input data**:
   1. If the input data (e.g., email or phone number) is invalid, the system notifies the user and prompts them to correct it.

##### **Postconditions:**

* The user’s profile is updated with the new information.
* The system ensures that the data is securely stored and updated in the backend.

### **Use Case 11: User Authentication and Authorization**

**Goal**: To ensure that only authorized users can access and perform actions on their account.

**Actors**: User (Customer), System (Authentication Service)

##### **Preconditions:**

* The user has an existing account.

##### **Basic Flow:**

1. **User opens the app**:
   1. The user launches the app and is prompted to log in.
2. **User enters login credentials**:
   1. The user enters their username and password.
3. **System authenticates the user**:
   1. The system checks the user’s credentials against the stored data in the backend.
4. **User is granted access**:
   1. If the credentials are valid, the user is granted access to their account and redirected to the home page.
5. **Two-factor authentication (if enabled)**:
   1. If two-factor authentication (2FA) is enabled, the system prompts the user to enter a verification code sent to their email or phone.
6. **User successfully logs in**:
   1. Once the 2FA is validated, the user gains access to their account.

##### **Alternative Flow:**

1. **Invalid login credentials**:
   1. If the username or password is incorrect, the system prompts the user to re-enter the correct credentials.
2. **Account locked**:
   1. After multiple failed attempts, the system locks the account for security reasons and sends a reset link to the user’s email.

##### **Postconditions:**

* The user is securely authenticated and authorized to access the app’s features.
* The system logs authentication attempts for auditing and security purposes.