Part 1 : Overview & Software Requirements Specification

1- Introduction

A) Purpose

The purpose of this Instant Money Transfers and Bank-Accounts is a real-time electronic fund transfer system designed to allow individuals and businesses to transfer money instantly and securely across participating financial institutions. The system aims Useful for paying suppliers or collecting payments quickly, Enables seamless payments for e-commerce purchases, Sending Money to Friends or

B) Project Scope

The system will offer the following features:

- User registration and profile management.
- Bank statement and available transactions.
- List of previous transactions.
- List of Enabled Banks.
- Transaction Limits and Fees.
- Request Money from Other User.
- Setting and Customer Servies.

C) Glossary and Abbreviations

Term	Definition.
IPA	Instant Payment Address.
P2P	(Peer-to-Peer) Direct transfer of funds between individuals.

IMT	(Instant Money Transfer) transfer of funds between accounts.
BAA	(Bank Account Access)
Transaction Limit	Maximum transfer amount per transaction.
24/7Availability	System operates round-the-clock, including weekends/holidays
OTP	(One-Time PIN) SMS-based security code for authorizing transactions.

D) List of System Stakeholders

1. Admins: Manage the overall system and maintain data integrity.
2. Users: Send/receive money instantly, check balances, pay bills.
3. Banks :Provide System services, ensure compliance, and earn transaction fees.
4. Vendors :Integrate with instant payment systems.
5. Customer Support Team: Resolve transaction issues user inquiries.
6. Payment Service Providers (PSPs) :Facilitate transactions

E) References

- Bangkok Sentral ng Filipinas.
- World Bank.
- European Banking .

2- Functional Requirements

a) User Requirements Specification

User Requirement	Rationale
Users must register their mobile number with their bank	Ensures that the app can identify and validate users based on their phone number, which is a foundational requirement for secure transactions.
Users must link all their bank accounts and prepaid cards	Provides a consolidated view of financial assets, simplifying account management and allowing users to conduct transactions from any linked account.
Users must be able to send money using a mobile number or IPA	Enhances ease of use by simplifying the transaction process and removing the need for lengthy account numbers.
Users must be able to check their account balance	Empowers users to stay informed about their finances at all times, improving trust and financial control.
Users must be able to view mini-statements for their accounts	Helps users track recent transactions easily without needing to visit a bank branch or access internet banking.
Users must follow step-by-step registration for their bank	Ensures seamless account onboarding with proper guidance, reducing user errors during the registration process.
Users must send money to any bank, mobile wallet, or card	Extends the app's versatility by allowing money transfers across multiple platforms, fostering a comprehensive and inclusive financial ecosystem.
Users must use a valid debit card for their accounts	Enables secure account verification and guarantees transactions adhere to banking standards.

b) System Requirements Specification

System Functionality	Rationale
The system must operate 24/7	Ensures that users can access the app at any time, improving convenience and service reliability.
The system must securely process transactions within 5 seconds	Reduces waiting time and ensures a high-performing, user-friendly experience that meets user expectations for an instant payment app.
The system must support encrypted data transmissions	Protects sensitive user information (e.g., personal data, account details) during transfers, adhering to security regulations and building trust.
The system must maintain a registry of supported banks and guide users	Ensures compatibility with banks while assisting users in linking accounts, minimizing onboarding issues.
The system must validate user identity using registered mobile numbers	Strengthens security by confirming that only authorized users can access and link accounts, aligning with Central Bank of Egypt regulations.

System Functionality	Rationale
The system must maintain a ledger of user transactions and mini-statements	Provides users access to recent transaction records while supporting audit requirements and historical tracking.
The system must integrate with the Central Bank of Egypt's Instant Payment Network (IPN)	Ensures real-time settlement of transactions across banks, improving overall system efficiency and compliance with regulatory requirements.
The system must facilitate onboarding with prepaid card numbers	Offers an alternative, simplified method for users without direct bank accounts, broadening the app’s appeal.

c) Requirements Priorities (Using the Moscow Scheme)

Priority	Requirement
Must Have	- User registration with mobile numbers linked to their accounts.
Must Have	- Secure data encryption and compliance with Central Bank regulations.
Must Have	- Instant money transfer functionality with a response time under 5 seconds.
Must Have	- Support for 24/7 operations to ensure user convenience.
Must Have	- Onboarding functionality for multiple accounts and prepaid cards.
Should Have	- Mini-statements displaying up to the last 10 transactions.
Should Have	- Compatibility with additional e-wallets or third-party payment platforms.
Could Have	- Advanced search for past transactions beyond mini-statements.
Could Have	- Optional biometric authentication for extra security (e.g., fingerprint or face recognition).
Won’t Have	- Offline transaction capabilities or cash withdrawals in the initial version.

3- NON-Functional Requirements

Look-and-Feel Requirements	These describe the product's visual appearance and the overall user interface (UI) experience. Example: • The system must have a clean, modern interface with a consistent color scheme to convey trust and professionalism. • The login screen should display the company's logo at the top and have clearly labeled fields for "Username" and "Password." • All buttons, such as "Transfer Funds" or "Check Balance," must have a responsive design that adjusts well to mobile screens.
Usability s Humanity Requirements	These specify how **easy** the product should be to use, and any **special usability** considerations related to the user. **Example:** • The system should allow users to complete a money transfer within **three simple steps**: Enter the recipient's details, enter the amount, and confirm the transfer. • There should be **tooltips** on key fields to assist users who may be unfamiliar with how the system works (e.g., explaining what a "SWIFT code" is). • The system should provide **error messages** that are **clear, friendly**, and **actionable** (e.g., "Your transfer failed due to insufficient funds. Please check your balance and try again.").
Performance Requirements	This covers the **speed, accuracy, safety**, and **reliability** with which the system should perform. **Example:** • The system should process **money transfer transactions within 2 seconds**.

	• The system must handle up to **1,000 concurrent transactions** per minute without slowing down. • Transfers should be **100% accurate** in terms of the amounts and recipient details.
Operational s Environmental Requirements	This category outlines the **environment** in which the product will operate, and any special conditions that must be considered. **Example:** • It must operate in environments where **low internet bandwidth** is available, allowing users to still perform essential operations like viewing their balance and initiating transfers. • **Backup power systems** must be in place to ensure the system remains operational during power outages
Maintainability s Support Requirements	This refers to how easily the system can be **updated** and **maintained** over time. **Example:** • The system must allow for **easy updates** to the backend, such as the addition of new payment methods or changes to fees, without requiring a full system shutdown. • **Clear documentation** should be provided for future developers to ensure the system can be easily maintained and modified over time. • **Error logging** must be implemented to help the support team quickly identify and address issues.
Cultural Requirements	These involve considerations related to the **people** involved in developing, using, or interacting with the system, reflecting their **cultural backgrounds** and preferences. **Example:**

	• The system must be available in **multiple languages**, such as **English**, **Arabic**, and **Spanish**, to serve users in different regions. • Currency symbols should automatically adapt based on the user's location (e.g., $ for US dollars, € for Euros). • **Date formats** and **time zones** must align with the cultural norms of the user's region (e.g., DD/MM/YYYY format in Europe).
Legal Requirements	These are the **laws, regulations**, and **standards** that apply to the system. **Example:** • The system must comply with **GDPR** (General Data Protection Regulation) in the European Union for **user data privacy**. • The system should follow **anti-money laundering (AML)** regulations and flag suspicious transactions for review. • It must ensure that all financial transactions are **fully auditable** and meet local **financial regulatory standards**.
Security Requirements	These define how **secure** and **confidential** the product must be, ensuring that sensitive data is protected. **Example:** • All user data (e.g., account numbers, personal information) must be **encrypted** during transmission and storage. • Users must authenticate with **two-factor authentication** (2FA) to complete sensitive actions such as initiating a large transfer. • The system should ensure **role-based access control** (RBAC) so that only authorized users can perform specific administrative tasks.

Some mathematical specifications

Security Requirements: Password Length ≥ 8 characters

4- Design & Implementation Constraints

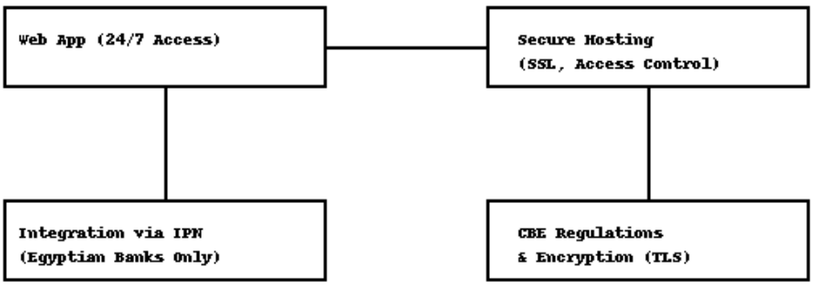
The design and implementation of the Instant Money Transfer and 24/7 Bank Account Access system are subject to several constraints. These include technological limitations, regulatory requirements from the Central Bank of Egypt, and security and integration policies. These constraints are essential to ensure system compliance, security, and operational efficiency.

Constraint Type	Description
Platform Constraint	The system must be implemented as a web application, and optionally support mobile responsiveness, to allow 24/7 access.
Technology Stack	The system is restricted to using HTML, CSS, JavaScript, and optionally a backend framework (like Node.js, Flask, or PHP) depending on team expertise.
Security Regulations	All data handling and transactions must comply with Central Bank of Egypt (CBE) regulations, using secure payment protocols and encryption standards (e.g., HTTPS, TLS).
Integration Limitation	The app must support integration with Egyptian banks only via Instant Payment Network (IPN), limiting compatibility with international banking APIs.
User Authentication	The mobile number must be pre-registered at the bank and linked to a valid debit or prepaid card, which is a requirement for user onboarding.
Third-Party Services	Only licensed financial institutions and APIs approved by the Central Bank of Egypt can be used to interact with user accounts.
Performance Requirement	The system must process transactions instantly, meaning any bottlenecks in API communication or database access must be minimized or optimized.
Deployment Constraints	The web app must be deployed on a secure hosting environment, with SSL certificates enabled and proper access controls in place.

Mathematical Specification

Let B be the set of supported banks, B □ Banks_Egypt. A transaction T is valid if and only if source_bank(T) □ B and target_bank(T) □ B.
Let U be the set of users, M be the set of mobile numbers linked to accounts.
□u □ U, □m □ M: linked(u, m) = True → onboard(u) = True

Graphical Notation (Diagram)



5) Anticipated Changes :
Part 1:

1. Hardware Evolution

Change	Rationale
Processing power improvements	Future hardware will offer increased computational capabilities, enabling more complex financial operations and real-time data processing
Storage capacity and speed enhancements	Banking transaction volumes grow continuously, requiring efficient storage and retrieval of larger datasets
Network bandwidth improvements	Financial systems increasingly rely on real-time communication and low-latency transactions

2. User Needs Evolution

Change	Rationale
Enhanced mobile experience	Users expect banking apps to work seamlessly across all devices with consistent interfaces
Biometric authentication	Growing user preference for convenient yet secure login methods beyond traditional passwords
Personalized financial insights	Users increasingly expect AI-powered analysis of their spending and saving patterns
Support for new payment methods	Digital wallets, cryptocurrencies, and other emerging payment technologies

3. Regulatory Evolution

Change	Rationale
Enhanced data privacy requirements	Laws like GDPR continue to evolve with stricter requirements for financial data
New anti-money laundering regulations	Financial systems must adapt to changing AML requirements

Open banking standards	Requirements for banks to share data through secure APIs with authorized third parties

Part 2: Impact on System Design 1. Design Principles for Future-Ready Banking Systems

Principle	Description	Impact
Modularity	System built with independent components	Allows updates to authentication methods without affecting transaction processing
API-First	All functionality exposed through well-defined APIs	Enables third-party integration and consistent experience across devices
Cloud-Native	System designed for cloud deployment	Facilitates scaling to handle varying transaction loads and geographic expansion
Configurable Features	Business rules stored in configuration not code	Allows quick adaptation to regulatory changes without code deployment

2. Implementation Strategies

Strategy	Description	Example
Feature Flags	Toggle features on/off via configuration	Gradually roll out new payment methods to specific user segments
Automated Testing	Comprehensive test coverage	Ensure new authentication methods don't break existing functionality
Continuous Integration	Frequent small updates	Quickly respond to security threats and regulatory changes
Versioned APIs	Support multiple API versions	Maintain backward compatibility as features evolve

Part 3: System Evolution Requirements Template

Description	Rationale	Priority
The system must support adding new banks without downtime	Allow expansion of banking partners without service interruption	High
The system must allow adding new authentication methods	Keep pace with security innovations and user preferences	High
The system must support multiple API versions	Prevent breaking changes for mobile apps during updates	Medium
The UI must adapt to different screen sizes	Maintain consistent user experience across all devices	Medium
Transaction processing must scale horizontally	Handle increased transaction volume during peak periods	High

6) Requirements Discovery Approaches

number	Approach	Description	Purpose	Rationale
1	Remote User Testing	Users test the app remotely on their own devices.	Observe natural interaction with the app.	Helps understand real behavior and user struggles.
2	A/B Testing	Present different versions of a feature or UI to users.	Improve the user interface based on real usage.	Determines which version performs better through data.
3	Data Analytics & Usage Metrics	Monitor in-app behavior using tools like Google Analytics.	Data-driven decision making.	Identifies frequently used features and common issues.
4	In-App Surveys	Surveys presented inside the app after key actions.	Collect immediate user feedback.	Provides real-time insights on the user experience.
5	Social Media Monitoring	Track user opinions on social platforms and forums.	Understand indirect feedback and emerging issues.	Users often share honest feedback on social media.
6	Transaction Logs Analysis	Analyze transaction records within the app.	Detect performance issues and failures.	Effective for identifying technical issues without user input.
7	Focus Groups	Online discussion sessions with selected users.	Collect detailed opinions and feedback.	Helps understand collective perspectives and ideas.
8	Interviews (remote & optional)	Remote interviews when deeper insights are needed.	Gain direct feedback.	Used cautiously due to sensitive nature of the data.
9	Questionnaires	Distribute digital forms after users have used the app.	Get broad feedback safely from many users.	Efficient and secure way to gather large-scale input.
10	Observation	Watch how users interact with the app in real-time.	Understand usability issues and behavior.	Reveals problems users might not report directly.
11	Prototyping	Test a prototype of the app with a group of users.	Collect feedback before full development.	Helps reduce cost of changes and ensures better design.

7) Requirements Validation Techniques

Requirements validation is a crucial step to ensure that the specified requirements accurately reflect user needs and align with the system's objectives. Below are the key validation techniques with illustrative examples:

1. Requirements Reviews

Description:
These are organized meetings involving system analysts, clients, end users, and developers to review the requirements document.

Example:
After preparing the requirement document for an electronic library management system, a meeting is held with the project team and the client to go through each item, ensuring that every requirement is well-understood and agreed upon.

2. Prototyping

Description:
This involves creating a simple prototype of the system to demonstrate ideas to users and gather feedback.

Example:
When designing a medical appointment booking app, interactive interfaces can be built using tools like Figma or Adobe XD and shared with doctors and users for feedback, which is then used to refine the requirements.

3. User Acceptance Testing (UAT)

Description:
This is the process of having the end user test the system to ensure that the implemented requirements meet their expectations.

Example:
Before delivering an attendance and leave management system to a company, a trial version is given to the HR team to test and provide feedback.

4. Requirements Traceability

Description:
This involves linking each requirement to a specific phase, function, or code element to track its implementation.

Example:
In an electronic payment system project, the requirement “Send a confirmation email after a payment” is linked to a specific function in the email-sending code to ensure it is implemented.

5. Surveys and Questionnaires

Description:
These are used to collect quantitative feedback from a wide range of users.

Example:
A questionnaire is sent to a group of students after presenting the initial specifications for a course registration system to assess how well it meets their needs.

Additional Detailed Examples of Requirements Validation Techniques

Example 1: Requirements Reviews in a Food Delivery App Project

While developing a food delivery app, a meeting was conducted with restaurant owners, drivers, and potential users to review the requirements. One requirement stated: “The app should display the estimated delivery time.” During the review, a driver suggested adding a feature to update the estimated time in case of traffic delays. As a result, the requirement was revised to be more accurate and realistic.

Example 2: Prototyping in an Event Ticket Booking System

A prototype of the user interface was created using Figma. After presenting it to users, it became clear that the seat selection process was unclear. Users suggested using color codes to differentiate between available and reserved seats. Based on this feedback, the design was updated to improve the user experience before moving to the development stage.

Example 3: UAT in a School Management System

A UAT was conducted with school staff after completing a beta version of the system. One teacher noted that the grade entry system did not allow partial grade submissions. The requirement was modified to allow saving and updating grades in stages.

Example 4: Real-Time Money Transfer and Bank Account Access System

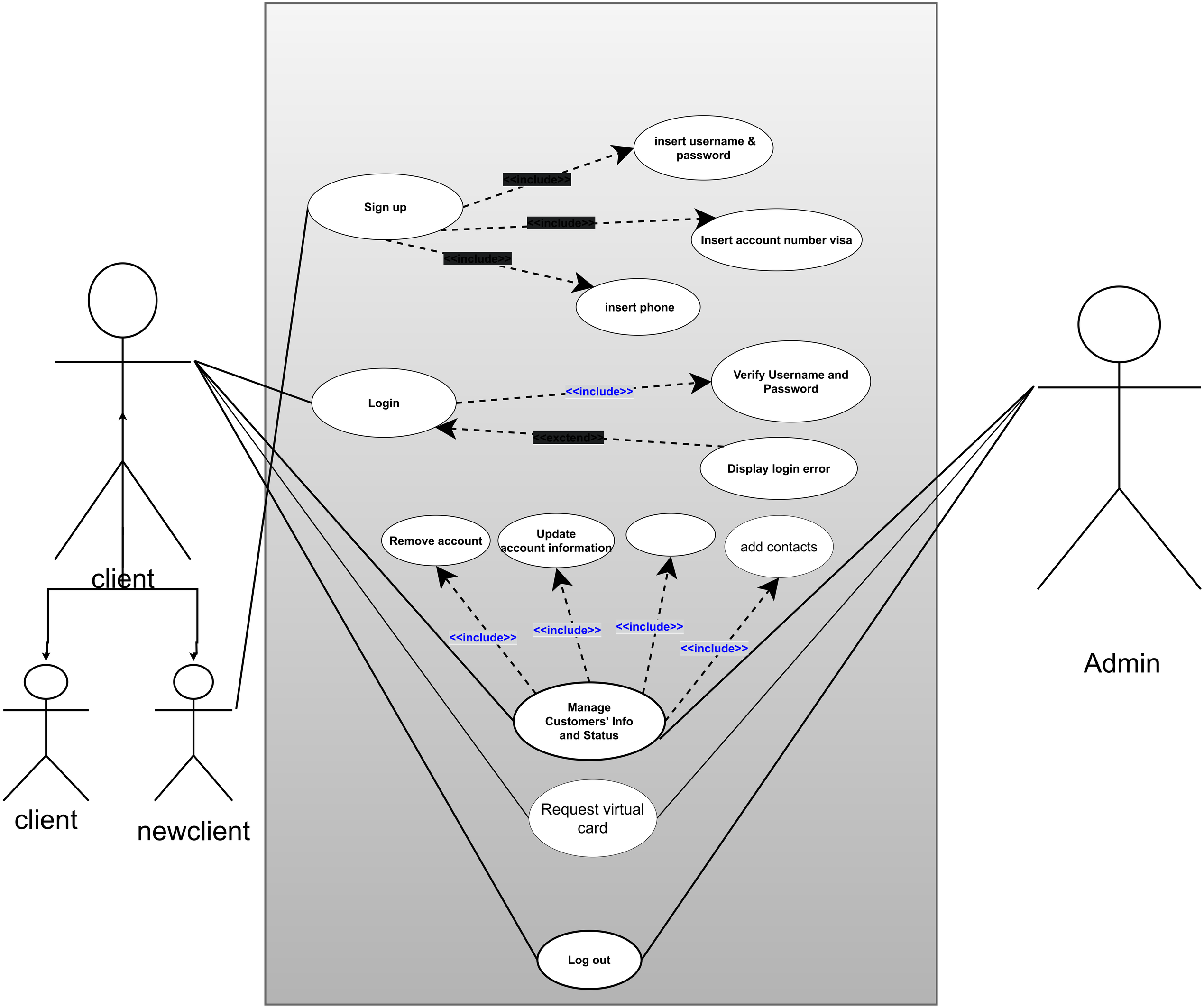
- Requirements Reviews: The security team suggested adding a requirement to send instant notifications for every transaction. This was not included in the initial version but was added after the review.
- Prototyping: A prototype of the money transfer interface was shown to users. Some reported that the bank account selection interface was unclear, prompting a redesign to show account names and balances more clearly.
- User Acceptance Testing: Users noticed that the system didn’t show a final confirmation before completing a transfer. A confirmation screen summarizing the transaction was added.
- Requirements Traceability: Every functional requirement, such as “Instant transfer between multiple accounts,” was linked to specific code modules to ensure nothing was overlooked during development

PART 2: System Design & Models

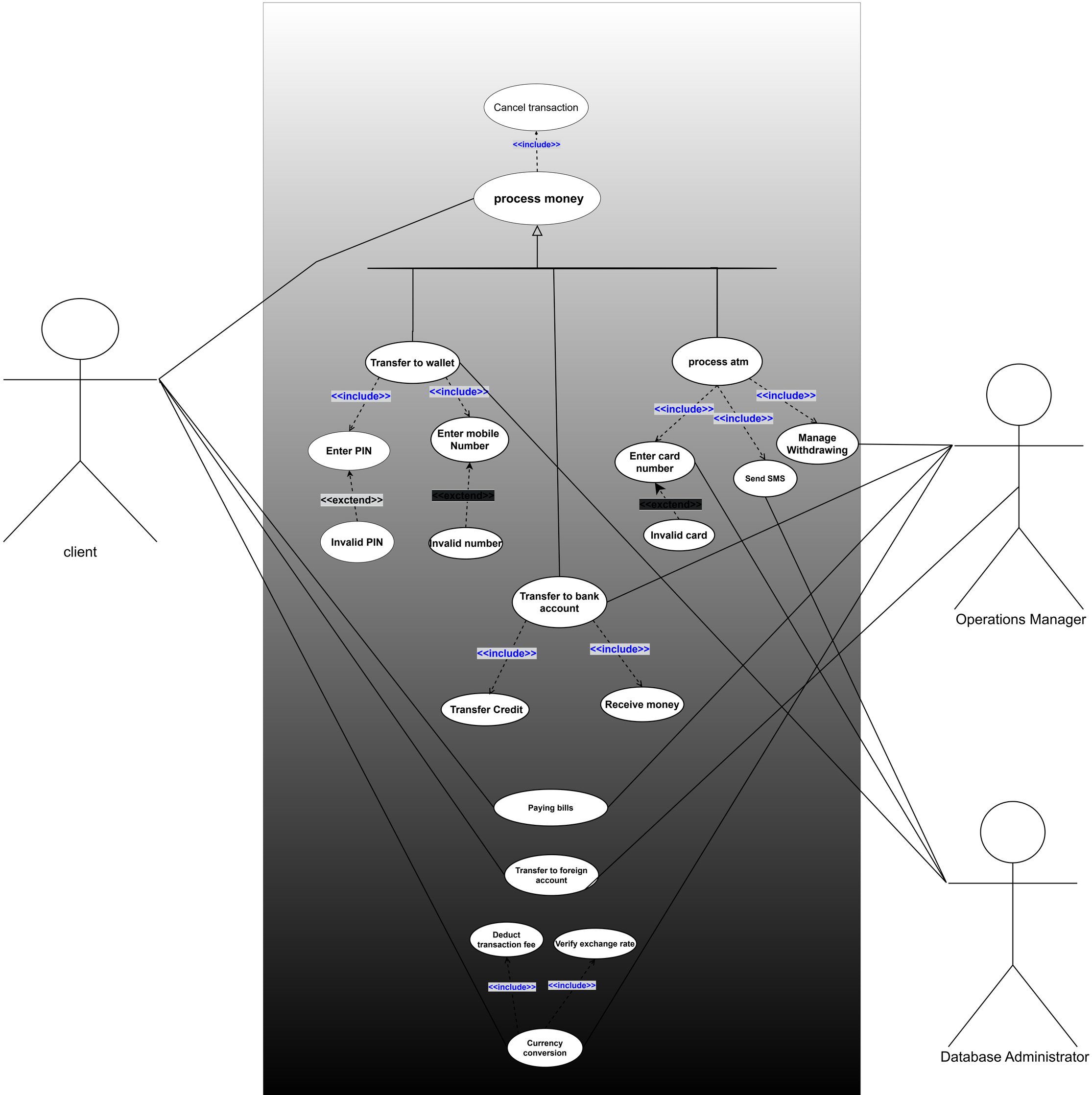
8) Functional diagrams

Use case diagram :

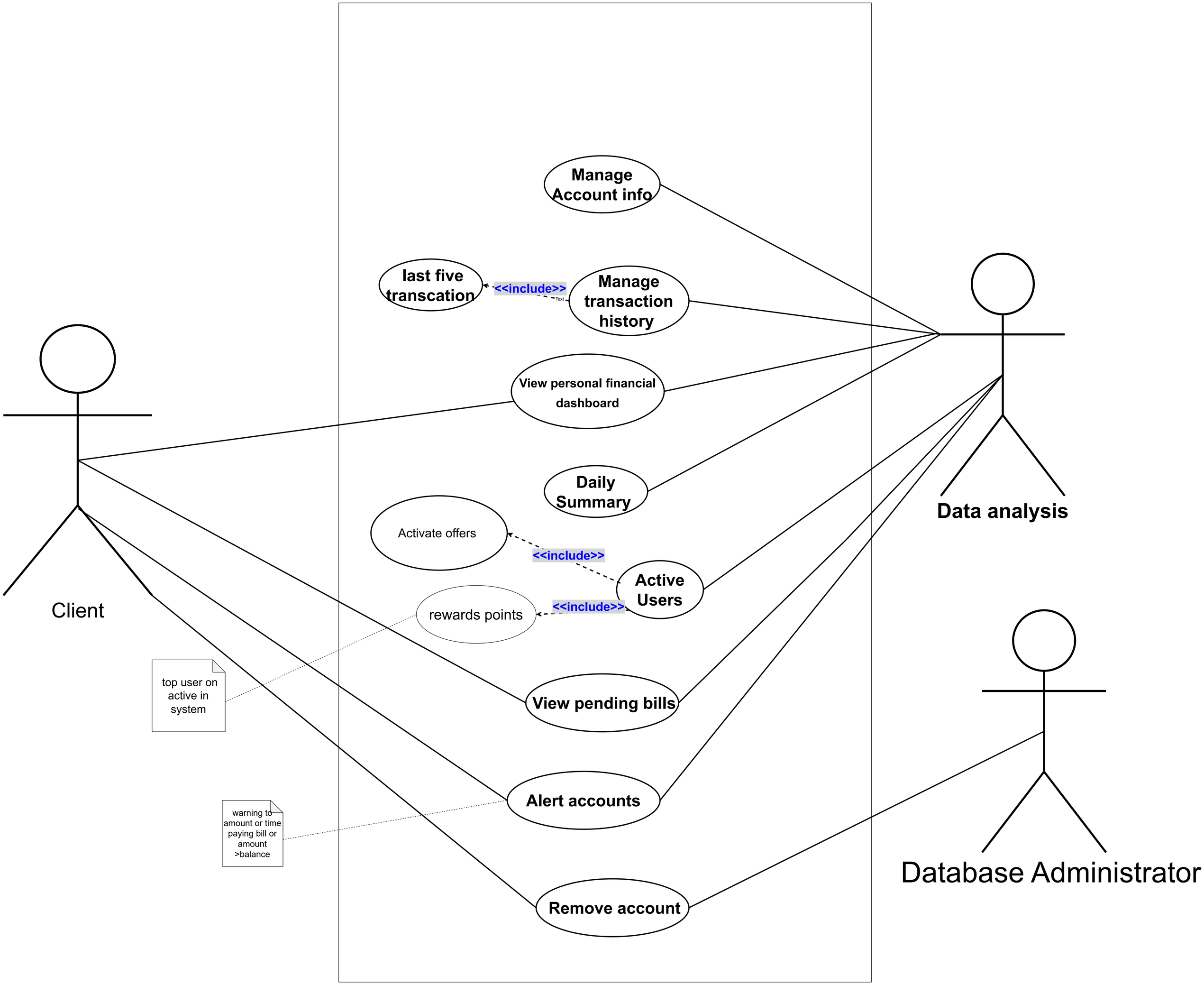
Banking System



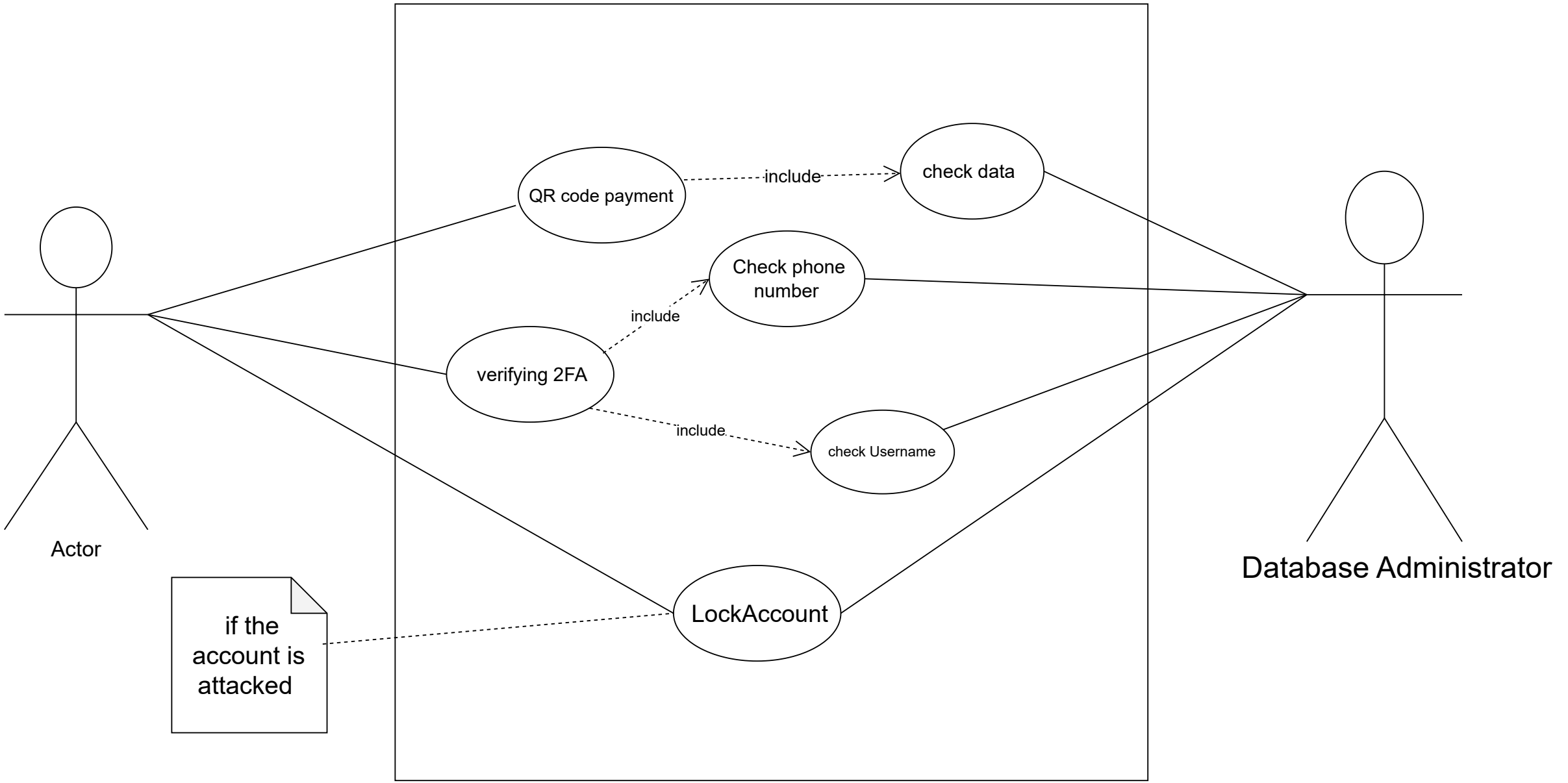
Banking System

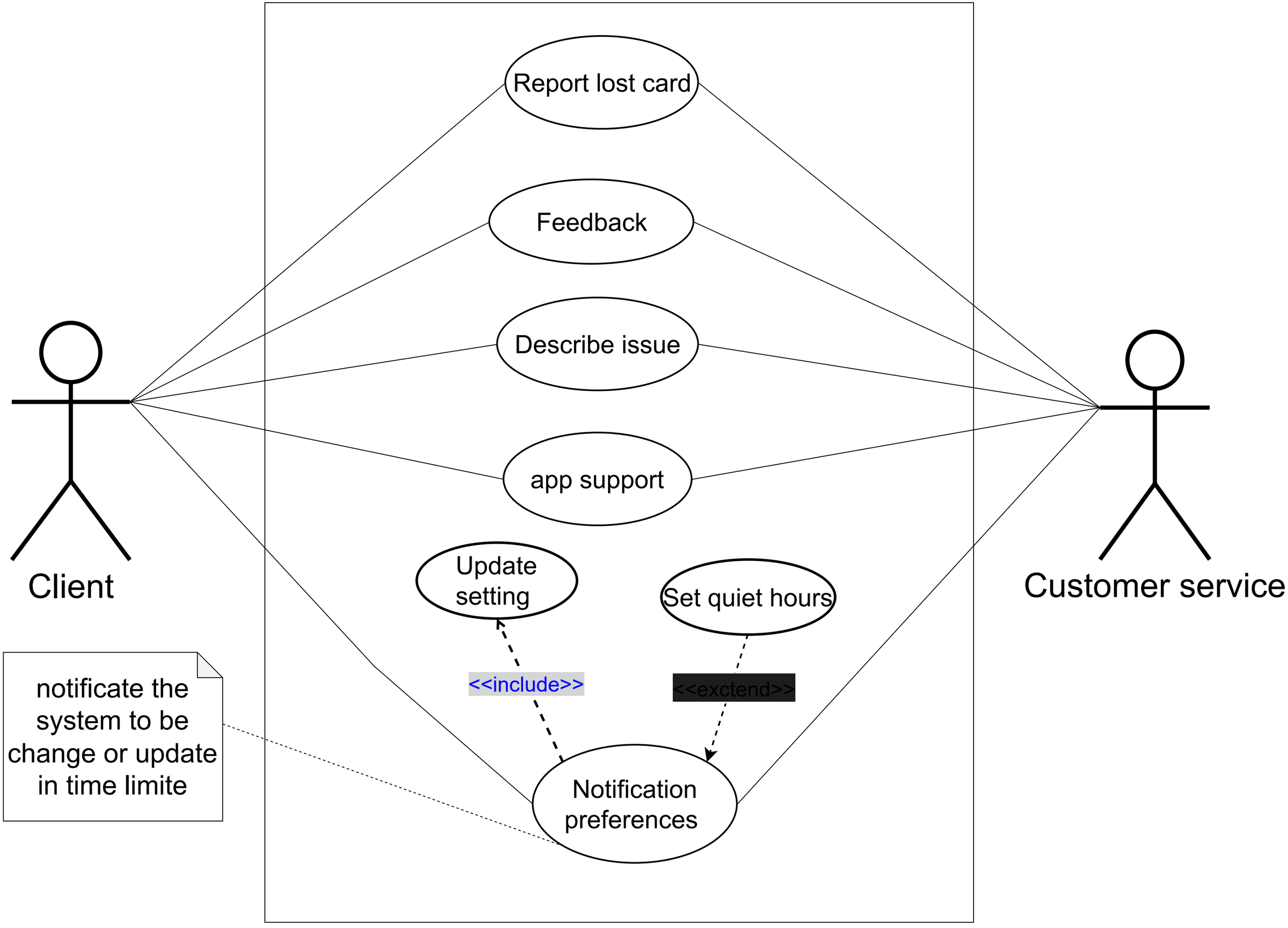


Banking System



Bank system


QR code payment
include
check data
Check phone number
include
verifying 2FA
include
check Username
LockAccount
Actor
Database Administrator
if the account is attacked


Report lost card
Feedback
Describe issue
app support
Update setting
Set quiet hours
<<include>>
<<exctend>>
Notification preferences
Client
Customer service
notificate the system to be change or update in time limite

Customer Support Use Cases

UC-001: Report Lost Card

Use Case ID	**UC-001**
Use Case Name	Report Lost Card
Goal	Allow users to report a lost or stolen card to block unauthorized transactions
Initiator	Client
Pre-condition(s)	1. User is registered in the system 2. User has at least one card linked to their account
Post-condition(s)	1. The reported card is marked as lost in the system 2. The card is blocked for transactions 3. A replacement card process may be initiated
Main Success Scenario	1. Client logs into the application 2. Client navigates to the card management section 3. Client selects the "Report Lost Card" option 4. Client selects the specific card to report as lost 5. Client confirms the action 6. System marks the card as lost and blocks it 7. System confirms the action to the client 8. System notifies the customer service department 9. Customer service contacts the client for further action
Alternative Scenarios	**Alternative Scenario 1:** - At step 6, if the system cannot process the request immediately: 1. System stores the request for processing 2. System informs the client that the request is being processed 3. System attempts to process the request again 4. Once successful, system notifies the client
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 5, if the network connection fails: 1. System displays an error message 2. System prompts the client to try again 3. Client is advised to contact customer service directly if the problem persists

UC-002: Feedback

Use Case ID	**UC-002**
Use Case Name	Feedback
Goal	Allow users to submit feedback about the application or services
Initiator	Client
Pre-condition(s)	1. User has an active account in the system
Post-condition(s)	1. Feedback is recorded in the system 2. Feedback is routed to the appropriate department
Main Success Scenario	1. Client logs into the application 2. Client navigates to the feedback section 3. Client selects the type of feedback (suggestion, complaint, appreciation) 4. Client enters the feedback details 5. Client submits the feedback 6. System confirms receipt of feedback 7. System routes the feedback to customer service for review
Alternative Scenarios	**Alternative Scenario 1:** - At step 3, if the client chooses to attach screenshots: 1. System prompts for an attachment 2. Client attaches relevant images 3. Process continues at step 4
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 5, if the feedback cannot be submitted: 1. System stores the feedback locally 2. System attempts to resubmit when the connection is available 3. System notifies the client of the status

UC-003: Describe Issue

Use Case ID	**UC-003**
Use Case Name	Describe Issue
Goal	Allow users to report specific problems or issues with the service
Initiator	Client
Pre-condition(s)	1. User has an active account in the system
Post-condition(s)	1. The issue is recorded in the system 2. A tracking number is assigned to the issue 3. The issue is routed to the appropriate technical team
Main Success Scenario	1. Client logs into the application 2. Client navigates to the support section 3. Client selects the Describe Issue option 4. Client selects an issue category (transaction, login, card, etc.) 5. Client describes the issue in detail 6. Client submits the issue report 7. System generates a tracking number 8. System confirms submission and provides the tracking number to the client 9. System routes the issue to customer service for resolution
Alternative Scenarios	**Alternative Scenario 1:** - After step 6, if the client wishes to attach transaction details: 1. System offers the option to attach transaction history 2. Client selects relevant transactions 3. System attaches the transaction details to the issue report
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - If submission fails at step 6: 1. System saves the draft locally 2. System prompts the client to retry or contact support via alternative means 3. System provides contact information for direct support

UC-004: App Support

Use Case ID	UC-004
Use Case Name	App Support
Goal	Provide users with technical assistance and information about app functionality
Initiator	Client
Pre-condition(s)	1. User has the application installed
Post-condition(s)	1. The client receives the necessary support or information
Main Success Scenario	1. Client opens the application 2. Client navigates to the support section 3. Client selects App Support 4. System displays categories of support (FAQs, tutorials, contact support) 5. Client selects the appropriate category 6. System provides the requested information or connects the client to a support representative 7. The client resolves their query or issue
Alternative Scenarios	**Alternative Scenario 1:** - At step 5, if the client selects live chat support: 1. System connects the client to an available support agent 2. Client communicates with the agent via the chat interface 3. The agent assists the client with their query
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 6, if support information is outdated: 1. System allows the client to flag the content as outdated 2. System records the feedback 3. System offers alternative support options

UC-005: Update Setting

Use Case ID	**UC-005**
Use Case Name	Update Setting
Goal	Allow users to modify their application settings
Initiator	Client
Pre-condition(s)	1. User is logged into the application
Post-condition(s)	1. The user's settings are updated according to their preferences
Main Success Scenario	1. Client logs into the application 2. Client navigates to the settings section 3. Client selects the setting they wish to modify 4. Client makes the desired changes 5. Client saves the changes 6. System confirms the changes have been saved 7. System applies the new settings immediately
Alternative Scenarios	**Alternative Scenario 1:** - At step 5, if the client decides to revert changes: 1. Client selects the Cancel option 2. System asks for confirmation 3. Client confirms the cancellation 4. System retains the original settings
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 5, if settings cannot be saved: 1. System displays an error message 2. System retains the original settings 3. System suggests possible troubleshooting steps

UC-006: Set Quiet Hours

Use Case ID	UC-006
Use Case Name	Set Quiet Hours
Goal	Allow users to specify times when they do not want to receive notifications
Initiator	Client
Pre-condition(s)	1. User is logged into the application
Post-condition(s)	1. Quiet hours preferences are saved in the system 2. Notifications are suppressed during the specified quiet hours
Main Success Scenario	1. Client logs into the application 2. Client navigates to the notification settings 3. Client selects Set Quiet Hours 4. Client specifies start and end times for quiet hours 5. Client selects the days of the week for quiet hours 6. Client saves the quiet hours settings 7. System confirms the settings are saved 8. System applies quiet hours settings for notifications
Alternative Scenarios	**Alternative Scenario 1:** - At step 4, if the client selects Do Not Disturb mode: 1. System offers predefined quiet hours settings 2. Client selects the preferred option 3. Process continues from step 6
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 6, if the settings conflict with critical notification requirements: 1. System displays a warning message 2. System suggests modifications to the settings 3. Client either adjusts the settings or acknowledges an override for critical notifications

UC-007: Notification Preferences

Use Case ID	UC-007
Use Case Name	Notification Preferences
Goal	Allow users to customize which notifications they receive and how
Initiator	Client
Pre-condition(s)	1. User is logged into the application
Post-condition(s)	1. The user's notification preferences are saved 2. Future notifications are delivered based on these preferences
Main Success Scenario	1. Client logs into the application 2. Client navigates to the notification settings 3. Client views the current notification preferences 4. Client modifies preferences (transaction alerts, security alerts, marketing, etc.) 5. Client selects the delivery methods (push, SMS, email) 6. Client saves the preferences 7. System confirms the changes 8. System applies the new notification settings
Alternative Scenarios	**Alternative Scenario 1:** - At step 4, if the client enables Update Setting: 1. System displays additional settings for system updates 2. Client specifies preferences for update notifications 3. Process continues from step 6
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 6, if certain critical notifications cannot be disabled: 1. System displays information about mandatory notifications 2. System allows customization for the other notifications 3. The system saves the changes that are permitted

UC-008: Sign Up

Use Case ID	UC-008
Use Case Name	Sign Up
Goal	Allow new users to create an account in the banking system
Initiator	New Client
Pre-condition(s)	1. User has the application installed 2. User is not already registered
Post-condition(s)	1. A new user account is created 2. The user can access the banking services
Main Success Scenario	1. New client opens the application 2. New client selects Sign Up 3. System requests personal information 4. New client enters a username and password 5. New client enters account number/visa details 6. New client enters a phone number 7. System validates the information 8. System creates a new account 9. System confirms successful registration 10. System sends a verification code to the provided phone number 11. New client enters the verification code 12. System activates the account
Alternative Scenarios	**Alternative Scenario 1:** - At step 4, if the username is already taken: 1. System notifies the client 2. System suggests alternative usernames 3. Client enters a different username 4. Process continues from step 5
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 7, if the provided bank account information cannot be verified: 1. System displays an error message 2. System suggests contacting the bank 3. The registration process is paused until verification is complete

UC-009: Login

Use Case ID	**UC-009**
Use Case Name	Login
Goal	Allow registered users to access their accounts
Initiator	Client
Pre-condition(s)	1. User has a registered account 2. User has the application installed
Post-condition(s)	1. The user is authenticated 2. The user can access their account and banking services
Main Success Scenario	1. Client opens the application 2. Client selects Login 3. Client enters the username and password 4. System verifies credentials 5. System authenticates the client 6. System displays the client's dashboard
Alternative Scenarios	**Alternative Scenario 1:** - After step 4, if the system requires additional verification: 1. System sends a verification code to the registered phone number 2. Client enters the verification code 3. Process continues from step 5
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 4, if credentials are incorrect: 1. System displays a login error 2. System prompts the client to try again 3. System offers a password recovery option after multiple failed attempts

UC-010: Manage Customers' Info and Status

Use Case ID	UC-010
Use Case Name	Manage Customers' Info and Status
Goal	Allow users to view and manage their personal information and account status
Initiator	Client
Pre-condition(s)	1. User is logged into the system
Post-condition(s)	1. The user's information is viewed or updated as requested
Main Success Scenario	1. Client logs into the application 2. Client navigates to the account management section 3. Client selects Manage Info and Status 4. System displays current personal information and account status 5. Client reviews the displayed information 6. Client makes any necessary updates 7. Client saves the changes 8. System confirms that the changes have been saved
Alternative Scenarios	**Alternative Scenario 1:** - At step 6, if the client selects Remove Account: 1. System requests confirmation 2. Client confirms account removal 3. System processes the request 4. System confirms account removal **Alternative Scenario 2:** - At step 6, if the client selects Update Account Information: 1. System allows editing of personal details 2. Client updates the information 3. System validates the new information 4. Process continues from step 7 **Alternative Scenario 3:** - At step 6, if the client selects Add Visa: 1. System prompts for new card details 2. Client enters the card information 3. System validates the card 4. Process continues from step 7 **Alternative Scenario 4:** - At step 6, if the client selects Add Contacts: 1. System displays a contact addition interface 2. Client enters the contact details 3. System validates the contact information 4. Process continues from step 7
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 7, if the system cannot validate the changes: 1. System displays an error message 2. System highlights the problematic fields 3. Client corrects the information 4. Client attempts to save again

UC-011: Request Virtual Card

Use Case ID	UC-011
Use Case Name	Request Virtual Card
Goal	Allow users to request a virtual card for online transactions
Initiator	Client
Pre-condition(s)	1. User is logged into the system 2. User has at least one active bank account
Post-condition(s)	1. A virtual card is created and linked to the user's account 2. The user can use the virtual card for online transactions
Main Success Scenario	1. Client logs into the application 2. Client navigates to the card management section 3. Client selects Request Virtual Card 4. System displays virtual card options and terms 5. Client selects the desired options 6. Client confirms the request 7. System processes the request 8. System generates a virtual card 9. System displays the virtual card details to the client 10. System confirms the successful creation of the virtual card
Alternative Scenarios	**Alternative Scenario 1:** - At step 5, if the client requests specific card limits: 1. System displays limit configuration options 2. Client sets the desired transaction limits 3. Process continues from step 6
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 7, if the client is not eligible for a virtual card: 1. System displays eligibility requirements 2. System suggests steps to become eligible 3. Request is declined

UC-012: Log Out

Use Case ID	**UC-012**
Use Case Name	Log Out
Goal	Allow users to securely exit their account session
Initiator	Client
Pre-condition(s)	1. User is logged into the system
Post-condition(s)	1. The user's session is terminated 2. The user is returned to the login screen
Main Success Scenario	1. Client selects the Log Out option 2. System asks for confirmation 3. Client confirms the logout 4. System terminates the session 5. System returns the client to the login screen 6. System confirms successful logout
Alternative Scenarios	**Alternative Scenario 1:** - At step 3, if the client cancels the logout: 1. System returns the client to the previous screen 2. The session remains active
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 4, if the network connection is lost: 1. System attempts to terminate the session locally 2. System displays a notice that the session may still be active on the server 3. System advises the client to log in and then out again when the connection is restored

UC-013: QR Code Payment

Use Case ID	UC-013
Use Case Name	QR Code Payment
Goal	Allow users to make payments via QR code scanning
Initiator	Actor (User)
Pre-condition(s)	1. User is logged into the application 2. User has an active account with sufficient funds
Post-condition(s)	1. The payment is processed successfully 2. Both user and merchant accounts are updated accordingly
Main Success Scenario	1. User logs into the application 2. User selects the QR Payment option 3. User scans the merchant's QR code using the device camera 4. System processes the QR code information 5. System invokes the check data functionality to validate transaction information 6. System displays payment details to the user 7. User confirms the payment 8. System processes the transaction 9. System confirms successful payment to both the user and the merchant
Alternative Scenarios	**Alternative Scenario 1:** - At step 4, if the QR code is damaged or unclear: 1. System prompts the user to try scanning again or to manually enter the code 2. User rescans or enters the code manually 3. Process continues from step 5
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 5, if payment information cannot be validated: 1. System displays an error message 2. System aborts the transaction 3. User is advised to contact the merchant for a valid QR code

UC-014: Check Data

Use Case ID	**UC-014**
Use Case Name	Check Data
Goal	Validate transaction information before processing payments
Initiator	System (triggered by QR Code Payment)
Pre-condition(s)	1. A transaction request is initiated 2. Payment information is available
Post-condition(s)	1. Transaction data is either validated or rejected
Main Success Scenario	1. System receives the transaction data 2. System verifies merchant information 3. System checks the validity of the transaction amount 4. System confirms the payment details format 5. System validates that the user has sufficient funds 6. System returns a validation success status
Alternative Scenarios	**Alternative Scenario 1:** - At step 5, if funds are insufficient: 1. System calculates the shortfall amount 2. System offers options for additional funding methods 3. User selects an alternative funding method 4. System validates the combined funding sources 5. Process continues to completion
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 2, if merchant information is invalid: 1. System flags a potential security risk 2. System rejects the transaction 3. System notifies the user of potential fraudulent activity

UC-015: Verifying 2FA

Use Case ID	UC-015
Use Case Name	Verifying 2FA
Goal	Authenticate user identity through two-factor authentication (2FA)
Initiator	Actor (User)
Pre-condition(s)	1. User has initiated a sensitive operation or login 2. User has set up 2FA previously
Post-condition(s)	1. The user's identity is successfully verified 2. The user can proceed with the requested operation
Main Success Scenario	1. System sends a verification code to the registered phone 2. System invokes the check username functionality 3. User enters the verification code 4. System validates the code 5. System grants access to the requested operation
Alternative Scenarios	**Alternative Scenario 1:** - At step 4, if the user requests the code via an alternative method: 1. System sends the code via email or an authenticator app 2. User receives the code through the alternative method 3. Process continues from step 5
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 6, if the code is invalid or expired: 1. System displays an error message 2. System offers to resend the code or try an alternative verification method 3. The authentication process is reset

UC-016: Check Phone Number

Use Case ID	UC-016
Use Case Name	Check Phone Number
Goal	Verify the authenticity and accessibility of the user's registered phone number
Initiator	System (triggered by Verifying 2FA)
Pre-condition(s)	1. User has a phone number registered in the system 2. The 2FA process has been initiated
Post-condition(s)	1. The phone number is verified as valid and accessible 2. The system can proceed with sending the verification code
Main Success Scenario	1. System retrieves the user's registered phone number 2. System validates the phone number format 3. System checks if the phone is registered to a legitimate user 4. System confirms that the phone number is active 5. System approves the phone number for verification code delivery
Alternative Scenarios	**Alternative Scenario 1:** - At step 3, if the system detects a recent phone number change: 1. System triggers additional security questions 2. User answers the security questions correctly 3. Process continues from step 4
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 4, if the phone number appears inactive: 1. System flags a potential security issue 2. System offers alternative verification methods 3. If no alternatives are available, the authentication process is halted

UC-017: Check Username

Use Case ID	**UC-017**
Use Case Name	Check Username
Goal	Validate user identity through username verification
Initiator	System (triggered by Verifying 2FA)
Pre-condition(s)	1. User has initiated the authentication process 2. Username has been entered into the system
Post-condition(s)	1. The username is verified as genuine and active
Main Success Scenario	1. System retrieves the entered username 2. System checks that the username exists in the database 3. System verifies that the username is active (not disabled) 4. System confirms that the username matches the account in the authentication process 5. System approves the username verification
Alternative Scenarios	**Alternative Scenario 1:** - At step 4, if the username shows suspicious recent activity: 1. System applies additional security checks 2. System requests additional verification information 3. User provides the required verification 4. Process continues if verification is successful
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 3, if the username is inactive or suspended: 1. System displays an appropriate message 2. System provides steps for account reactivation 3. The authentication process is halted

UC-018: LockAccount

Use Case ID	**UC-018**
Use Case Name	LockAccount
Goal	Secure a user account that shows signs of compromise
Initiator	System or Database Administrator
Pre-condition(s)	1. The account shows suspicious activity or a security breach attempt 2. The account is currently active in the system
Post-condition(s)	1. The account is locked and secured from unauthorized access 2. The user is notified of the account status
Main Success Scenario	1. System detects suspicious activity on the account 2. System evaluates the threat level based on the activity pattern 3. System determines that the account is under attack 4. System automatically locks the account 5. System records the locking event with a timestamp and the reason 6. System notifies the user via alternative contact methods 7. System notifies the Database Administrator of the locked account
Alternative Scenarios	**Alternative Scenario 1:** - At step 3, if a Database Administrator manually initiates the lock: 1. System requests confirmation and a reason 2. Administrator provides the reason and confirms 3. Process continues from step 4
Unsuccessful Scenarios	**Unsuccessful Scenario 1:** - At step 6, if no alternative contact methods are available: 1. System maintains the lock status 2. System creates a high-priority alert for customer service 3. System waits for the user to contact support through verified channels

InstaPay Banking System - Use Case Descriptions

Use Case 19

Item	Description
Use Case ID	UC-19
Use Case Name	Manage Account Info
Goal	Allow users to view and update their account information
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has at least one bank account linked
Post-condition(s)	- User account information is updated successfully - Changes are stored in the system
Main Success Scenario	1. User selects "Manage Account Info" option 2. System displays current account information 3. User modifies desired information (contact details, password, etc.) 4. User submits changes 5. System validates the information 6. System updates the account information 7. System displays confirmation message
Alternative Scenarios	1a. User only views account information without making changes: 1. User exits the account management screen

Use Case 20

Item	Description
Use Case ID	UC-20
Use Case Name	Manage Transaction History
Goal	Allow users to view and filter their complete transaction history
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has conducted at least one transaction
Post-condition(s)	- Full or filtered transaction history is displayed to the user
Main Success Scenario	1. User selects "Manage Transaction History" option 2. System displays transaction filtering options 3. User selects desired filters (or none for full history) 4. System retrieves transactions based on filters 5. System displays transaction list with complete details
Alternative Scenarios	3a. User selects to view last five transactions (include "Last Five Transaction"): 1. System retrieves only the five most recent transactions 2. System displays these transactions with date, amount, type, and recipient/sender information 4a. No transactions match the selected filters: 1. System displays message that no transactions match the criteria 2. User modifies filters or views all transactions

Use Case 21

Item	Description
Use Case ID	UC-21
Use Case Name	View Personal Financial Dashboard
Goal	Provide users with a comprehensive overview of their financial status
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has at least one bank account linked
Post-condition(s)	- Dashboard with financial information is displayed
Main Success Scenario	1. User selects "View Personal Financial Dashboard" option 2. System retrieves account balances, recent transactions, spending patterns 3. System generates visual representations of financial data (charts, graphs) 4. System displays the complete dashboard with all financial information
Alternative Scenarios	2a. System unable to retrieve some information: 1. System displays partial dashboard with available information 2. System shows error message for unavailable sections

Use Case 22

Item	Description
Use Case ID	UC-22
Use Case Name	Daily Summary
Goal	Provide users with a summary of daily financial activities
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has at least one bank account linked
Post-condition(s)	- Daily financial summary is displayed to the user
Main Success Scenario	1. User selects "Daily Summary" option 2. System calculates total inflows and outflows for the current day 3. System identifies significant transactions 4. System displays summary with total spending, income, and balance changes
Alternative Scenarios	2a. No transactions for current day: 1. System displays message indicating no activity for the day 2. System offers to show previous day's summary instead

Use Case 23

Item	Description
Use Case ID	UC-23
Use Case Name	Active Users
Goal	Display and manage active user status and statistics
Initiator	Data Analysis/Database Administrator
Pre-condition(s)	- Administrator is logged into the system - System has registered users
Post-condition(s)	- Active user information is displayed
Main Success Scenario	1. Administrator selects "Active Users" option 2. System retrieves current active user data 3. System displays list of active users with session information 4. Administrator can view detailed information for specific users
Alternative Scenarios	1a. Administrator accesses rewards points information (include "Rewards Points"): 1. System retrieves rewards points data for all users 2. System displays points statistics and redemption activity 1b. Administrator activates special offers (include "Activate Offers"): 1. System displays offer management interface 2. Administrator creates or modifies available offers 3. System applies offers to eligible users 3a. No active users: 1. System displays message indicating no users are currently active

Use Case 24

Item	Description
Use Case ID	UC-24
Use Case Name	View Pending Bills
Goal	Allow users to view and manage their pending bill payments
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has registered for bill payment services
Post-condition(s)	- Pending bills are displayed to the user
Main Success Scenario	1. User selects "View Pending Bills" option 2. System retrieves pending bill information 3. System displays list of bills with due dates and amounts 4. User can select a bill to pay (optional) 5. System processes payment if selected
Alternative Scenarios	2a. No pending bills: 1. System displays message indicating no pending bills 5a. Payment processing fails: 1. System displays error message 2. User is prompted to try again or use alternative payment method

Use Case 25

Item	Description
Use Case ID	UC-25
Use Case Name	Alert Accounts
Goal	Set up and manage account alerts for various conditions
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has at least one bank account linked
Post-condition(s)	- Alert settings are configured and saved
Main Success Scenario	1. User selects "Alert Accounts" option 2. System displays current alert settings 3. User configures alerts for low balance, large transactions, bill payment reminders 4. User specifies notification method (SMS, email, app notification) 5. System saves alert configuration 6. System displays confirmation message
Alternative Scenarios	3a. User disables all alerts: 1. System displays confirmation prompt 2. User confirms disabling alerts 3. System saves settings

Use Case 26

Item	Description
Use Case ID	UC-26
Use Case Name	Remove Account
Goal	Allow users to remove linked bank accounts from the system
Initiator	Client/Database Administrator
Pre-condition(s)	- User is logged into the system - User has at least one bank account linked
Post-condition(s)	- Selected account is removed from the system
Main Success Scenario	1. User selects "Remove Account" option 2. System displays list of linked accounts 3. User selects account to remove 4. System requests confirmation 5. User confirms removal 6. System removes the account 7. System displays confirmation message
Alternative Scenarios	3a. User attempts to remove last remaining account: 1. System displays warning 2. User must confirm understanding implications 5a. User cancels removal: 1. System returns to account list without making changes

Use Case 27

Item	Description
Use Case ID	UC-27
Use Case Name	Process Money
Goal	Enable users to process various money transactions
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has at least one bank account linked - Account has sufficient funds (for transfers/payments)
Post-condition(s)	- Money transaction is processed successfully
Main Success Scenario	1. User selects "Process Money" option 2. System displays transaction type options 3. User selects desired transaction type 4. System guides user through transaction-specific workflow 5. User enters necessary transaction details 6. System validates information 7. System processes transaction 8. System displays confirmation
Alternative Scenarios	3a. User selects to cancel a transaction (include "Cancel Transaction"): 1. System displays list of pending transactions eligible for cancellation 2. User selects transaction to cancel 3. System requests confirmation 4. User confirms cancellation 5. System cancels the transaction 3b. User selects transfer to wallet (include "Transfer to Wallet"): 1. System displays available wallets and accounts 2. User selects source account and destination wallet 3. User enters transfer amount 4. User enters PIN for authentication (include "Enter PIN") 5. If PIN is invalid (extend "Invalid PIN"), system shows error and prompts for re-entry 6. User enters mobile number for verification (include "Enter Mobile Number") 7. If mobile number is invalid (extend "Invalid Number"), system shows error 8. System processes wallet transfer 3c. User selects transfer to bank account (include "Transfer to Bank Account"): 1. System displays linked bank accounts 2. User selects source and destination accounts 3. User transfers credit (include "Transfer Credit") or receives money (include "Receive Money") 4. System processes bank transfer 3d. User selects ATM processing (include "Process ATM"): 1. System displays ATM transaction options 2. User enters card number (include "Enter Card Number") 3. If card number is invalid (extend "Invalid Card"), system shows error 4. System handles withdrawal management (include "Manage Withdrawing") 5. System sends SMS confirmation (include "Send SMS") 3e. User selects bill payment (include "Paying Bills"): 1. System displays saved billers and option to add new biller 2. User selects biller and enters bill details 3. System processes bill payment 3f. User selects international transfer (include "Transfer to Foreign Account"): 1. System displays international transfer form 2. User enters recipient's international bank details 3. System deducts transaction fee (include "Deduct Transaction Fee") 4. System verifies exchange rate (include "Verify Exchange Rate") 5. System performs currency conversion (include "Currency Conversion") 6. System processes international

Item	Description
	transfer 6a. Invalid transaction details: 1. System displays error message 2. User corrects information and resubmits

Use Case 28

Item	Description
Use Case ID	UC-28
Use Case Name	Last Five Transaction
Goal	Display the five most recent transactions for quick reference
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has transaction history
Post-condition(s)	- Last five transactions are displayed to the user
Main Success Scenario	1. User selects "Last Five Transaction" option 2. System retrieves the five most recent transactions from the database 3. System displays transactions with date, amount, type, and recipient/sender information
Alternative Scenarios	2a. User has fewer than five transactions: 1. System displays all available transactions 2b. User has no transactions: 1. System displays message indicating no transaction history is available

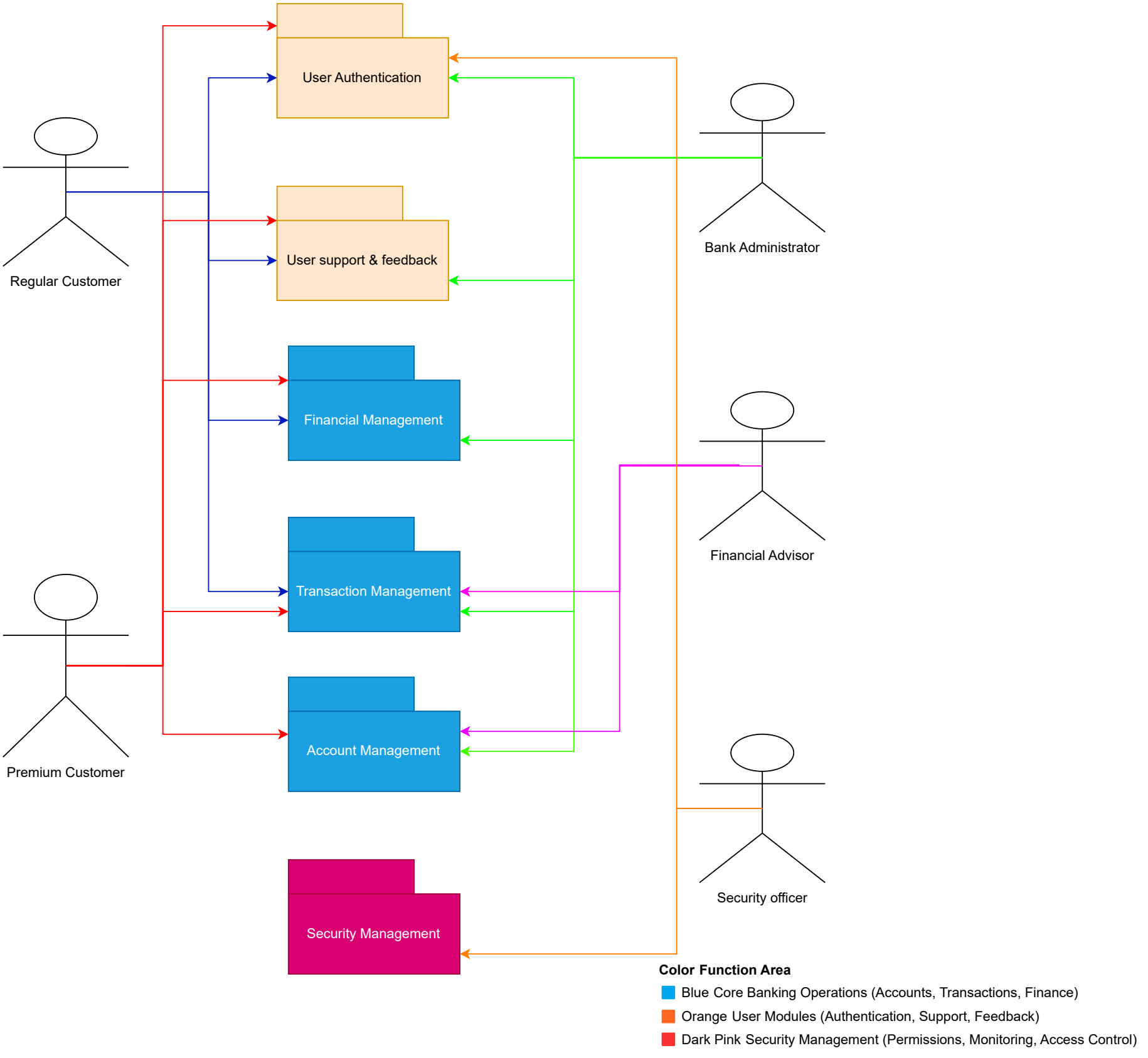
Use Case 29

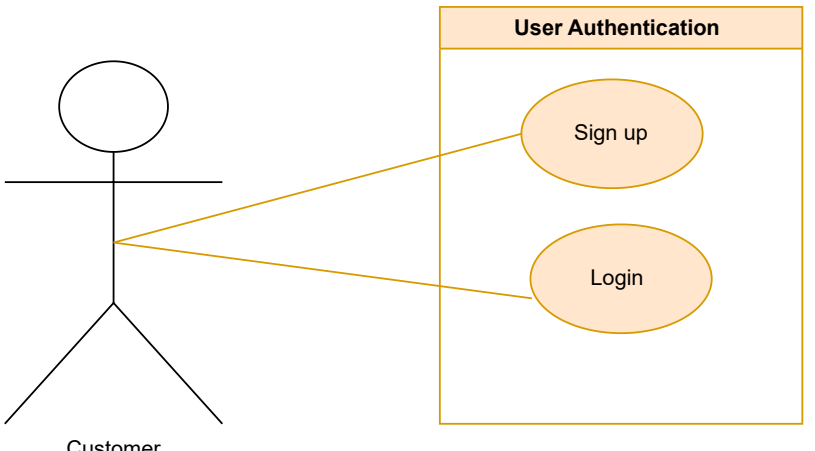
Item	Description
Use Case ID	UC-29
Use Case Name	Activate Offers
Goal	Allow users to activate special offers and promotions
Initiator	Client
Pre-condition(s)	- User is logged into the system - Special offers are available for the user
Post-condition(s)	- Selected offer is activated for the user's account
Main Success Scenario	1. User selects "Activate Offers" option 2. System displays list of available offers 3. User selects specific offer to activate 4. System confirms eligibility 5. System activates the offer 6. System displays confirmation message
Alternative Scenarios	2a. No offers available: 1. System displays message that no offers are currently available 4a. User not eligible for selected offer: 1. System displays message explaining eligibility requirements 2. User is returned to offers list

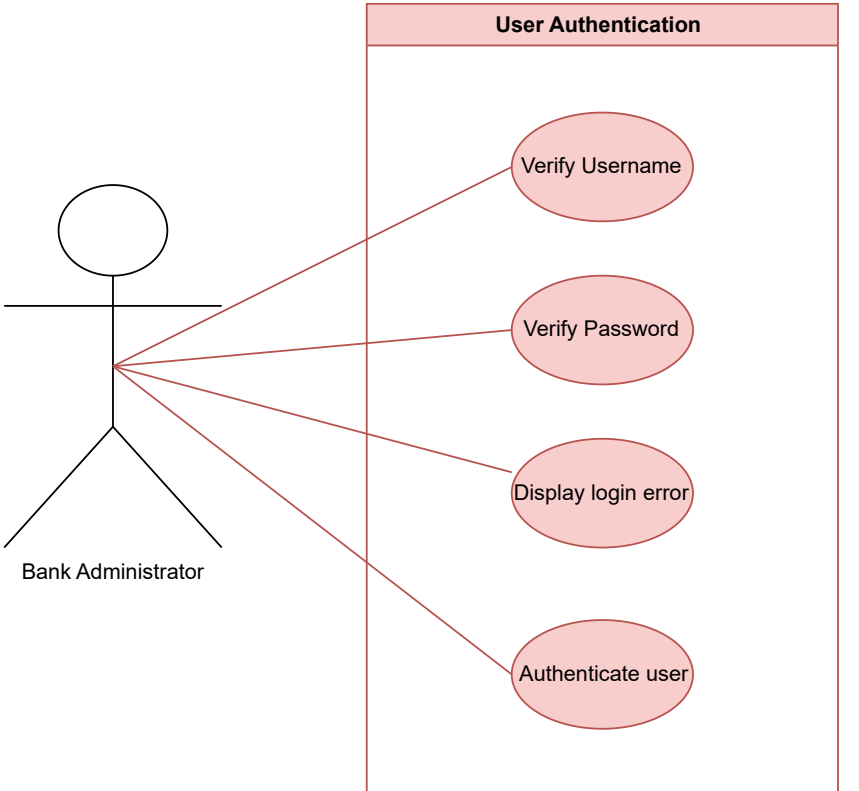
Use Case 30

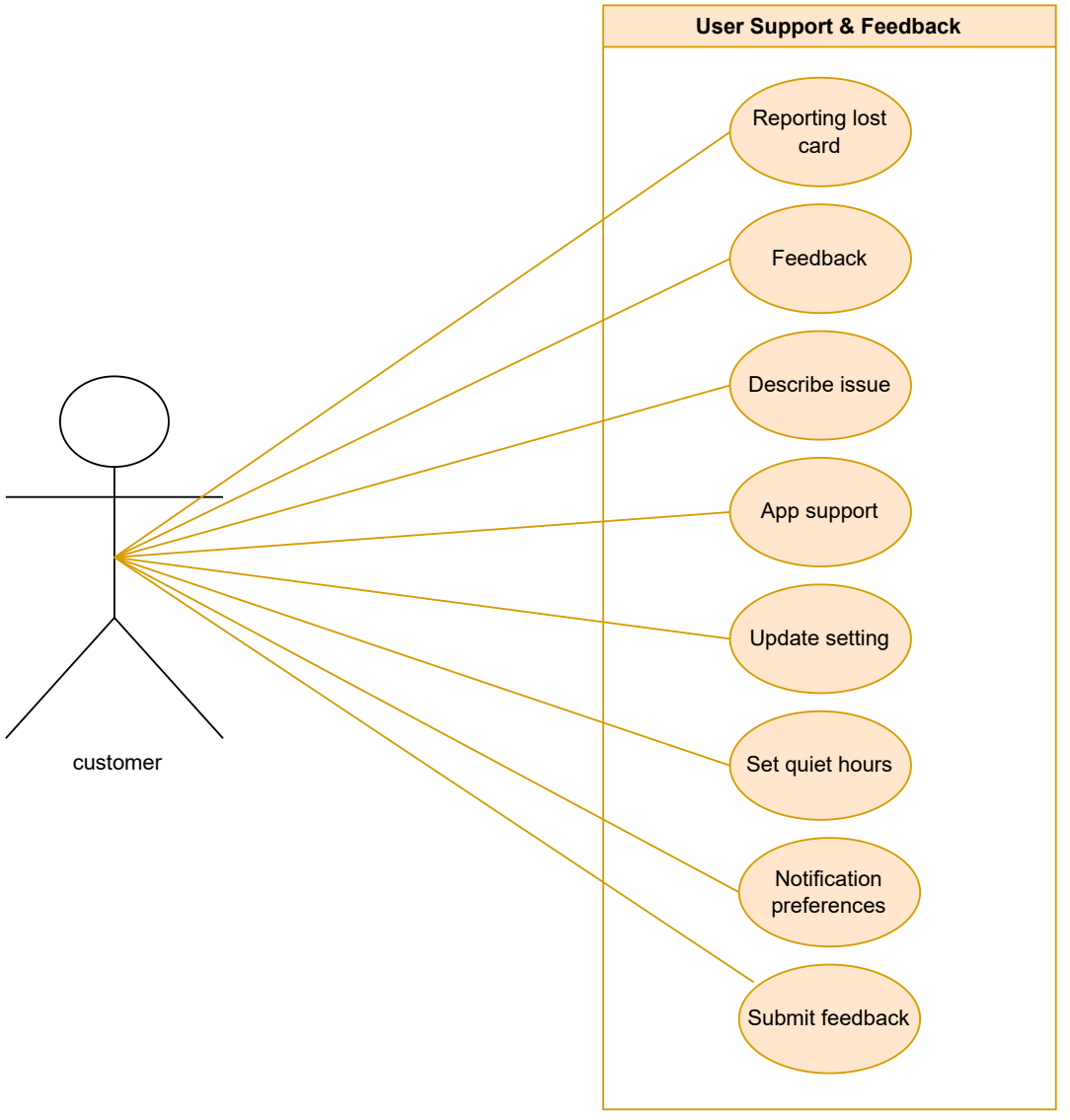
Item	Description
Use Case ID	UC-30
Use Case Name	Rewards Points
Goal	Allow users to view and redeem accumulated rewards points
Initiator	Client
Pre-condition(s)	- User is logged into the system - User has rewards points program enabled
Post-condition(s)	- User can view points balance and redemption options
Main Success Scenario	1. User selects "Rewards Points" option 2. System retrieves current points balance 3. System displays points balance and available redemption options 4. User selects redemption option (optional) 5. System processes redemption 6. System updates points balance and displays confirmation
Alternative Scenarios	2a. User has no rewards points: 1. System displays message explaining how to earn points 5a. Redemption unsuccessful: 1. System displays error message 2. User is returned to redemption options

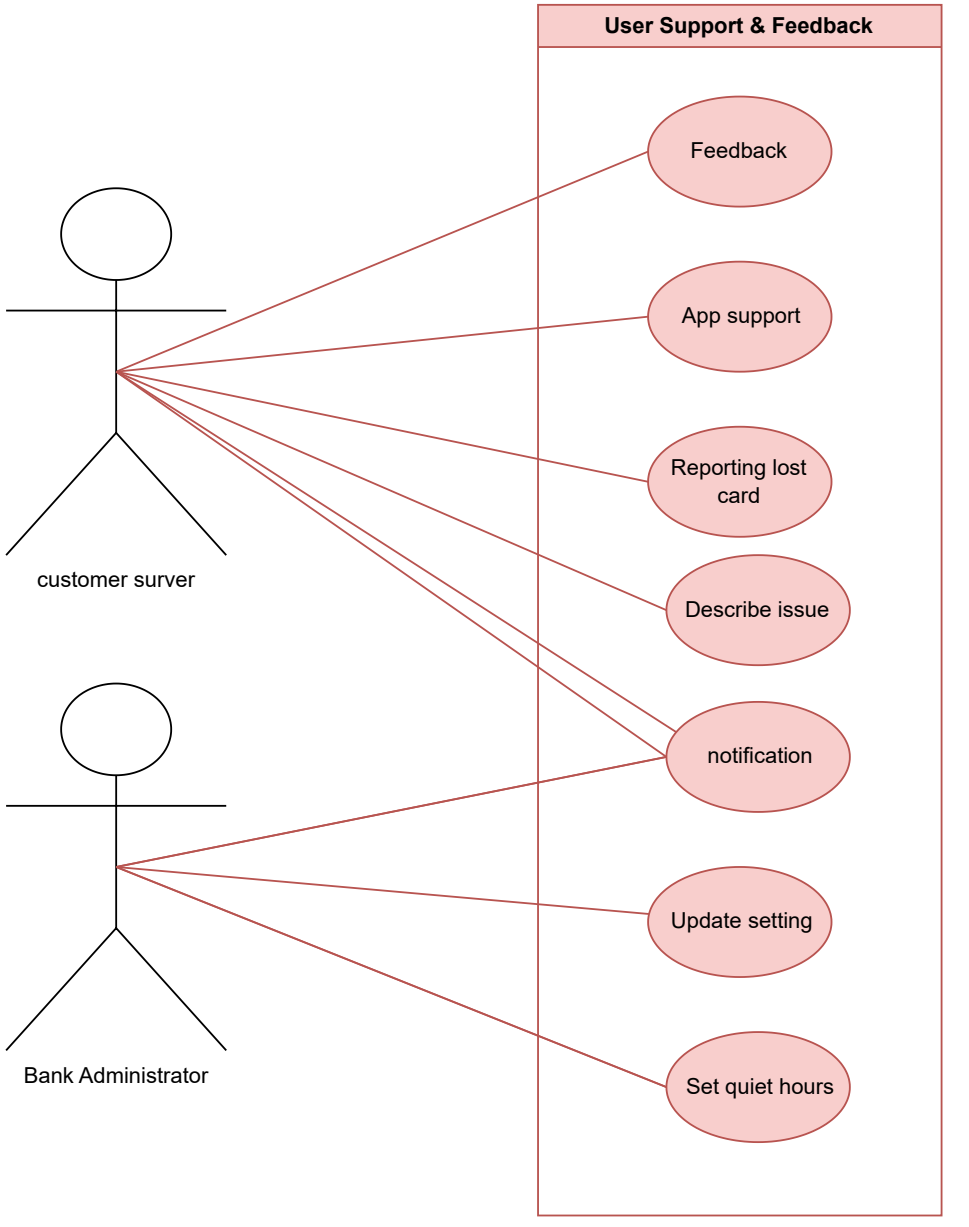
Package Diagram :

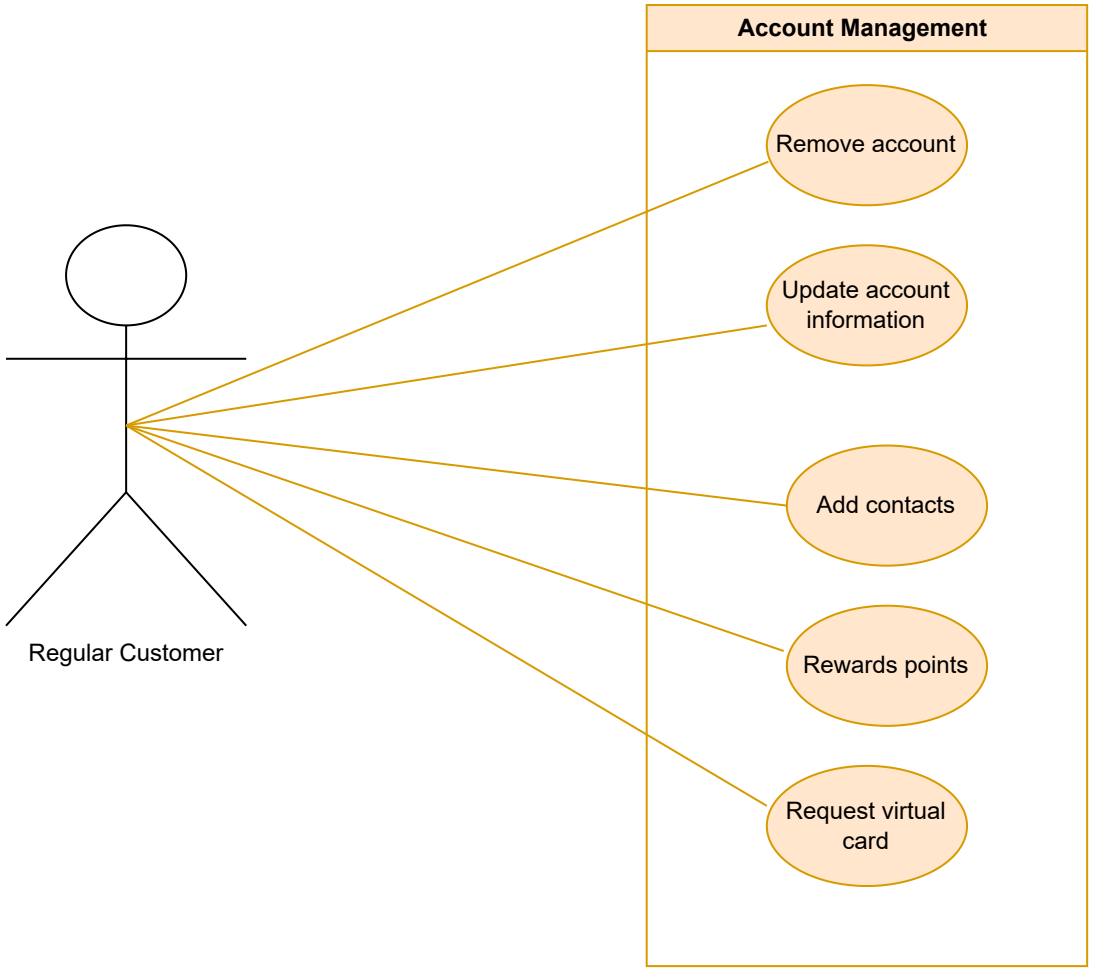

User Authentication
User support & feedback
Financial Management
Transaction Management
Account Management
Security Management
Regular Customer
Premium Customer
Bank Administrator
Financial Advisor
Security officer
Color Function Area
Blue Core Banking Operations (Accounts, Transactions, Finance)
Orange User Modules (Authentication, Support, Feedback)
Dark Pink Security Management (Permissions, Monitoring, Access Control)

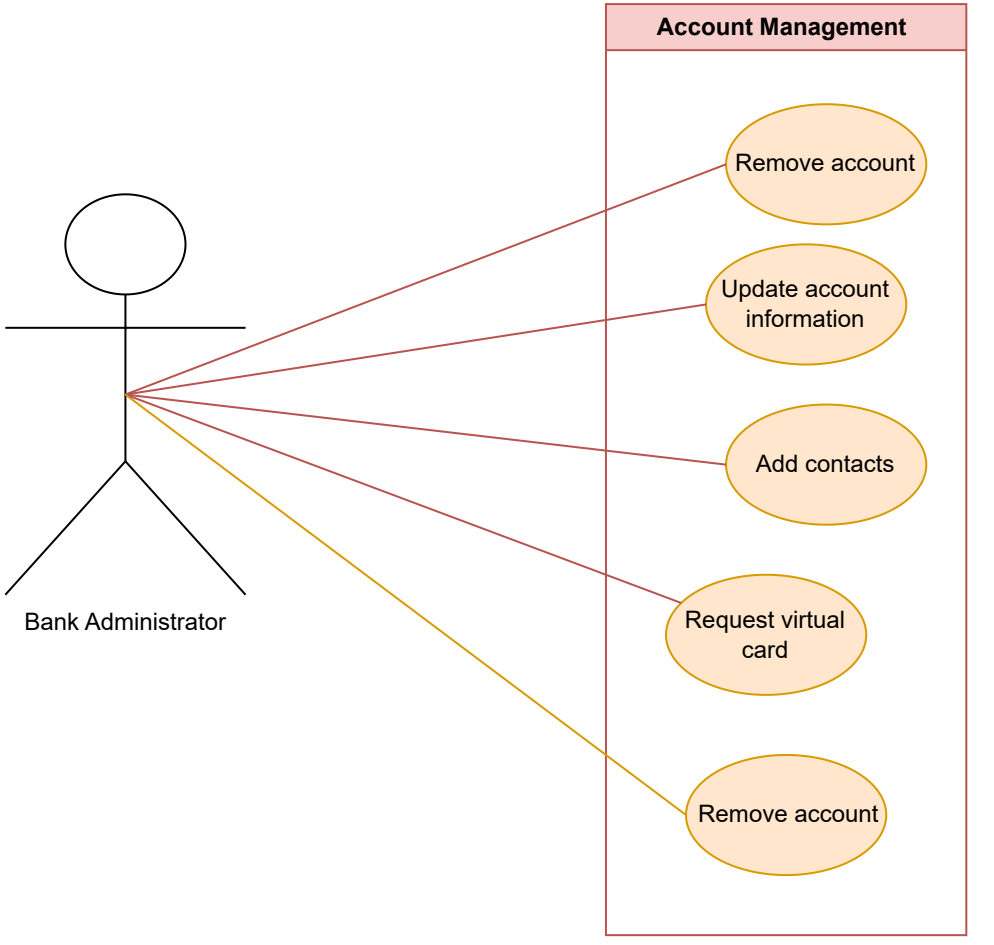

User Authentication
Sign up
Login
Customer

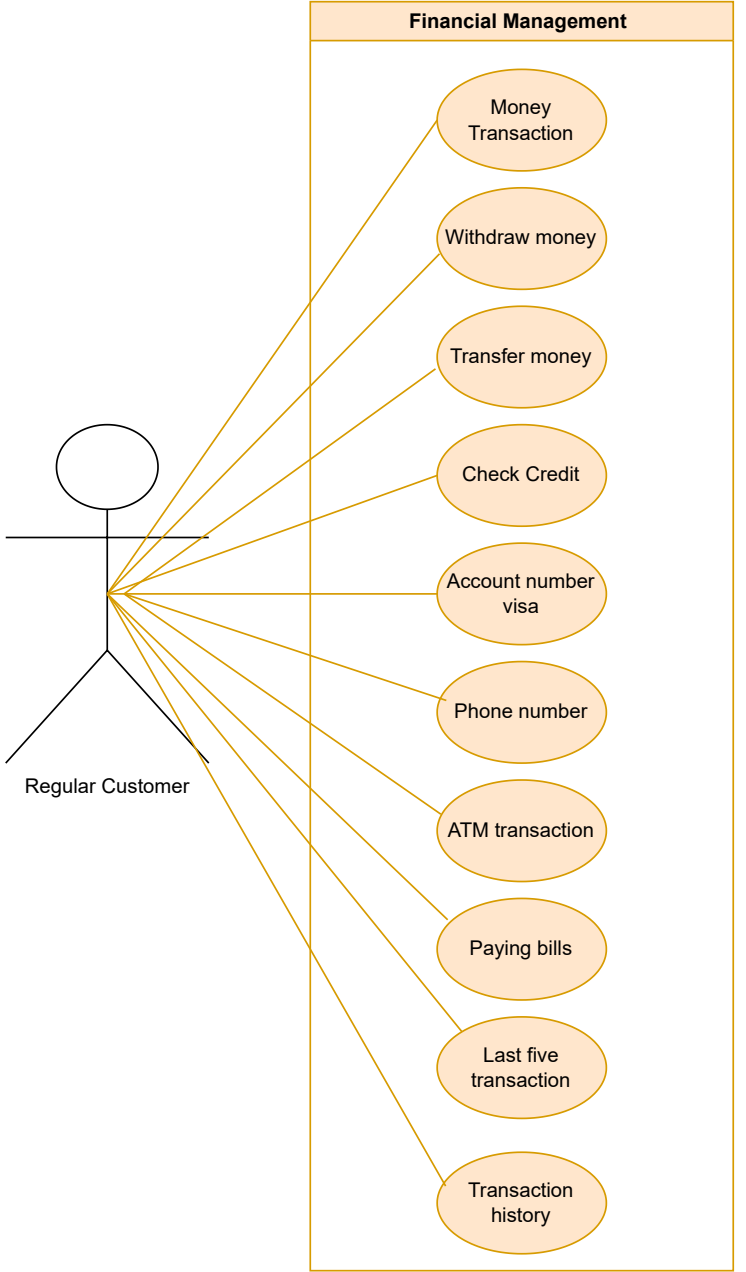

User Authentication
Verify Username
Verify Password
Display login error
Authenticate user
Bank Administrator

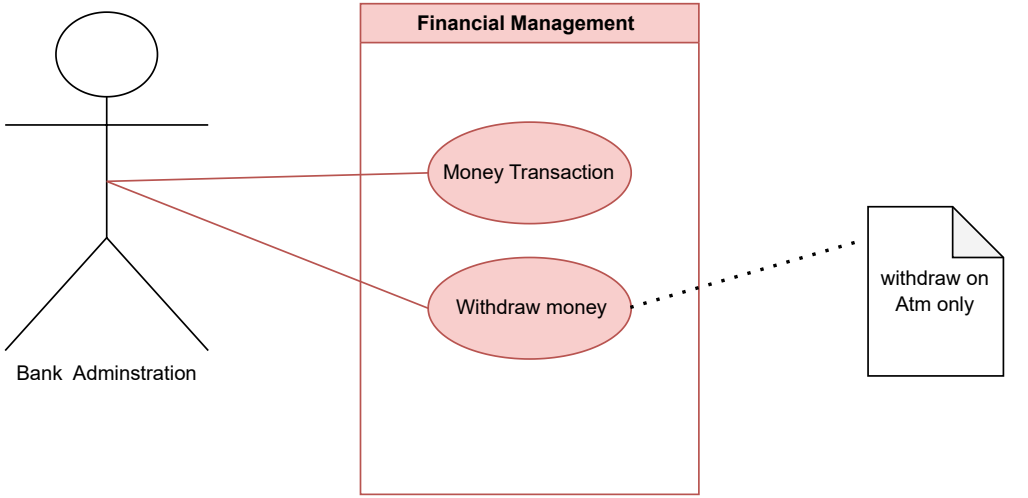

User Support & Feedback
customer
Reporting lost card
Feedback
Describe issue
App support
Update setting
Set quiet hours
Notification preferences
Submit feedback

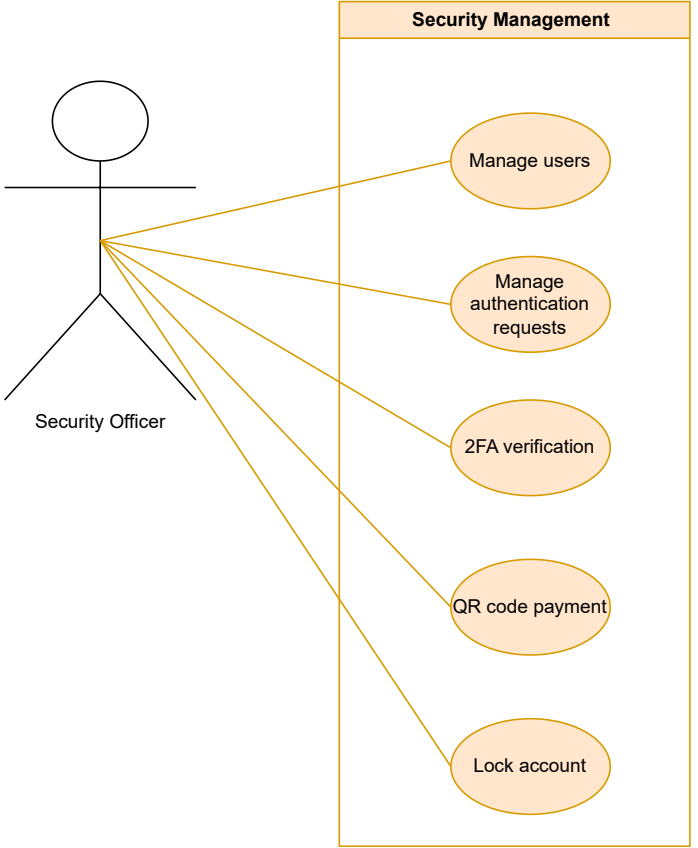

User Support & Feedback
customer surver
Bank Administrator
Feedback
App support
Reporting lost card
Describe issue
notification
Update setting
Set quiet hours

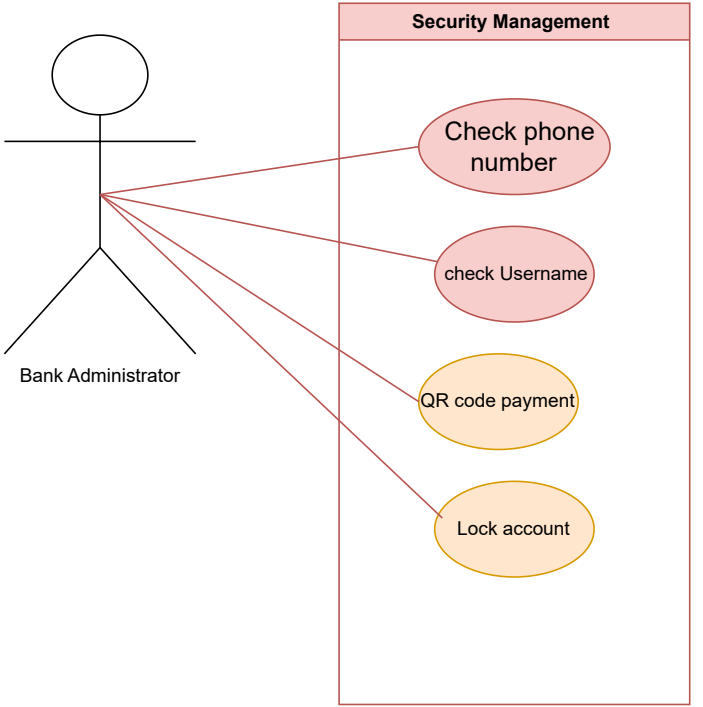

Account Management
Remove account
Update account information
Add contacts
Rewards points
Request virtual card
Regular Customer

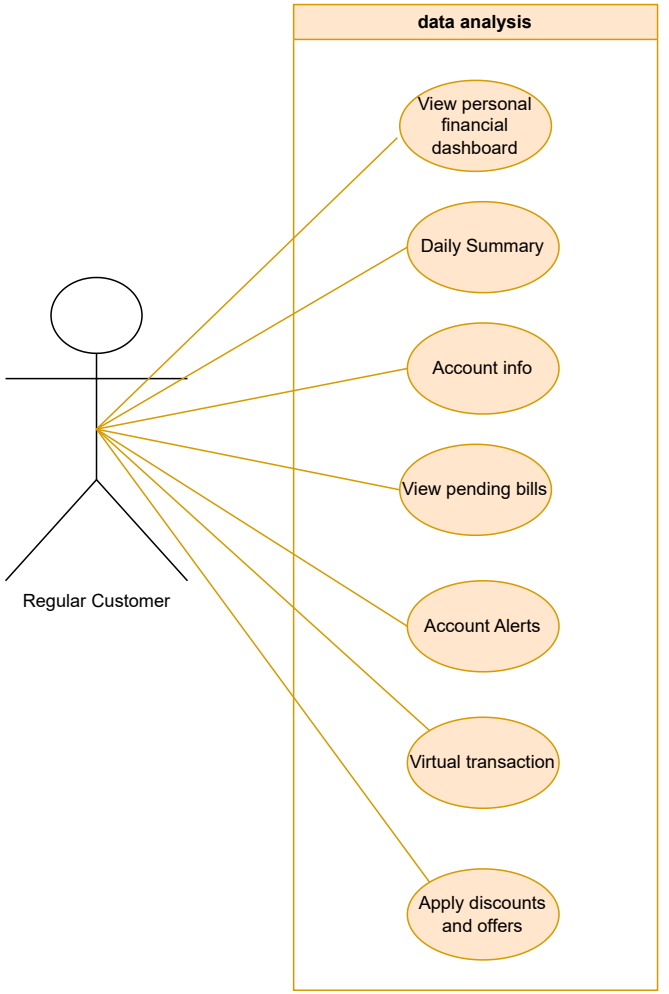

Account Management
Remove account
Update account information
Add contacts
Request virtual card
Remove account
Bank Administrator

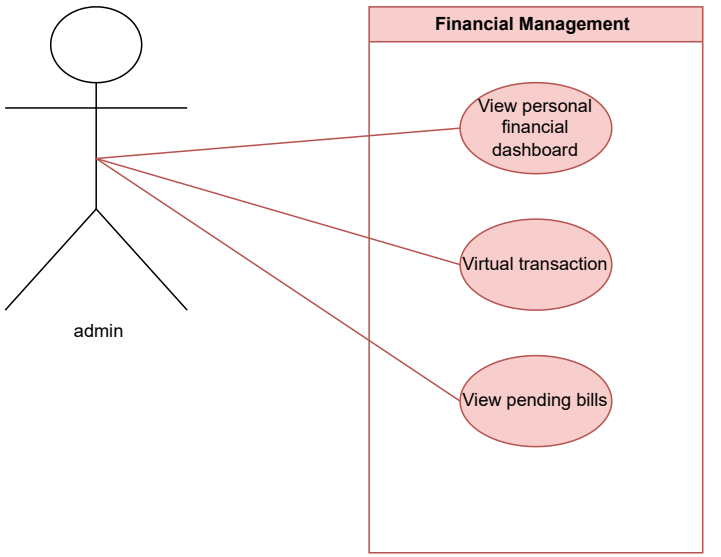

Financial Management
Money Transaction
Withdraw money
Transfer money
Check Credit
Account number visa
Phone number
ATM transaction
Paying bills
Last five transaction
Transaction history
Regular Customer

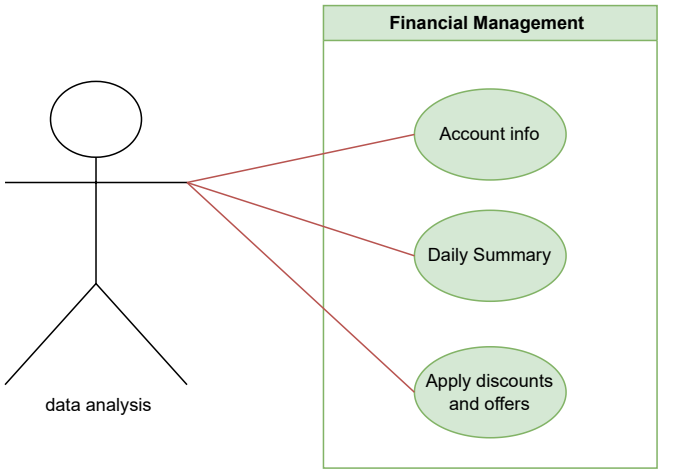

Financial Management
Money Transaction
Withdraw money
Bank Adminstration
withdraw on Atm only


Security Management
Security Officer
Manage users
Manage authentication requests
2FA verification
QR code payment
Lock account


Security Management
Bank Administrator
Check phone number
check Username
QR code payment
Lock account



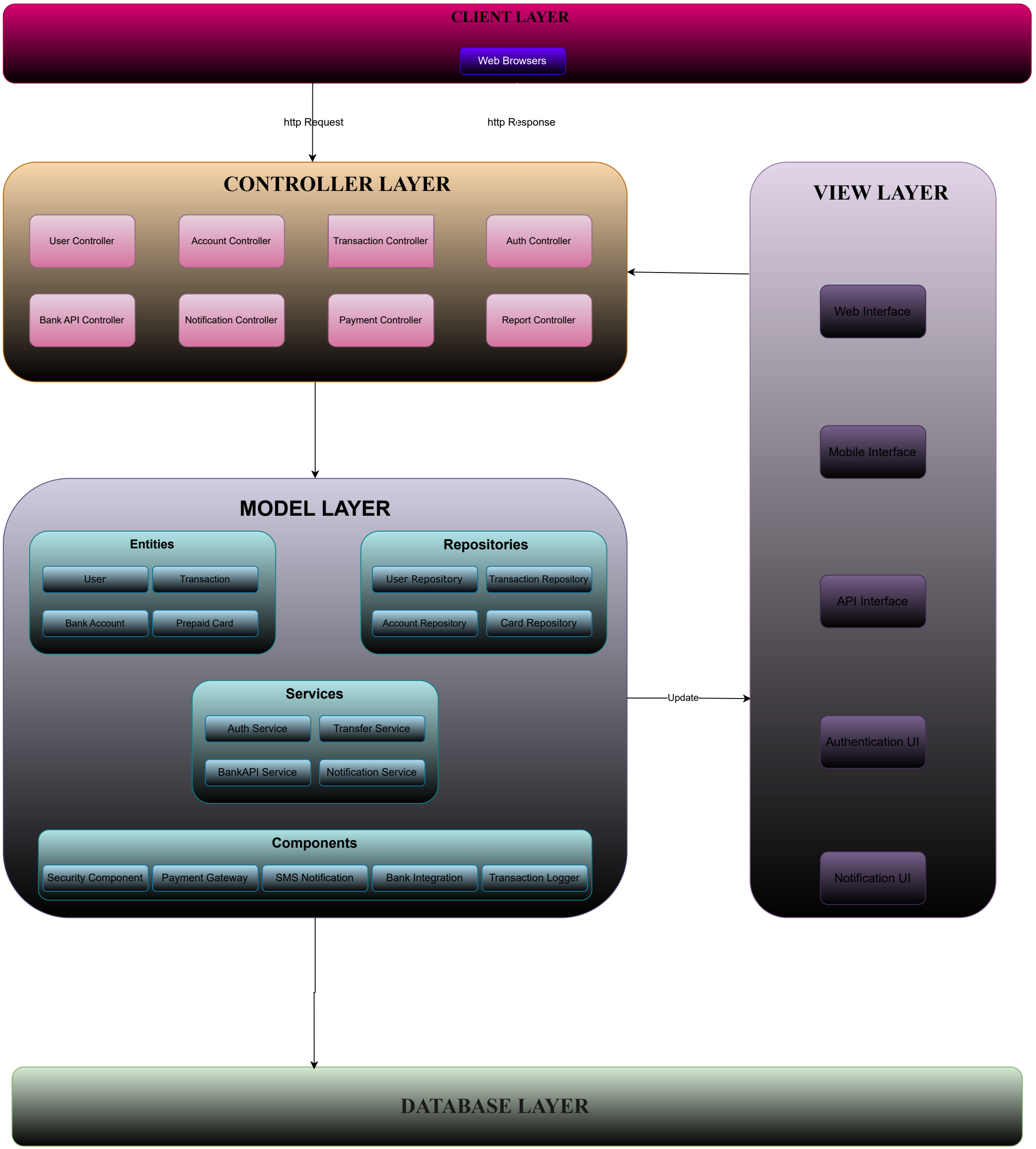




9) Structural & Behavioural Diagrams :

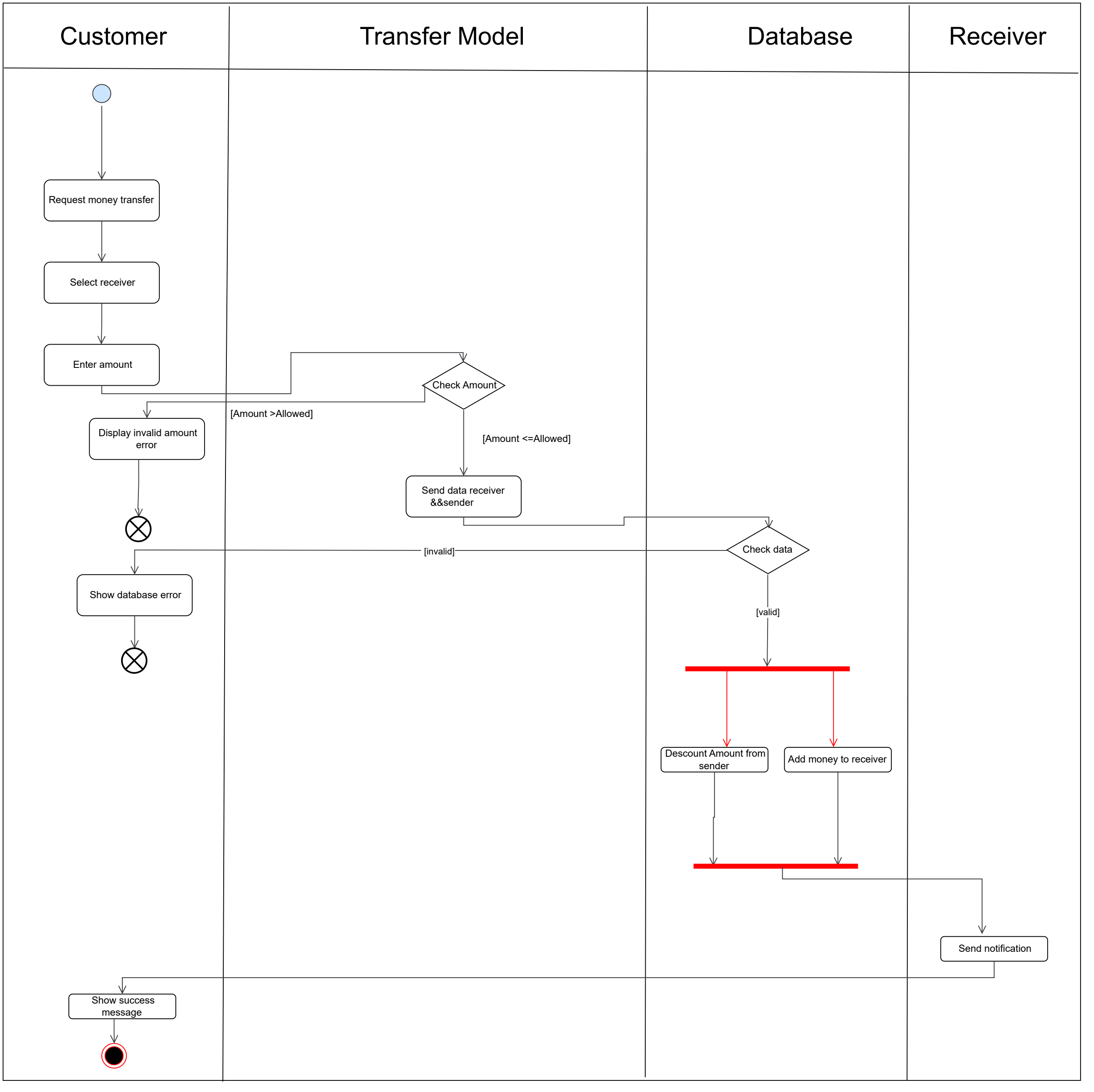
System Architecture :

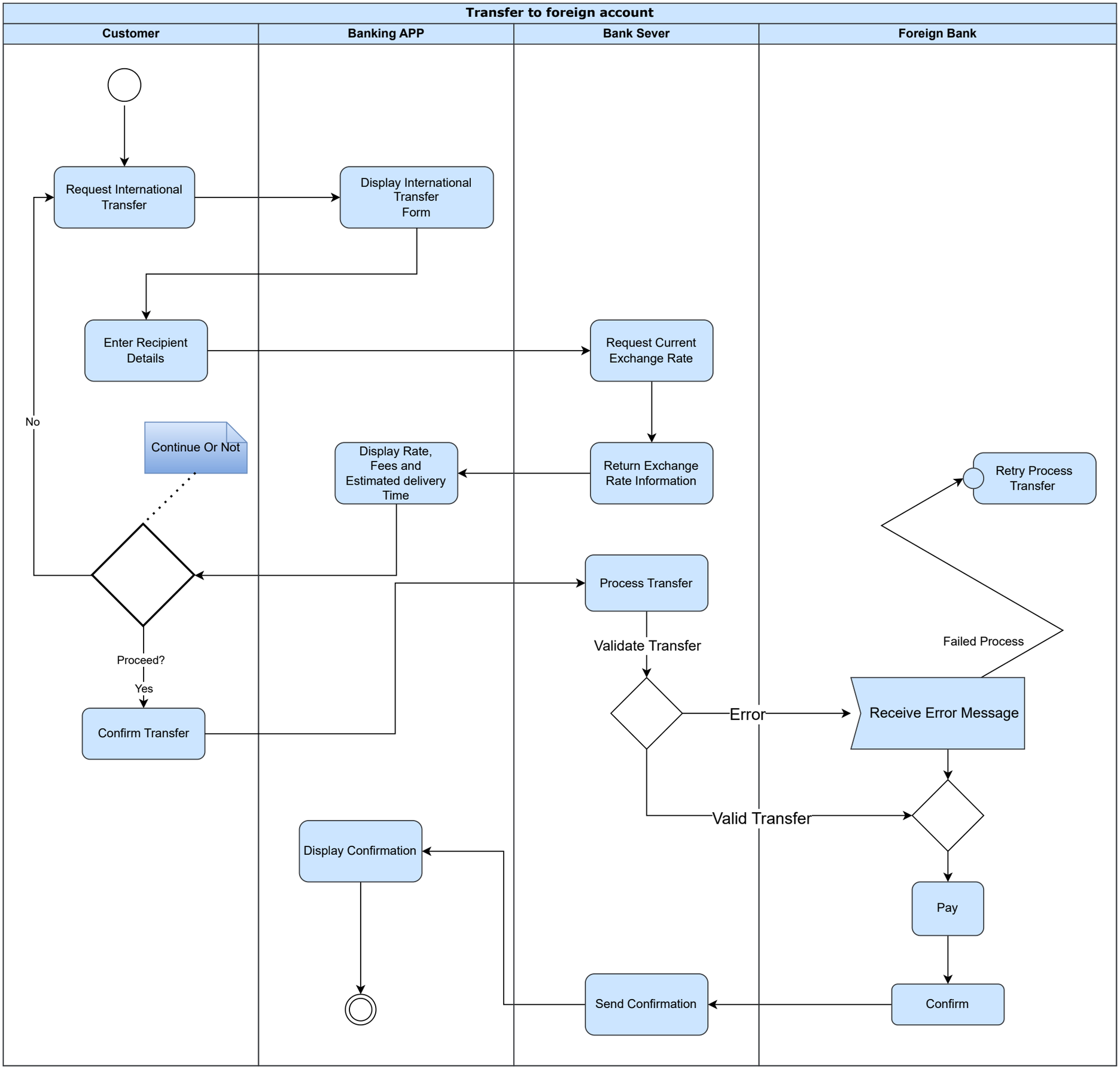
Instant Money Transfers and Bank-Accounts Access 24/7 MVC Architecture



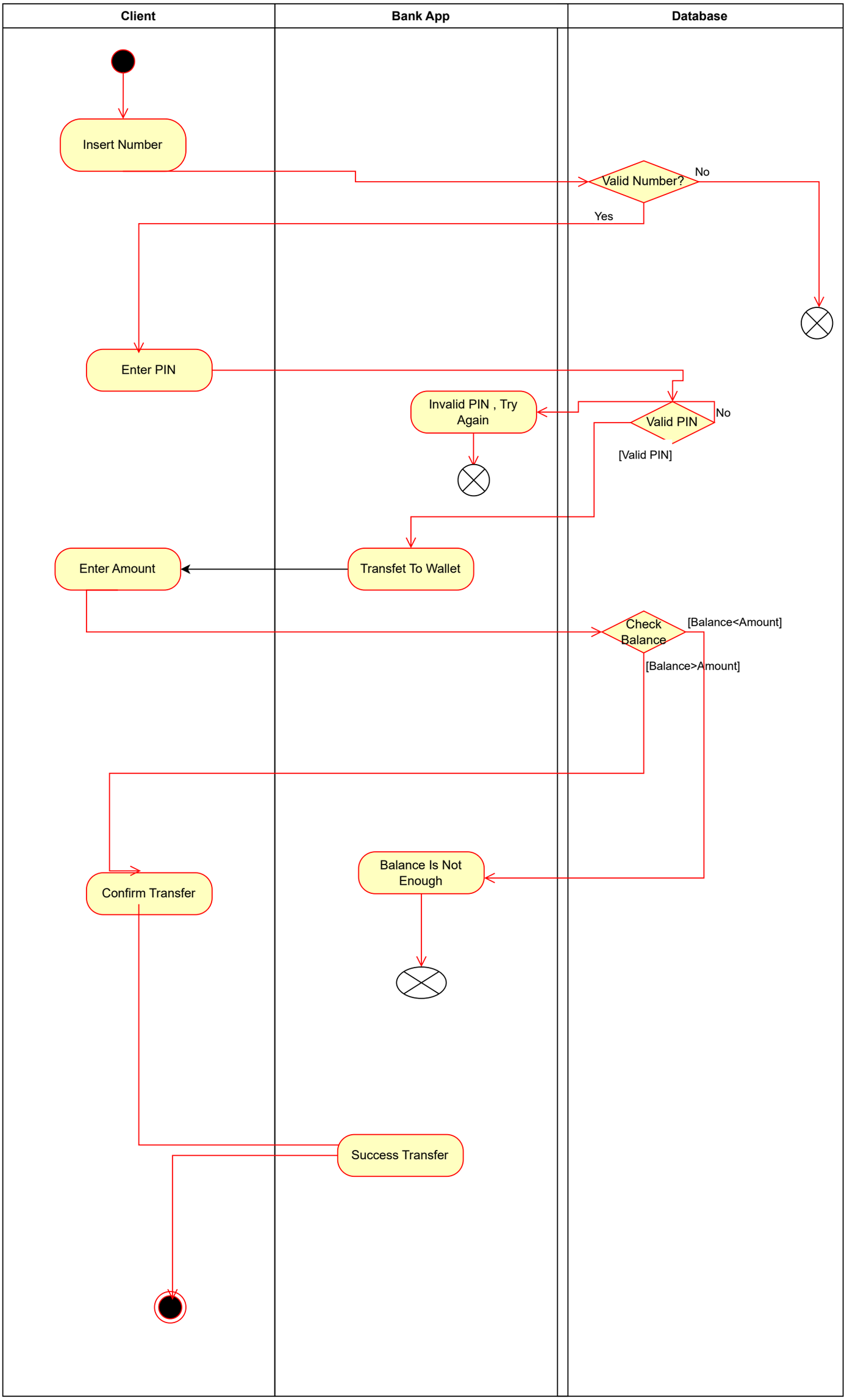
Activity diagrams :

Transaction to bank account

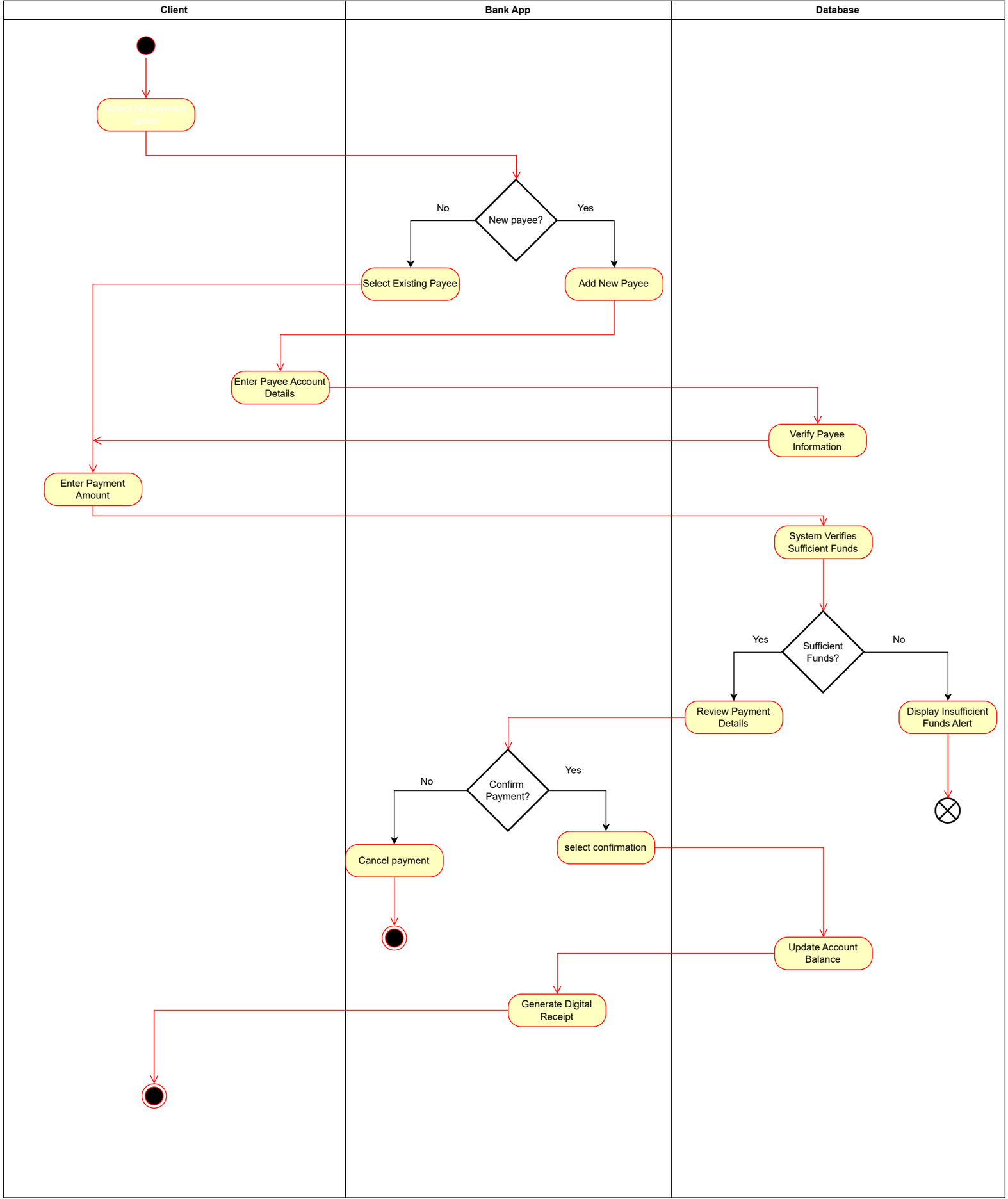



Transfer to foreign account
Customer
Banking APP
Bank Sever
Foreign Bank
Request International Transfer
Display International Transfer Form
Enter Recipient Details
Request Current Exchange Rate
No
Continue Or Not
Display Rate, Fees and Estimated delivery Time
Return Exchange Rate Information
Retry Process Transfer
Process Transfer
Proceed?
Yes
Validate Transfer
Failed Process
Confirm Transfer
Error
Receive Error Message
Valid Transfer
Display Confirmation
Pay
Send Confirmation
Confirm

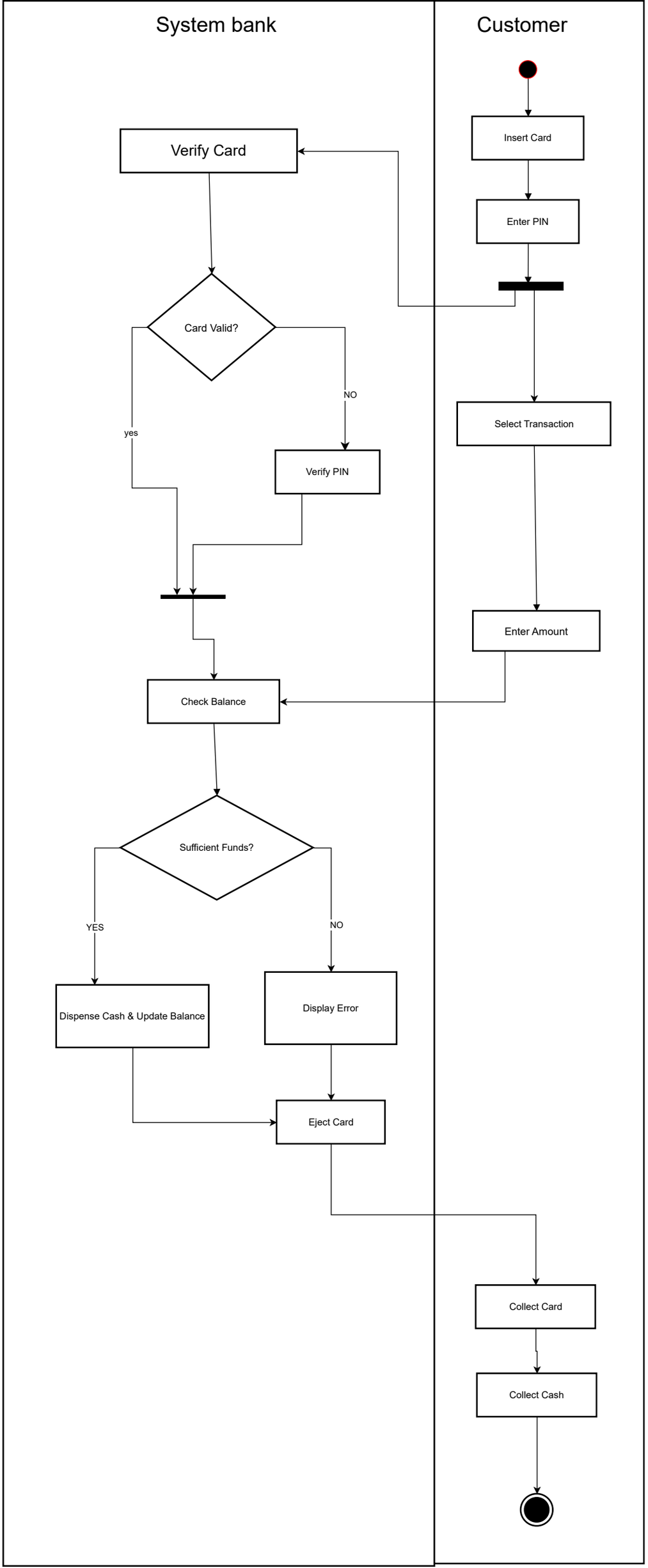
TRANSFER TO WALLET

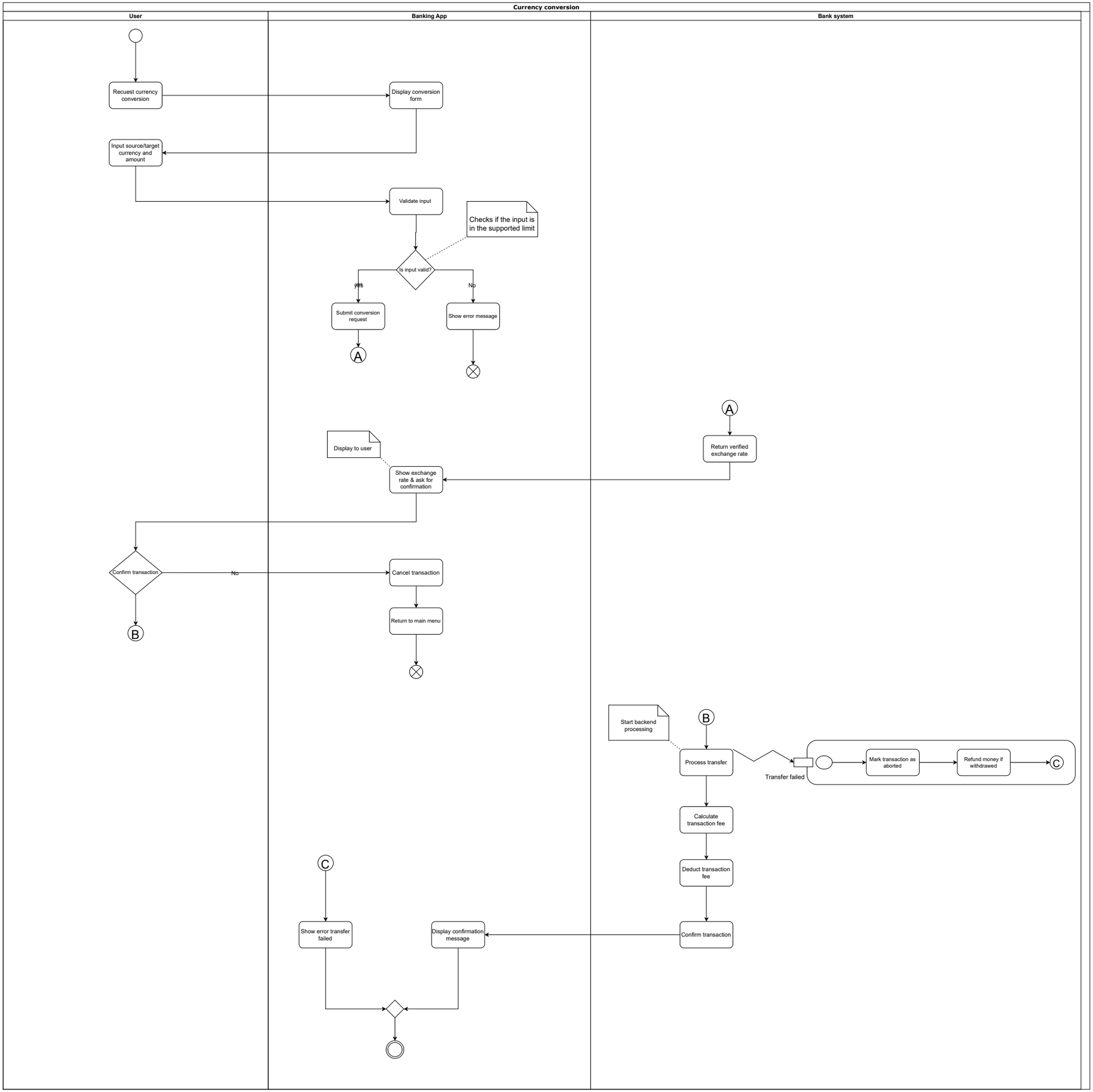


Paying bills



ATM


System bank
Customer
Insert Card
Enter PIN
Verify Card
Card Valid?
NO
yes
Verify PIN
Select Transaction
Enter Amount
Check Balance
Sufficient Funds?
YES
NO
Dispense Cash & Update Balance
Display Error
Eject Card
Collect Card
Collect Cash


Currency conversion
User
Banking App
Bank system
Recuest currency conversion
Display conversion form
Input source/target currency and amount
Validate input
Checks if the input is in the supported limit
Is input valid?
yes
No
Submit conversion request
Show error message
A
A
Return verified exchange rate
Display to user
Show exchange rate & ask for confirmation
Confirm transaction
No
Cancel transaction
Return to main menu
B
Start backend processing
B
Process transfer
Transfer failed
Mark transaction as aborted
Refund money if withdrawed
C
Calculate transaction fee
Deduct transaction fee
Confirm transaction
C
Show error transfer failed
Display confirmation message

List of User Interfaces Required for the Banking System

1. Client Interfaces

Authentication Interfaces

1. **Login Interface**
   - Username and password input fields
   - Login button
   - Link to sign up page
   - Error message display area
2. **Sign Up Interface**
   - Account information input fields (username, password)
   - Personal information input fields
   - Phone number verification
   - Submit button
3. **PIN Entry Interface**
   - Numeric PIN pad
   - Submit button
   - Error message display

Money Transfer Interfaces

4. **Money Transfer Dashboard**
   - Transfer type selection (bank account, wallet, foreign account)
   - Recent recipients list
   - Quick transfer options
5. **Bank Account Transfer Interface**
   - Recipient selection/input
   - Amount input
   - Transfer details confirmation
   - Success/failure message display
6. **Wallet Transfer Interface**
   - Wallet number input

- PIN verification
- Amount input
- Balance check display
- Confirmation screen
- Success/failure message display

7. **Foreign Account Transfer Interface**
    - Recipient details input form
    - Exchange rate and fees display
    - Estimated delivery time information
    - Confirmation screen
    - Success/failure notification

Bill Payment Interfaces

8. **Bill Payment Interface**
    - Payee selection (existing/new)
    - Payment amount input
    - Payee account details input (for new payees)
    - Payment confirmation screen
    - Digital receipt generation
    - Success/failure notification

Currency Conversion Interface

9. **Currency Conversion Interface**
    - Source/target currency selection
    - Amount input
    - Exchange rate display
    - Fees information
    - Confirmation screen
    - Success/failure notification

Account Management Interfaces

10. **Account Dashboard**
    - Account summary

- Balance information
- Recent transactions
- Quick action buttons (transfer, bill payment, etc.)

11. **Personal Financial Dashboard**
    - Financial analytics visualizations
    - Last five transactions
    - Pending bills view
    - Rewards points display
    - Offers and promotions
12. **Card Management Interface**
    - Virtual card request form
    - Card details display
    - Card status (active/inactive)
    - Report lost card option
13. **Contact Management Interface**
    - Add/edit/remove contact form
    - Contacts list view
    - Recent transfers to contacts
14. **Notification Preferences Interface**
    - Notification types toggles
    - Quiet hours setting
    - Update notification settings form
15. **Feedback and Support Interface**
    - Issue reporting form
    - Lost card reporting form
    - Support ticket status view
    - App support contact options

2. Admin Interfaces

16. **Admin Dashboard**
    - System overview metrics

- Quick access to management functions
- Alert notifications

17. **Customer Management Interface**
    - Customer search and filtering
    - Customer information display
    - Account status management
    - Add/update/remove customer accounts
    - Add visa services to accounts

3. Operations Manager Interfaces

18. **Transaction Management Interface**
    - Transaction monitoring dashboard
    - Cancel transaction functionality
    - Transaction verification tools

19. **Exchange Rate Management Interface**
    - Exchange rate verification
    - Rate update interface
    - Transaction fee management

4. Database Administrator Interfaces

20. **Data Analysis Dashboard**
    - User activity metrics
    - Daily transaction summaries
    - Top active users reports

21. **Account Alert Management Interface**
    - Alert configuration for account activities
    - Warning threshold settings (amount/time)
    - Low balance alert configuration

22. **Security Management Interface**
    - 2FA verification settings
    - Account lock management
    - Security breach monitoring

- QR code payment verification settings

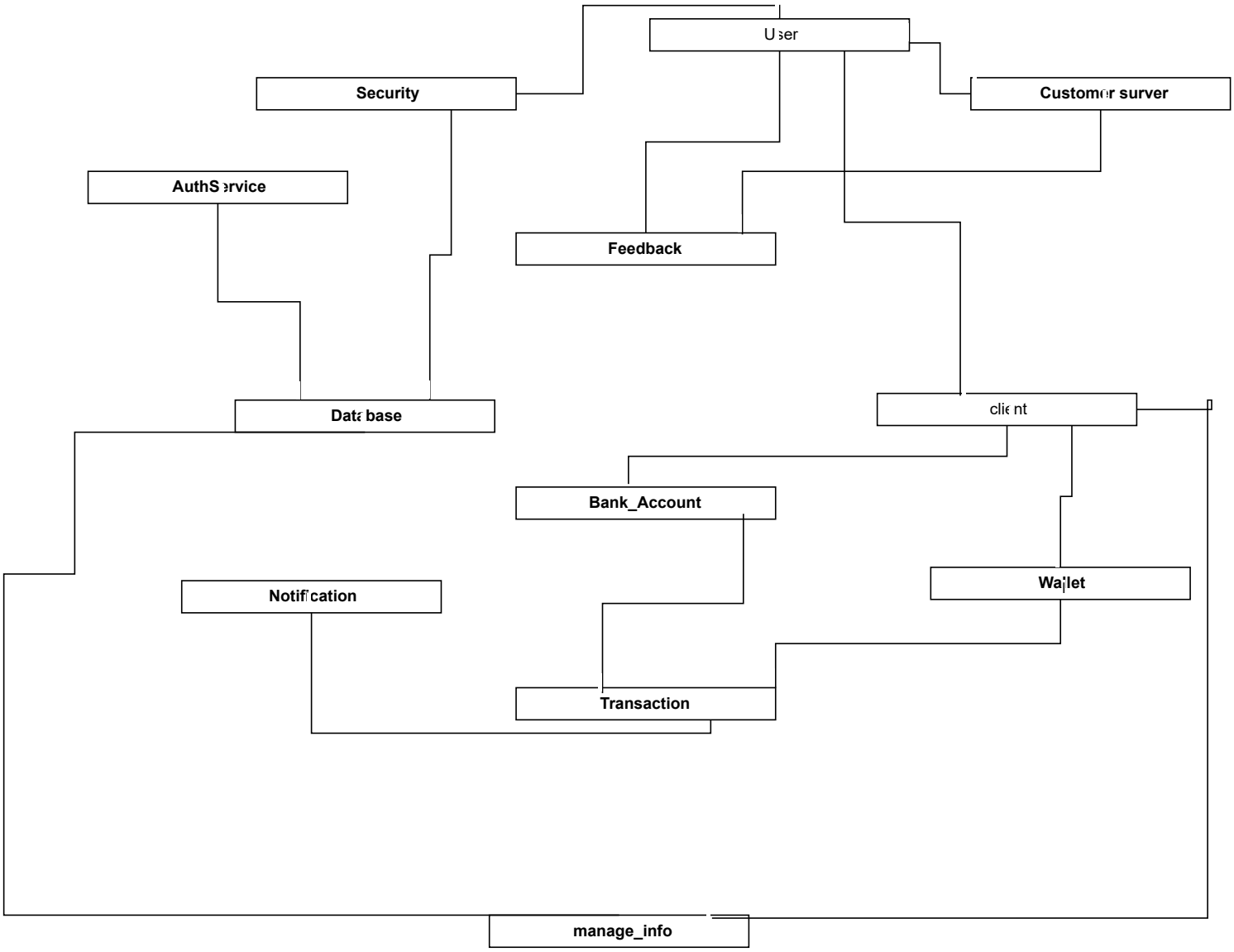
5. ATM Interfaces

23. **ATM User Interface**
    - Card insertion detection
    - PIN entry screen
    - Transaction type selection
    - Amount entry screen
    - Receipt printing option
    - Success/error messages

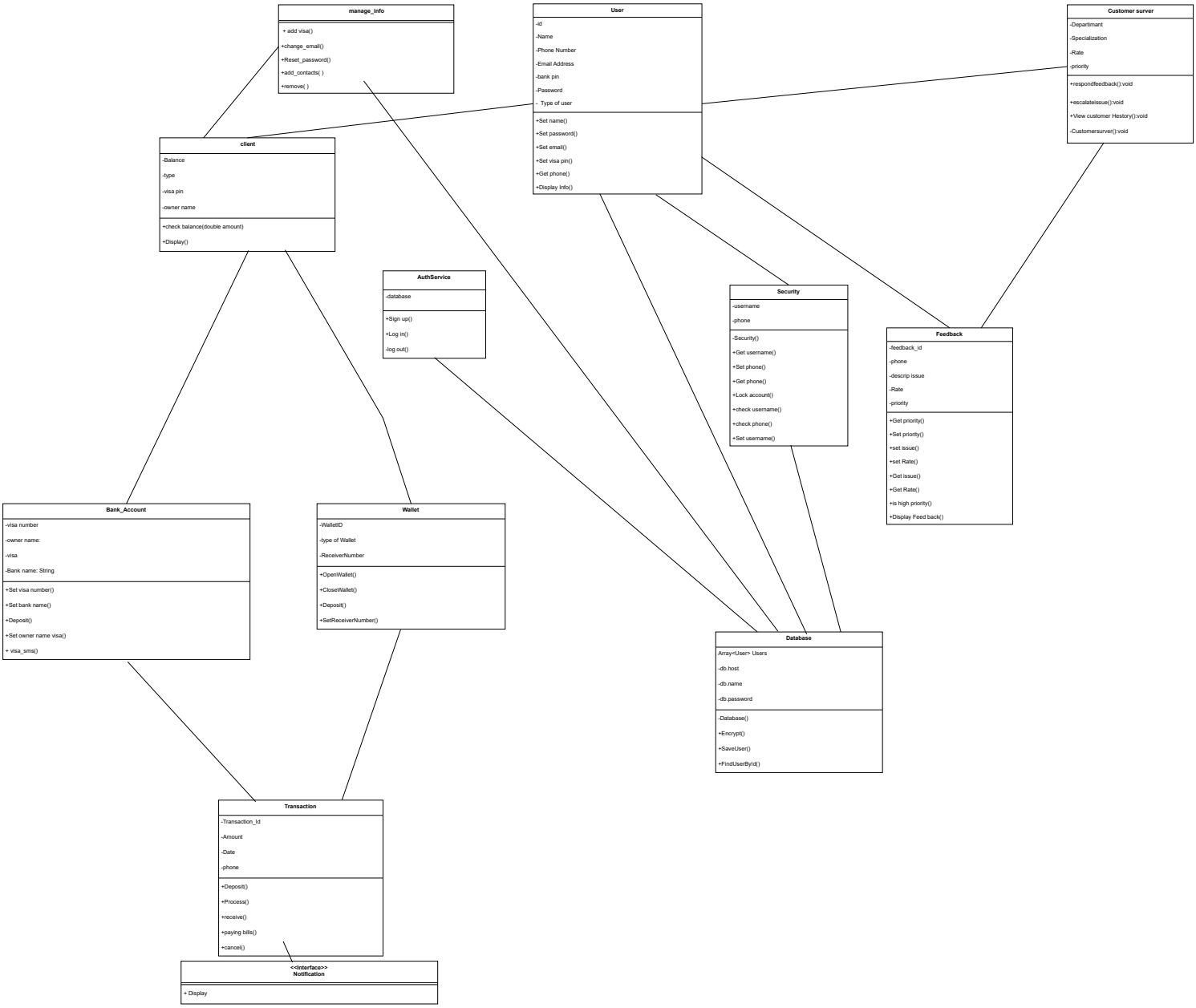
6. Bank System Interfaces (Backend)

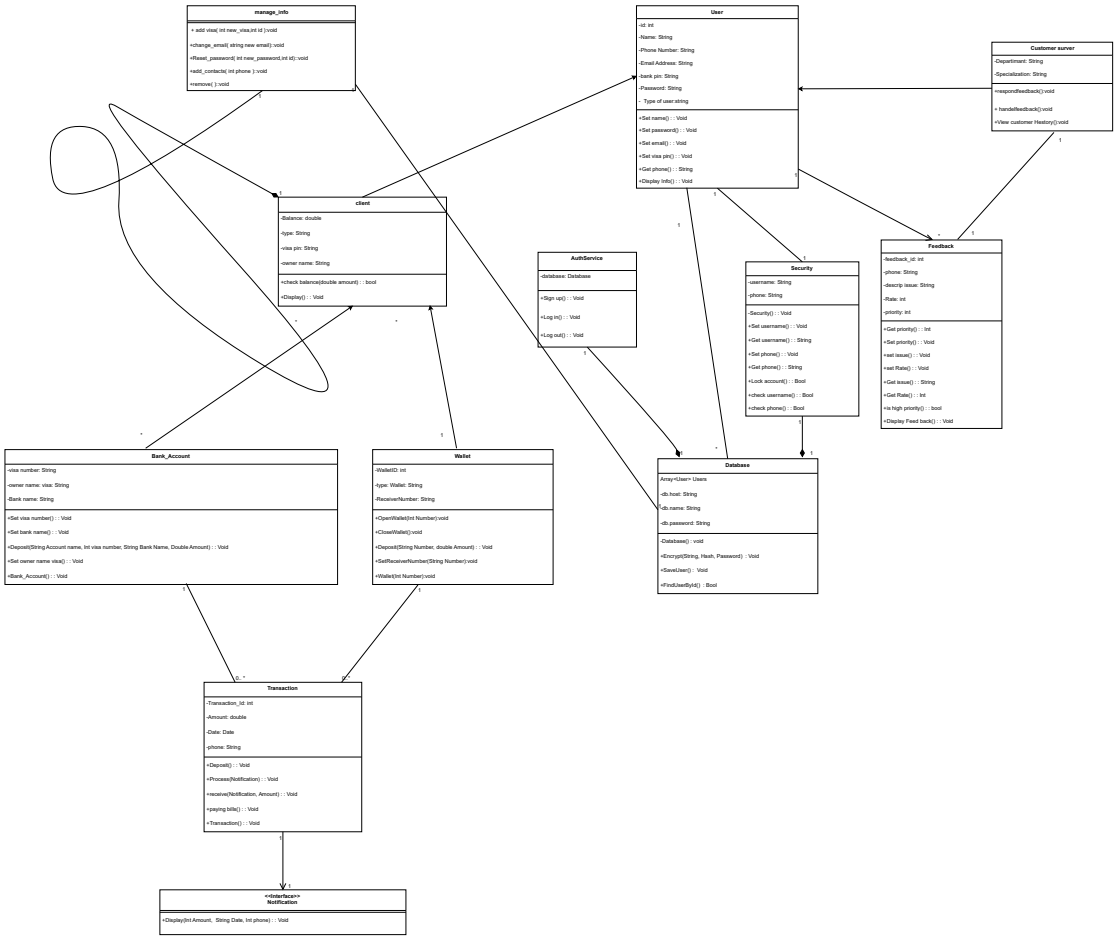
24. **Transaction Processing Interface**
    - Transaction validation tools
    - Fund transfer processing
    - Fee calculation management
    - Error handling and reporting
25. **Notification Management System**
    - SMS notification configuration
    - Email notification settings
    - Push notification management

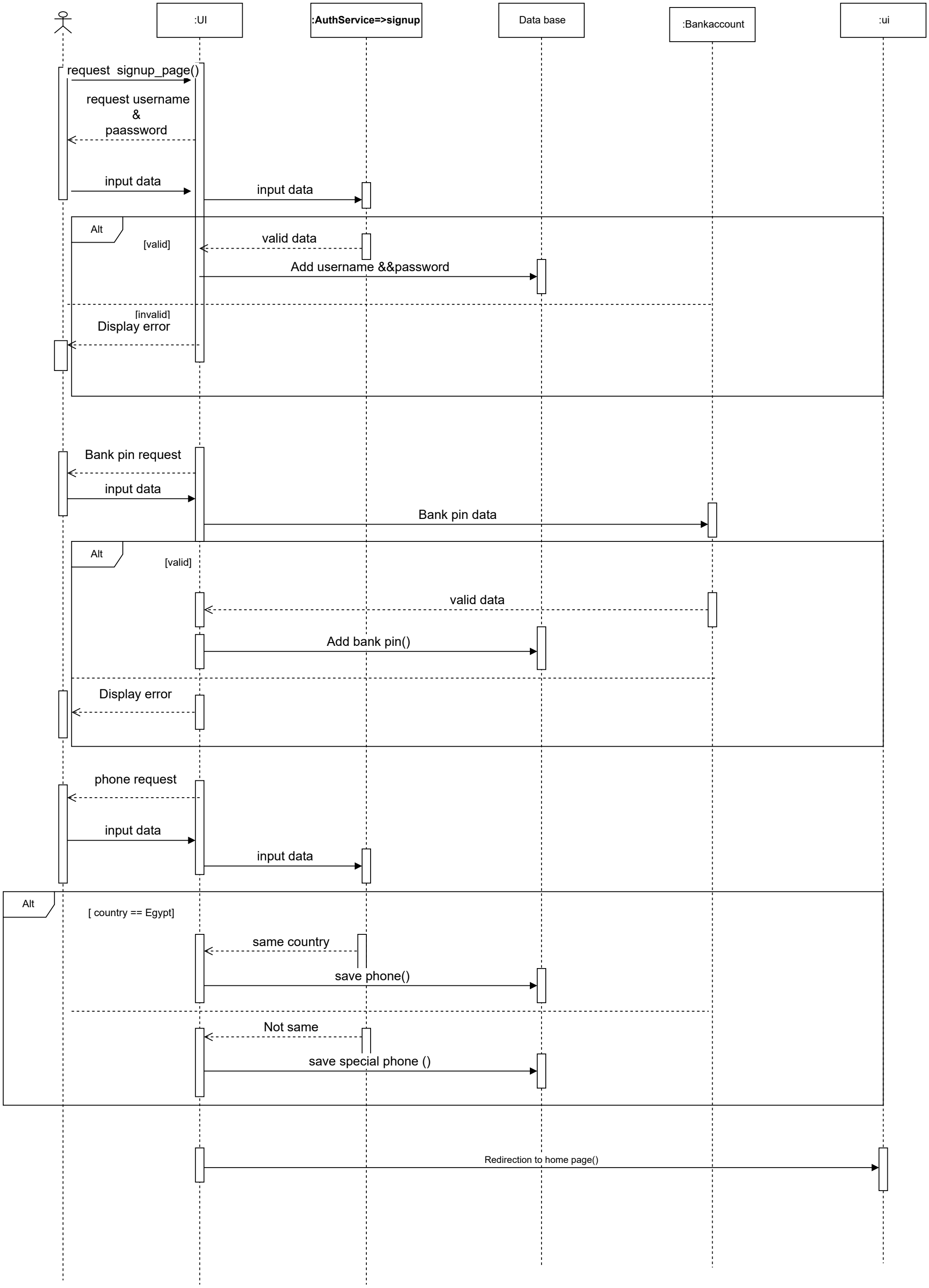
Class Diagram 1 :

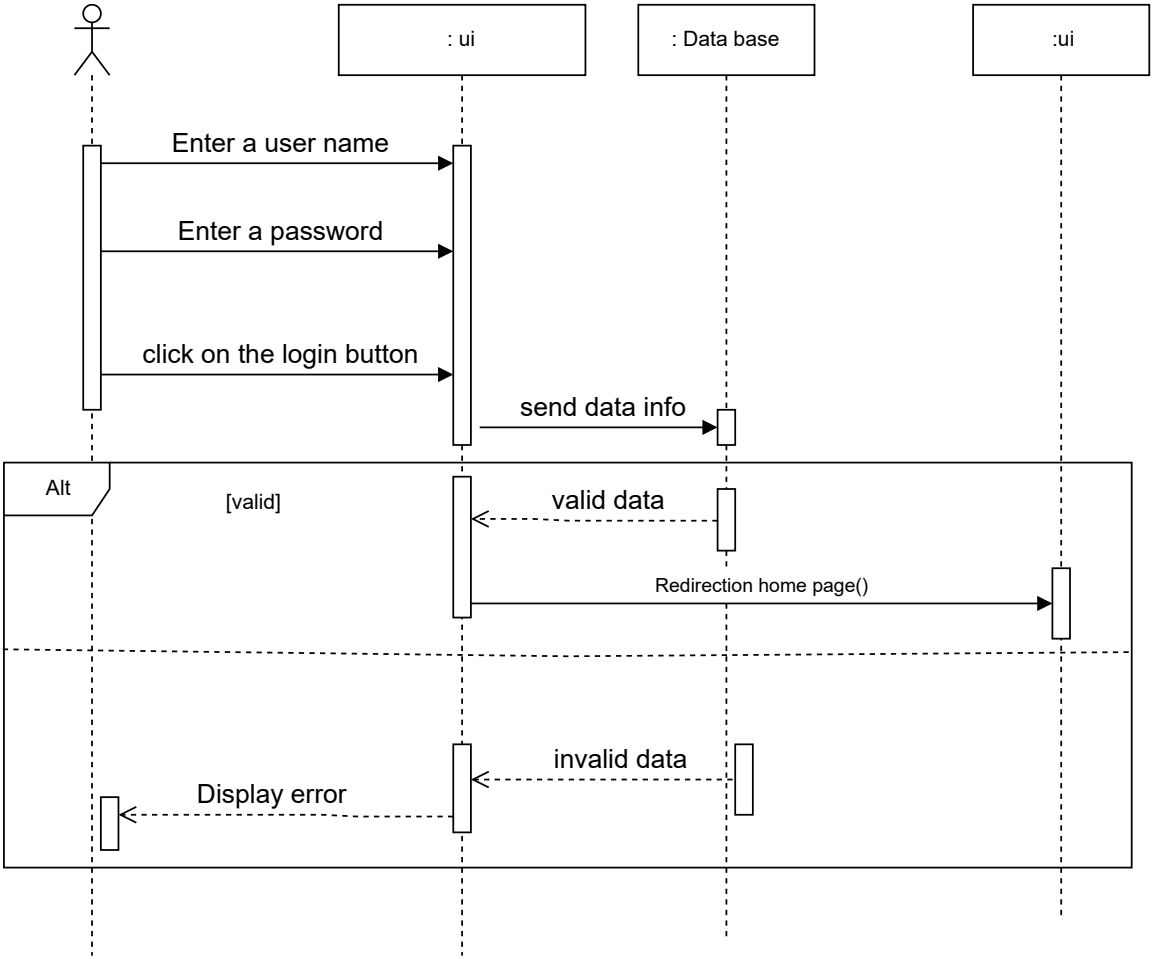


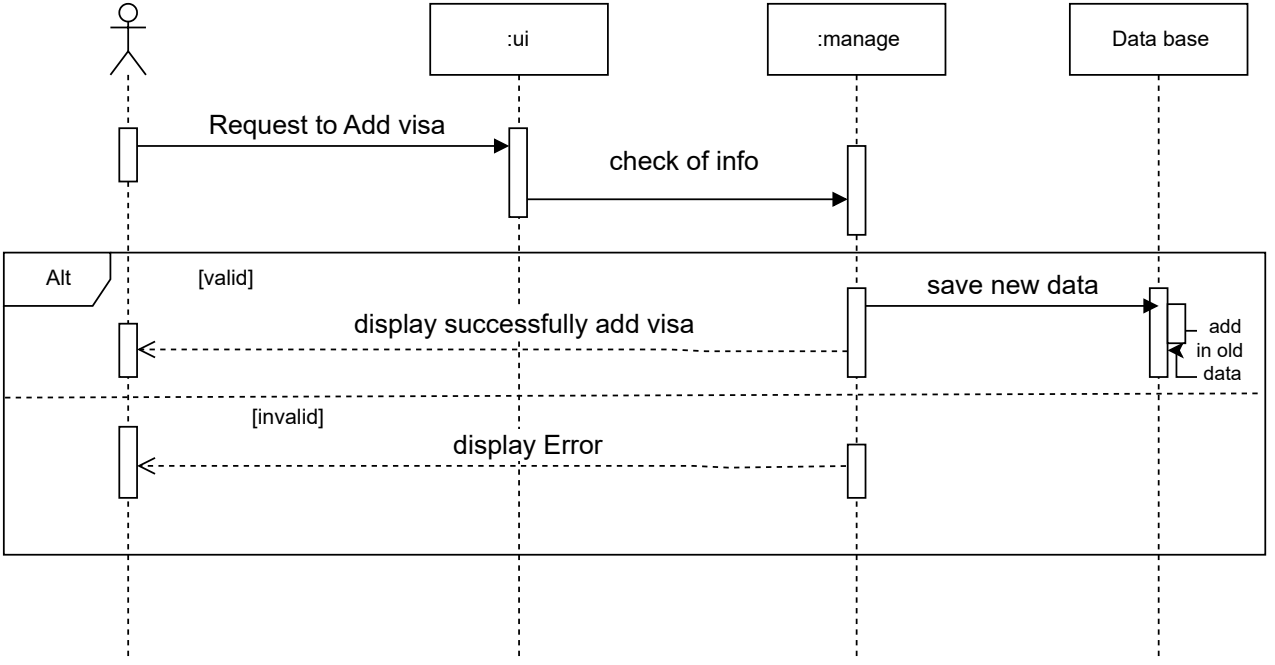
Sequence Diagrams :

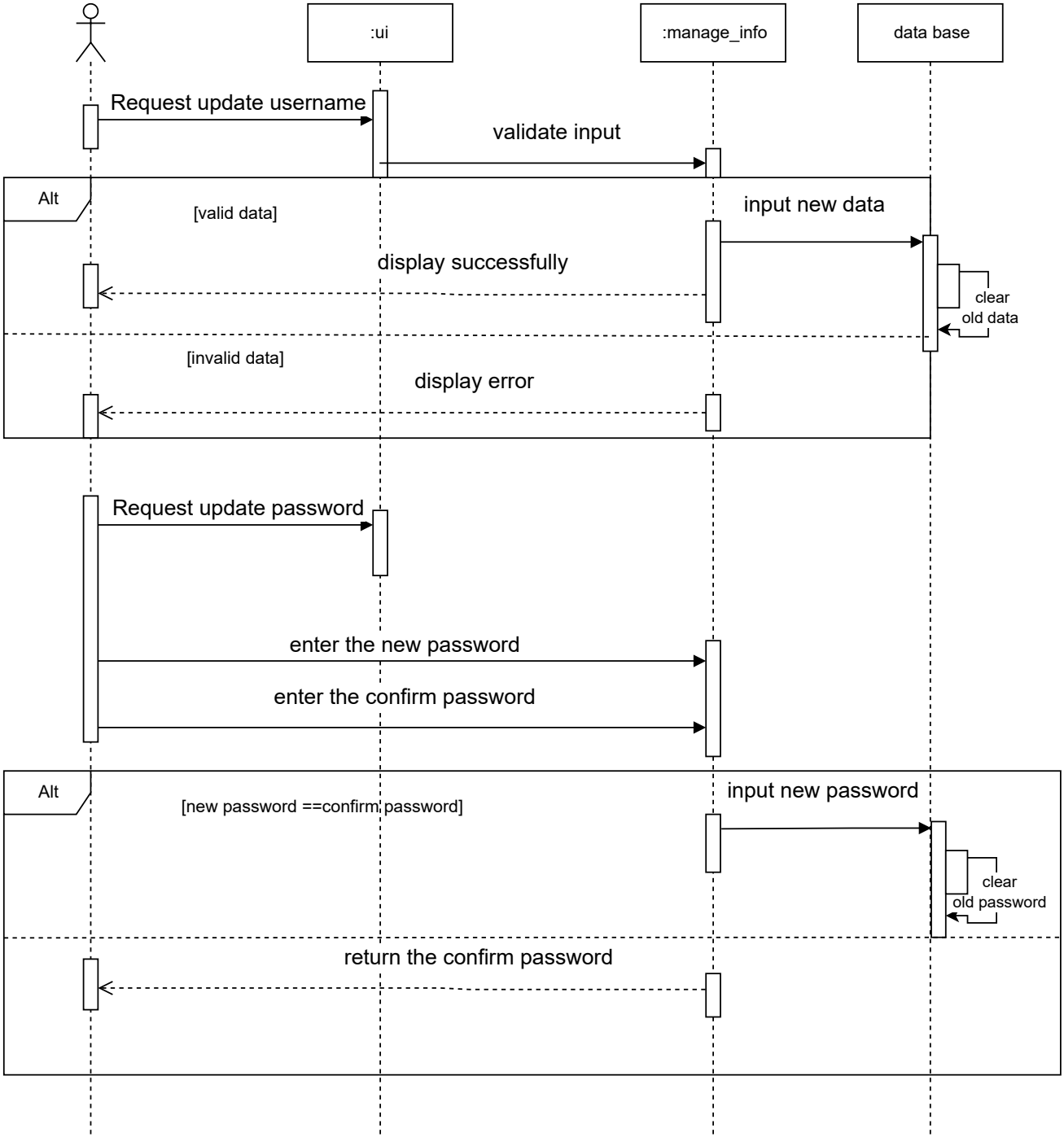

manage_info
+ add visa()
+change_email()
+Reset_password()
+add_contacts( )
+remove( )
User
-Id
-Name
-Phone Number
-Email Address
-bank pin
-Password
- Type of user
+Set name()
+Set password()
+Set email()
+Set visa pin()
+Get phone()
+Display Info()
Customer surver
-Departimant
-Specialization
-Rate
-priority
+respondfeedback():void
+escalateissue():void
+View customer Hestory():void
-Customersurver():void
client
-Balance
-type
-visa pin
-owner name
+check balance(double amount)
+Display()
AuthService
-database
+Sign up()
+Log in()
-log out()
Security
-username
-phone
-Security()
+Get username()
+Set phone()
+Get phone()
+Lock account()
+check username()
+check phone()
+Set username()
Feedback
-feedback_id
-phone
-descrip issue
-Rate
-priority
+Get priority()
+Set priority()
+set issue()
+set Rate()
+Get issue()
+Get Rate()
+is high priority()
+Display Feed back()
Bank_Account
-visa number
-owner name:
-visa
-Bank name: String
+Set visa number()
+Set bank name()
+Deposit()
+Set owner name visa()
+ visa_sms()
Wallet
-WalletID
-type of Wallet
-ReceiverNumber
+OpenWallet()
+CloseWallet()
+Deposit()
+SetReceiverNumber()
Database
Array<User> Users
-db.host
-db.name
-db.password
-Database()
+Encrypt()
+SaveUser()
+FindUserById()
Transaction
-Transaction_Id
-Amount
-Date
-phone
+Deposit()
+Process()
+receive()
+paying bills()
+cancel()
<<Interface>>
Notification
+ Display

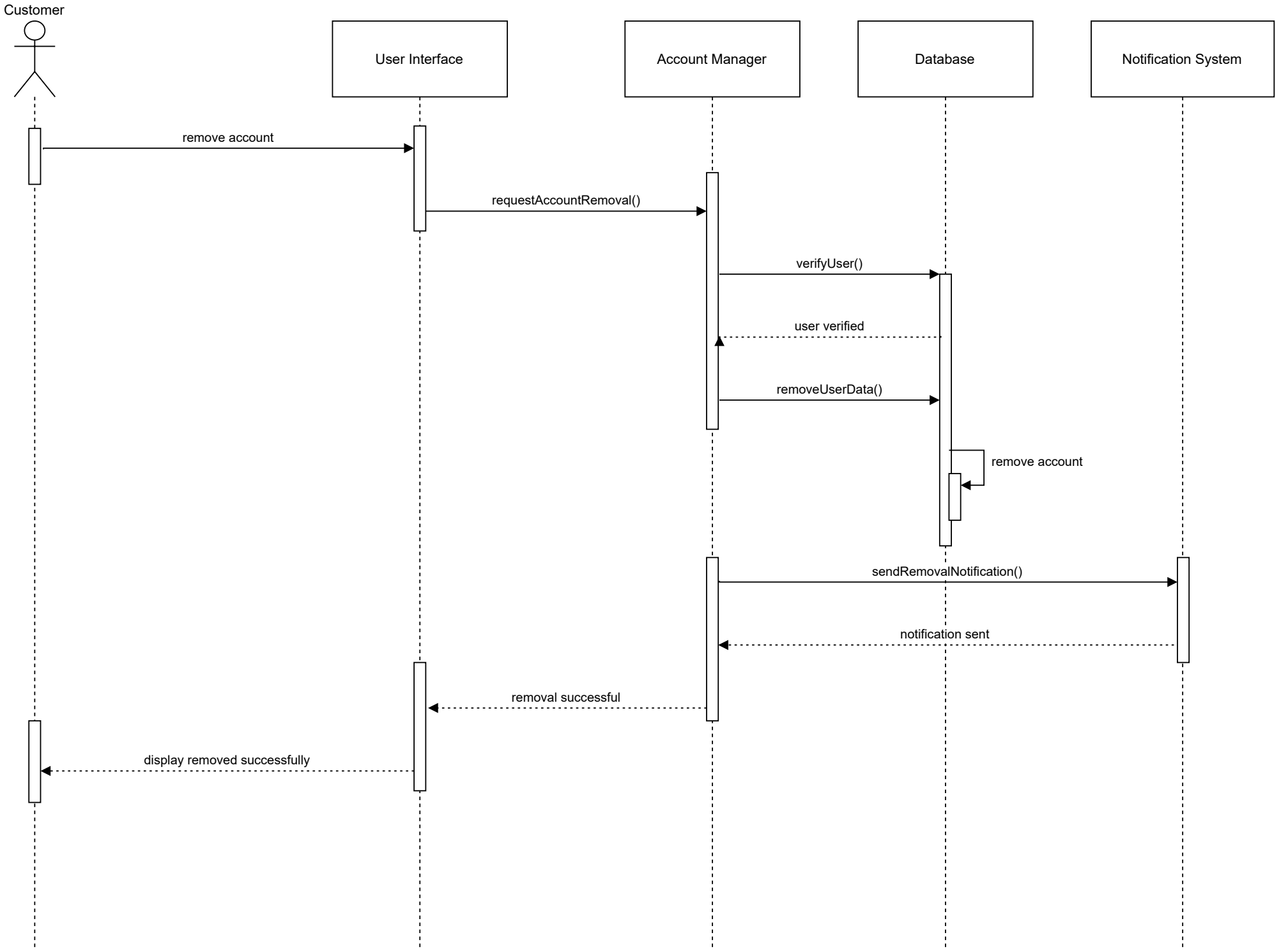

manage_info
+ add visa(int new_visa,int id):void
+change_email(string new email):void
+Reset_password(int new_password,int id):void
+add_contacts(int phone):void
+remove( ):void
User
-id: int
-Name: String
-Phone Number: String
-Email Address: String
-bank pin: String
-Password: String
- Type of user:string
+Set name() : : Void
+Set password() : : Void
+Set email() : : Void
+Set visa pin() : : Void
+Get phone() : : String
+Display Info() : : Void
Customer surver
-Departiment: String
-Specialization: String
+respondfeedback():void
+ handelfeedback():void
+View customer Hestory():void
client
-Balance: double
-type: String
-visa pin: String
-owner name: String
+check balance(double amount) : : bool
+Display() : : Void
AuthService
-database: Database
+Sign up() : : Void
+Log in() : : Void
+Log out() : : Void
Security
-username: String
-phone: String
-Security() : : Void
+Set username() : : Void
+Get username() : : String
+Set phone() : : Void
+Get phone() : : String
+Lock account() : : Bool
+check username() : : Bool
+check phone() : : Bool
Feedback
-feedback_id: int
-phone: String
-descrip issue: String
-Rate: int
-priority: int
+Get priority() : : Int
+Set priority() : : Void
+set issue() : : Void
+set Rate() : : Void
+Get issue() : : String
+Get Rate() : : Int
+is high priority() : : bool
+Display Feed back() : : Void
Bank_Account
-visa number: String
-owner name: visa: String
-Bank name: String
+Set visa number() : : Void
+Set bank name() : : Void
+Deposit(String Account name, Int visa number, String Bank Name, Double Amount) : : Void
+Set owner name visa() : : Void
+Bank_Account() : : Void
Wallet
-WalletID: int
-type: Wallet: String
-ReceiveNumber: String
+OpenWallet(Int Number):void
+CloseWallet():void
+Deposit(String Number, double Amount) : : Void
+SetReceiverNumber(String Number):void
+Wallet(Int Number):void
Database
Array<User> Users
-db.host: String
-db.name: String
-db.password: String
-Database() : void
+Encrypt(String, Hash, Password) : Void
+SaveUser() : Void
+FindUserById() : Bool
Transaction
-Transaction_Id: int
-Amount: double
-Date: Date
-phone: String
+Deposit() : : Void
+Process(Notification) : : Void
+receive(Notification, Amount) : : Void
+paying bills() : : Void
+Transaction() : : Void
<<Interface>>
Notification
+Display(Int Amount, String Date, Int phone) : : Void

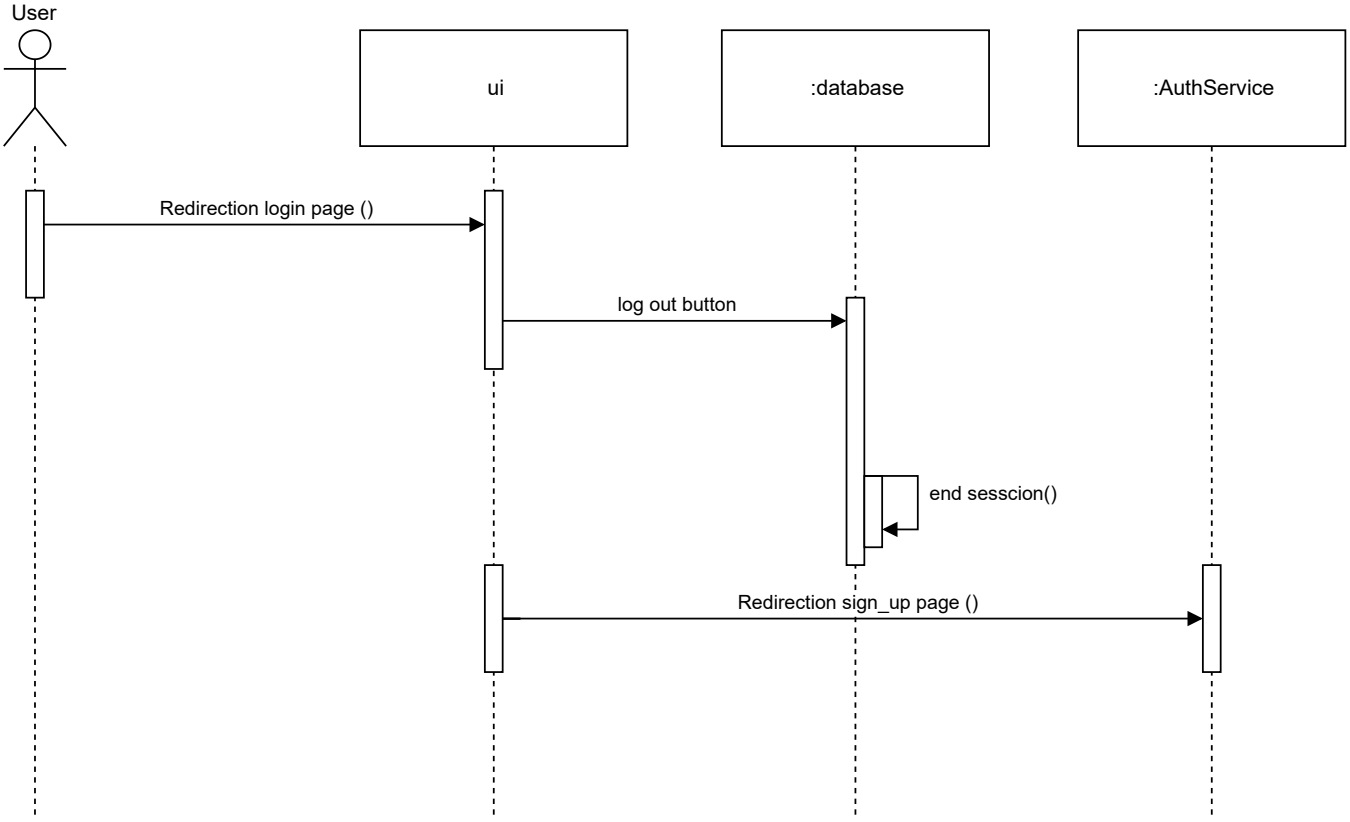

:UI
:AuthService=>signup
Data base
:Bankaccount
:ui
request signup_page()
request username
&
paassword
input data
input data
Alt
[valid]
valid data
Add username &&password
[invalid]
Display error
Bank pin request
input data
Bank pin data
Alt
[valid]
valid data
Add bank pin()
Display error
phone request
input data
input data
Alt
[country == Egypt]
same country
save phone()
Not same
save special phone ()
Redirection to home page()

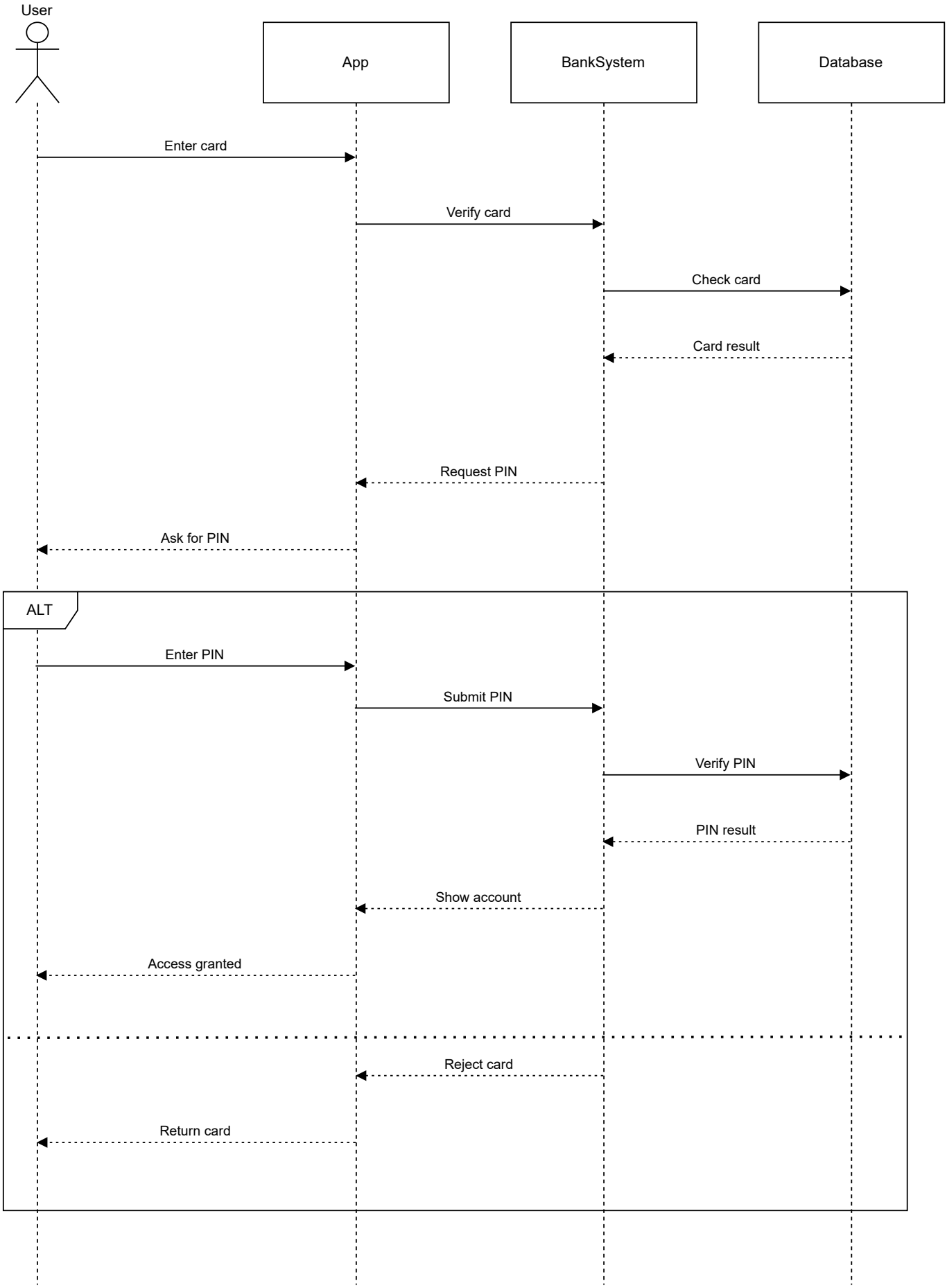

: ui
: Data base
:ui
Enter a user name
Enter a password
click on the login button
send data info
Alt
[valid]
valid data
Redirection home page()
invalid data
Display error

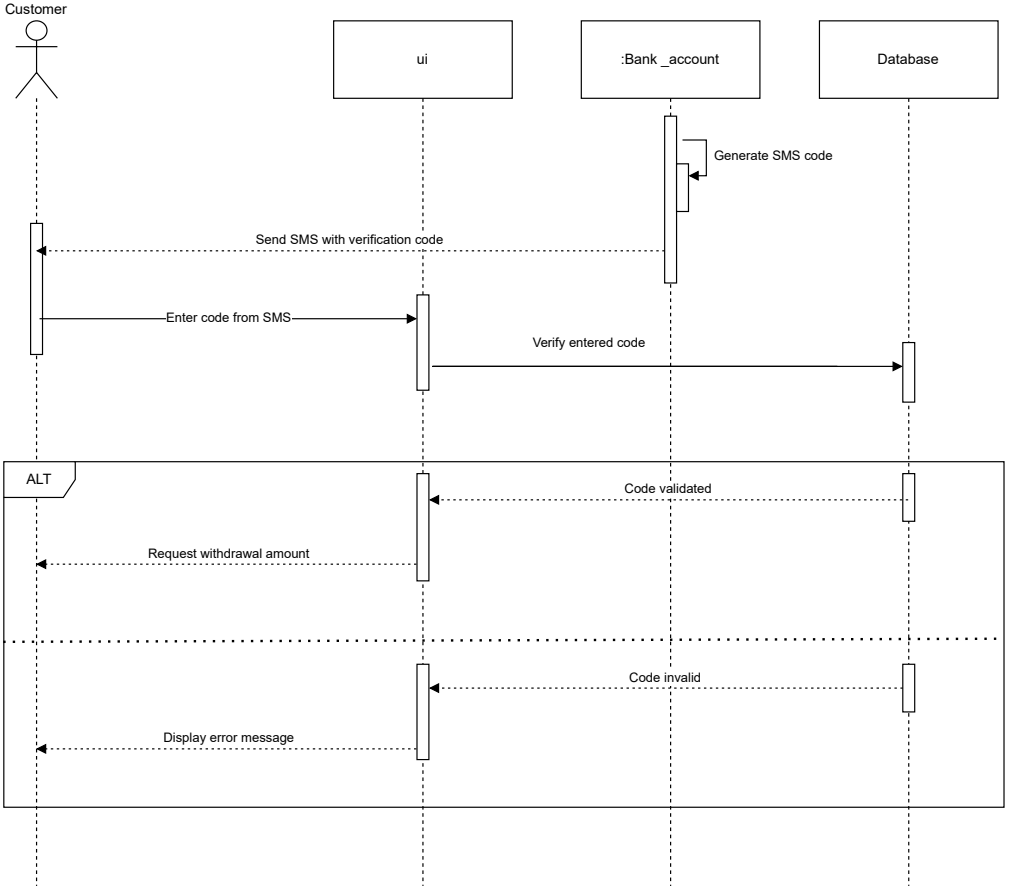

:ui
:manage
Data base
Request to Add visa
check of info
Alt
[valid]
save new data
display successfully add visa
add
in old
data
[invalid]
display Error

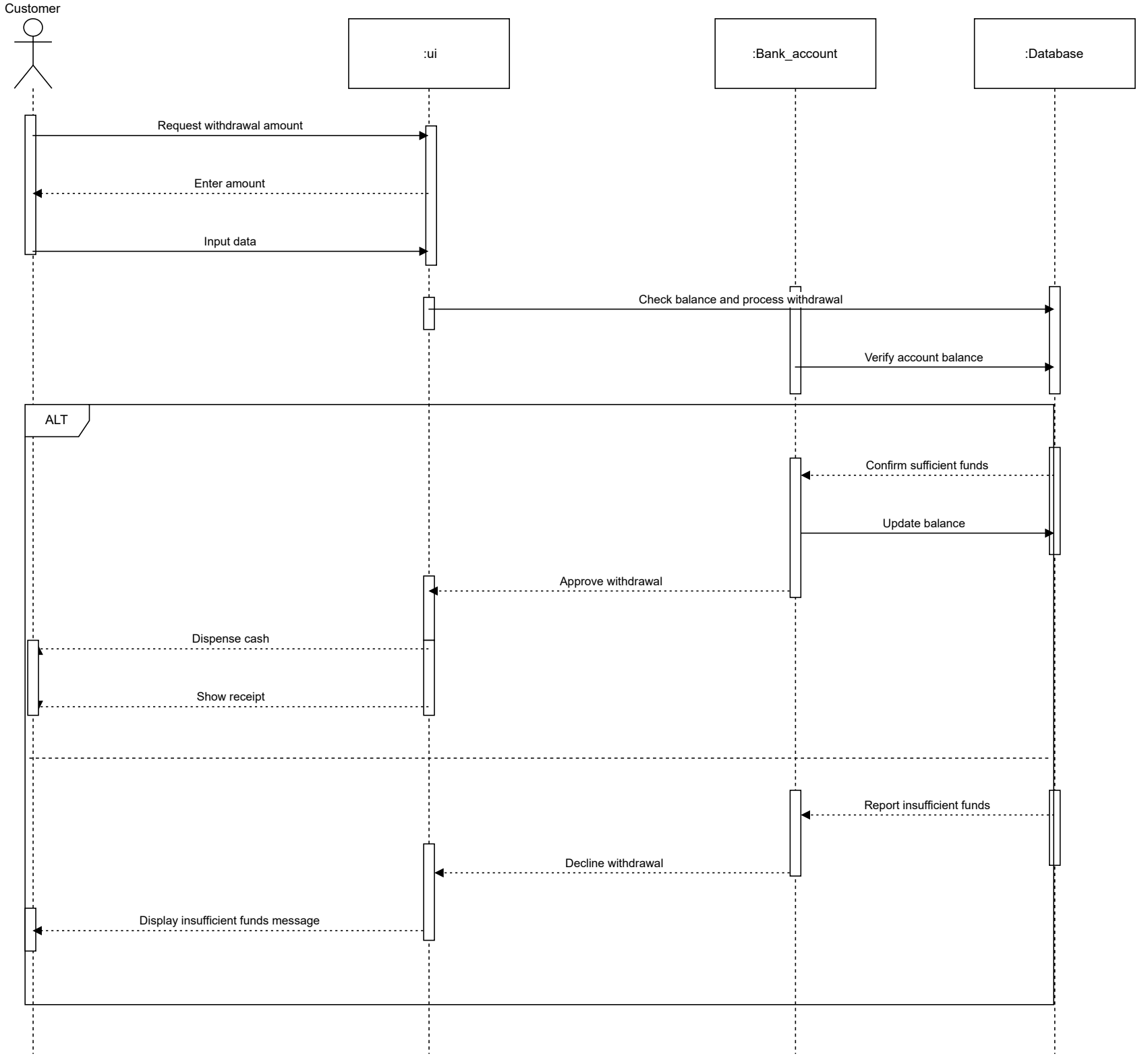

:ui
:manage_info
data base
Request update username
validate input
Alt
[valid data]
input new data
display successfully
clear
old data
[invalid data]
display error
Request update password
enter the new password
enter the confirm password
Alt
[new password ==confirm password]
input new password
clear
old password
return the confirm password

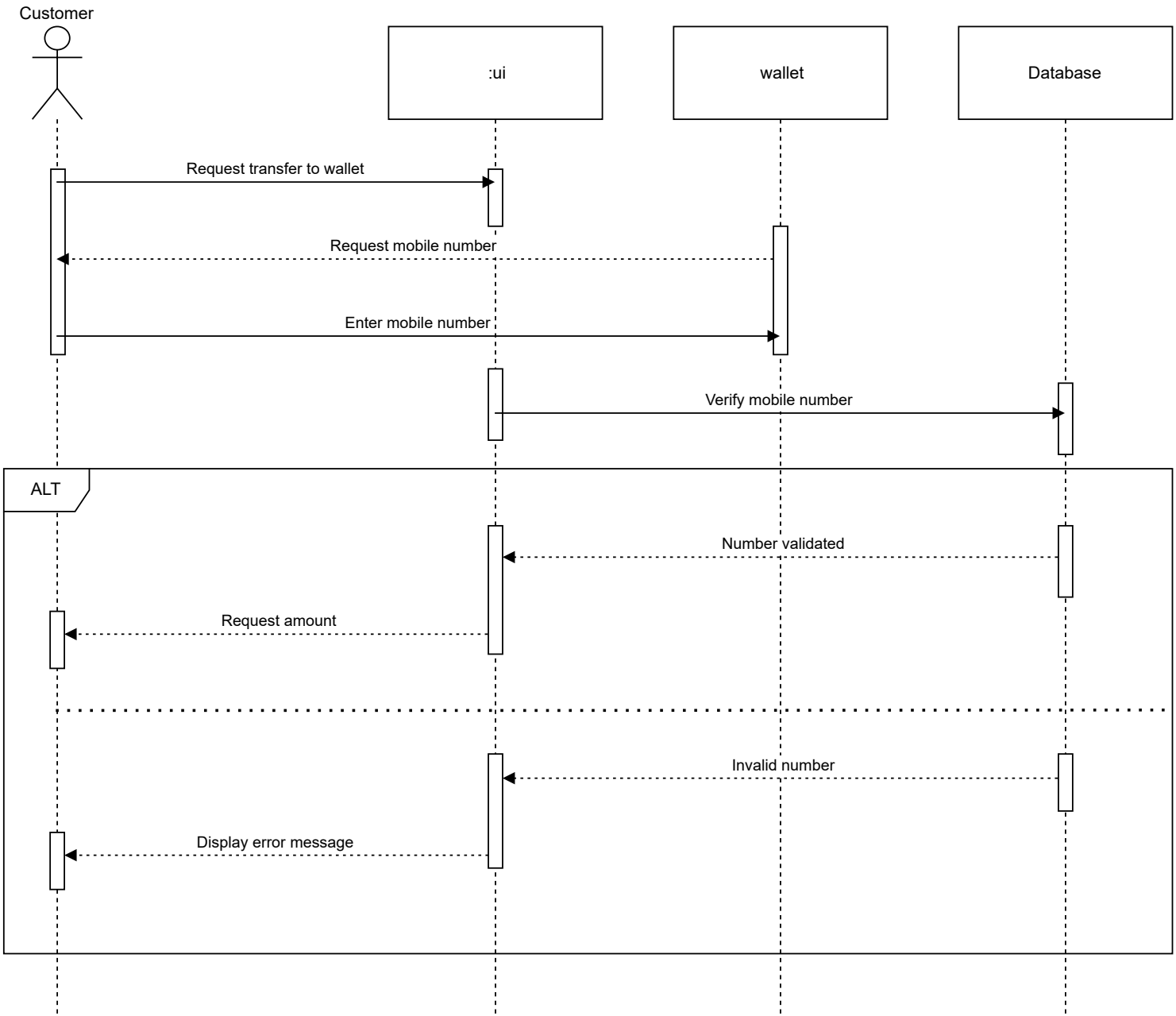

Customer
User Interface
Account Manager
Database
Notification System
remove account
requestAccountRemoval()
verifyUser()
user verified
removeUserData()
remove account
sendRemovalNotification()
notification sent
removal successful
display removed successfully

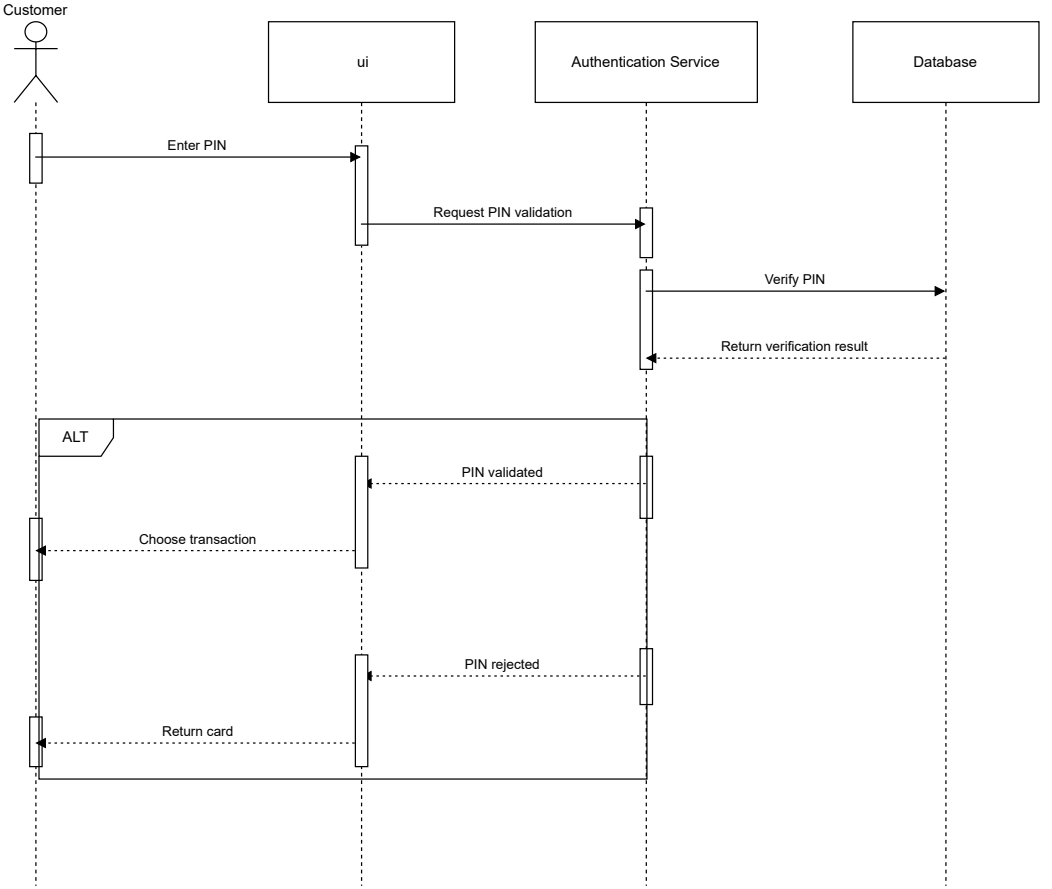

User
ui
:database
:AuthService
Redirection login page ()
log out button
end sesscion()
Redirection sign_up page ()

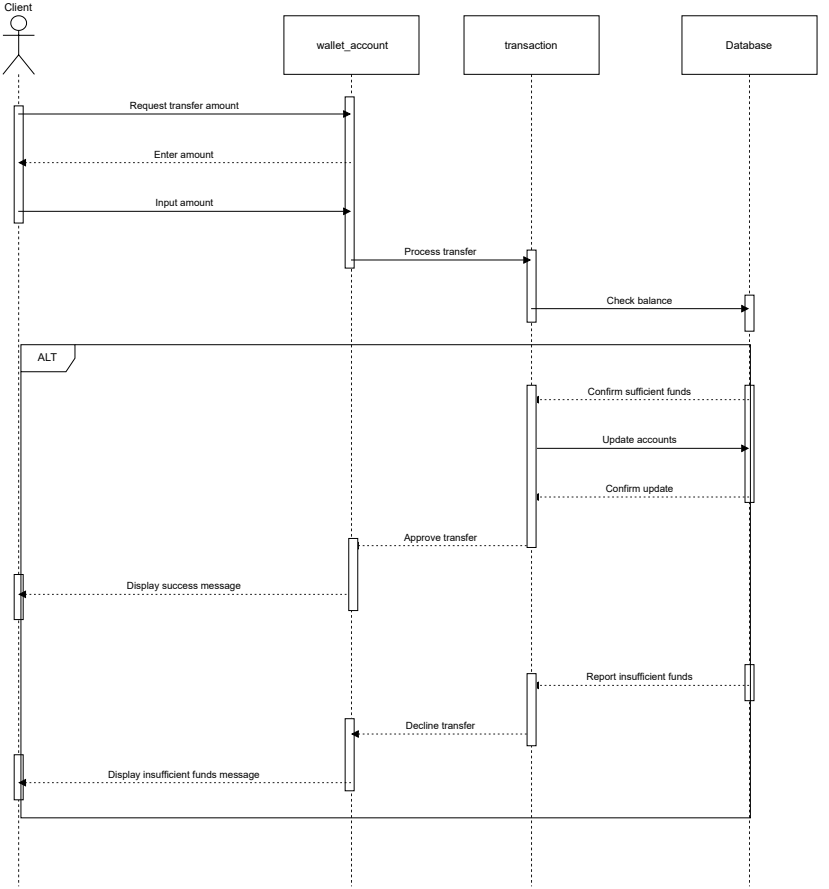

User
App
BankSystem
Database
Enter card
Verify card
Check card
Card result
Request PIN
Ask for PIN
ALT
Enter PIN
Submit PIN
Verify PIN
PIN result
Show account
Access granted
Reject card
Return card

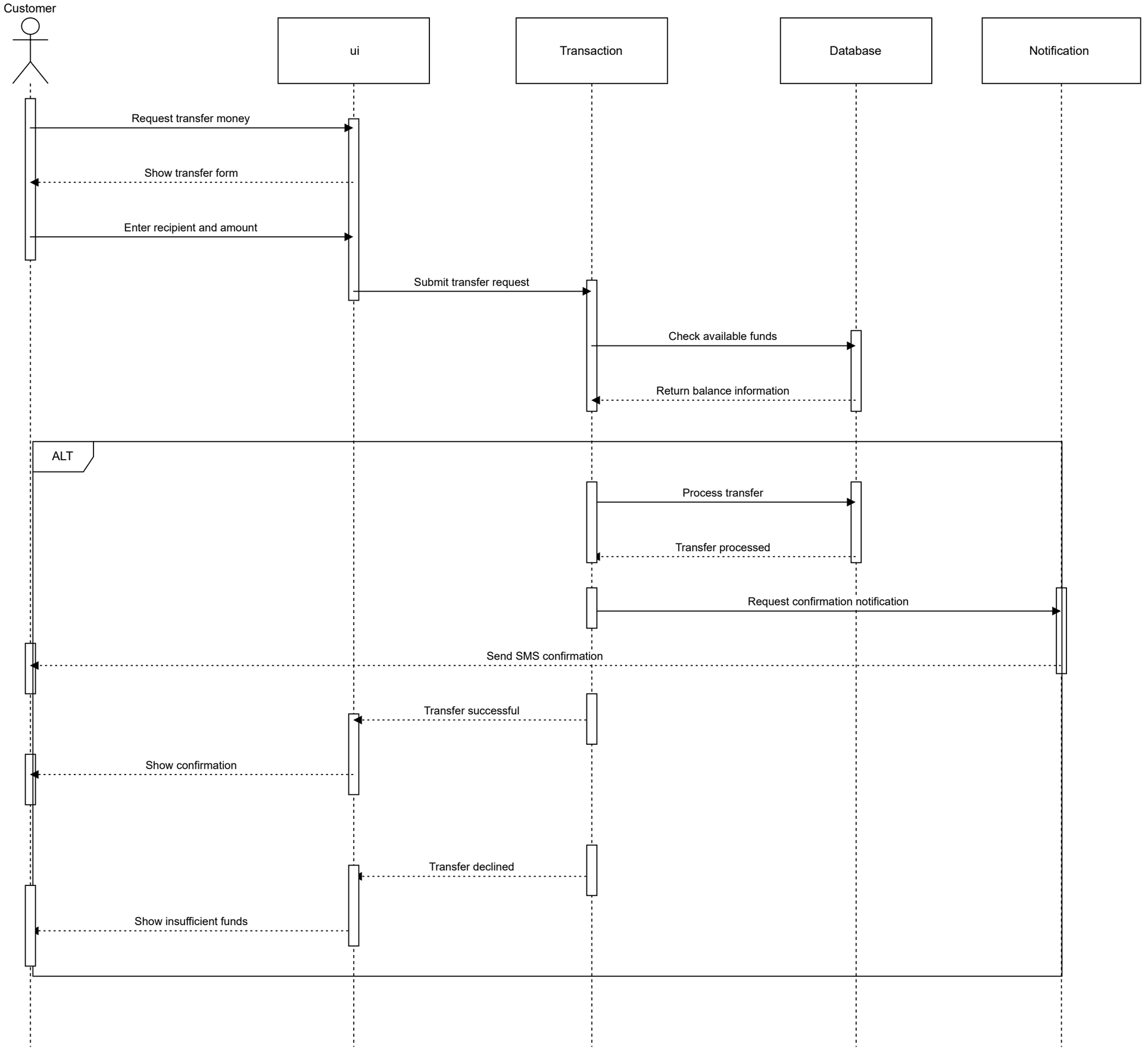

Customer
ui
:Bank _account
Database
Generate SMS code
Send SMS with verification code
Enter code from SMS
Verify entered code
ALT
Code validated
Request withdrawal amount
Code invalid
Display error message

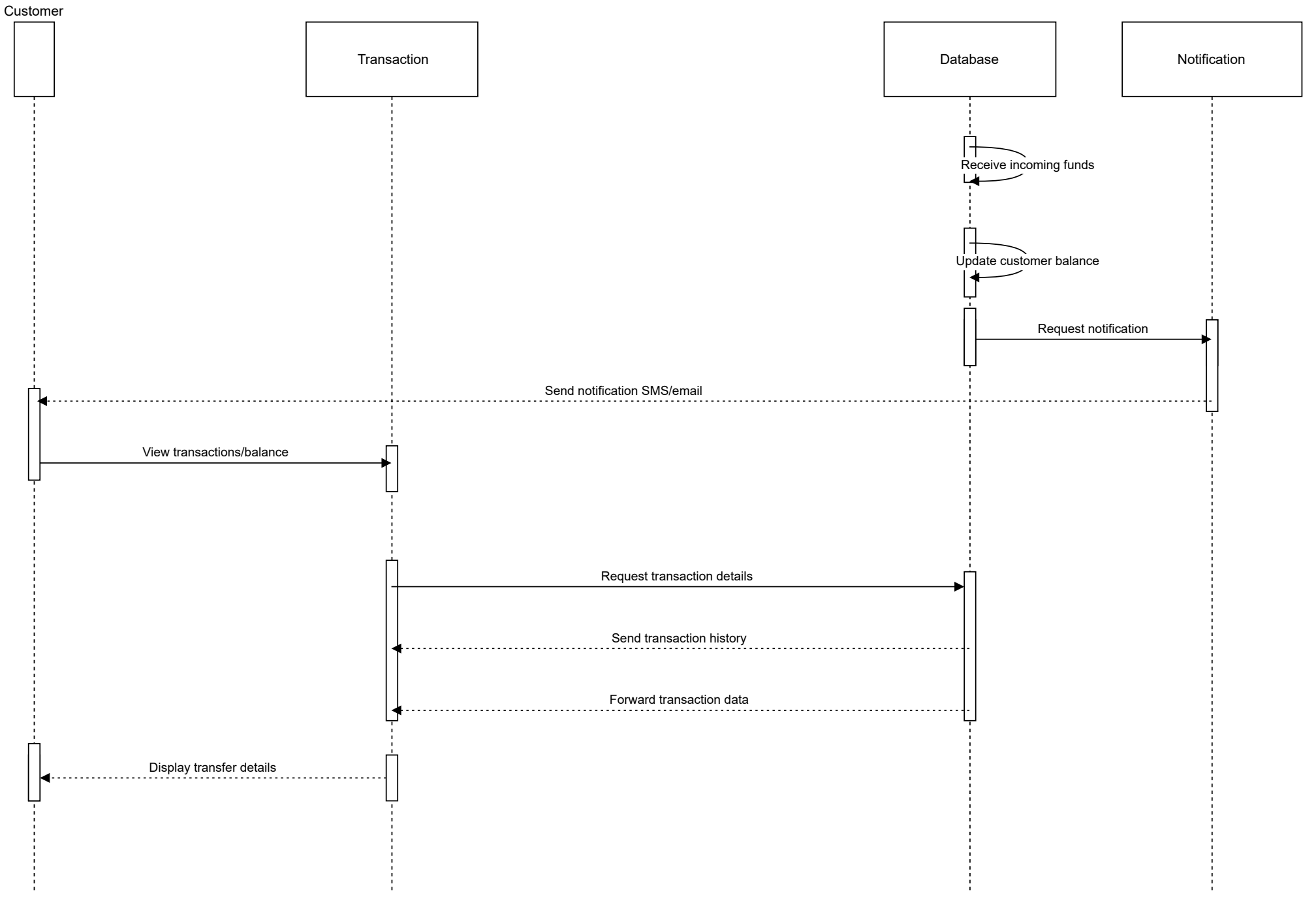

Customer
:ui
:Bank_account
:Database
Request withdrawal amount
Enter amount
Input data
Check balance and process withdrawal
Verify account balance
ALT
Confirm sufficient funds
Update balance
Approve withdrawal
Dispense cash
Show receipt
Report insufficient funds
Decline withdrawal
Display insufficient funds message

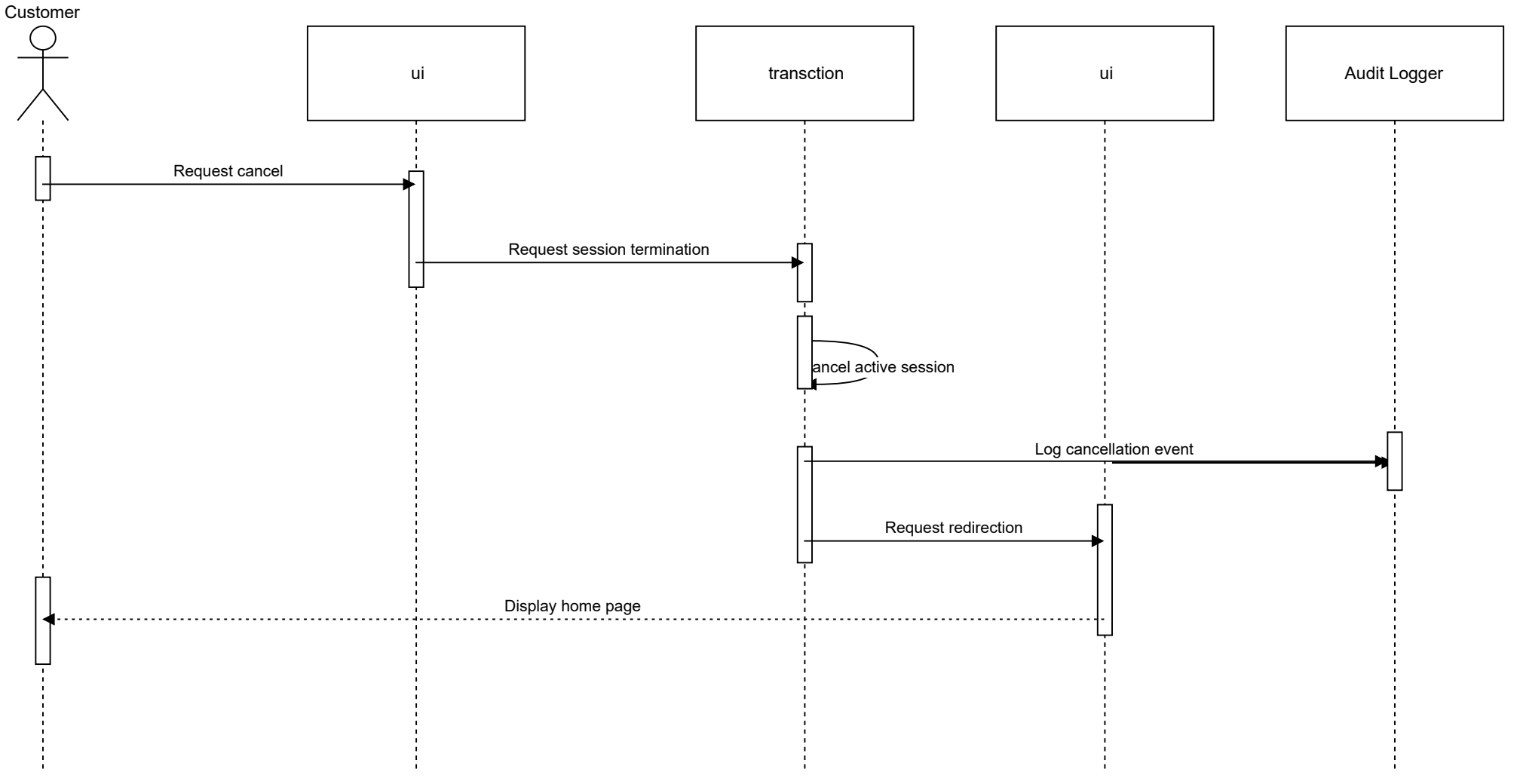

Customer
:ui
wallet
Database
Request transfer to wallet
Request mobile number
Enter mobile number
Verify mobile number
ALT
Number validated
Request amount
Invalid number
Display error message

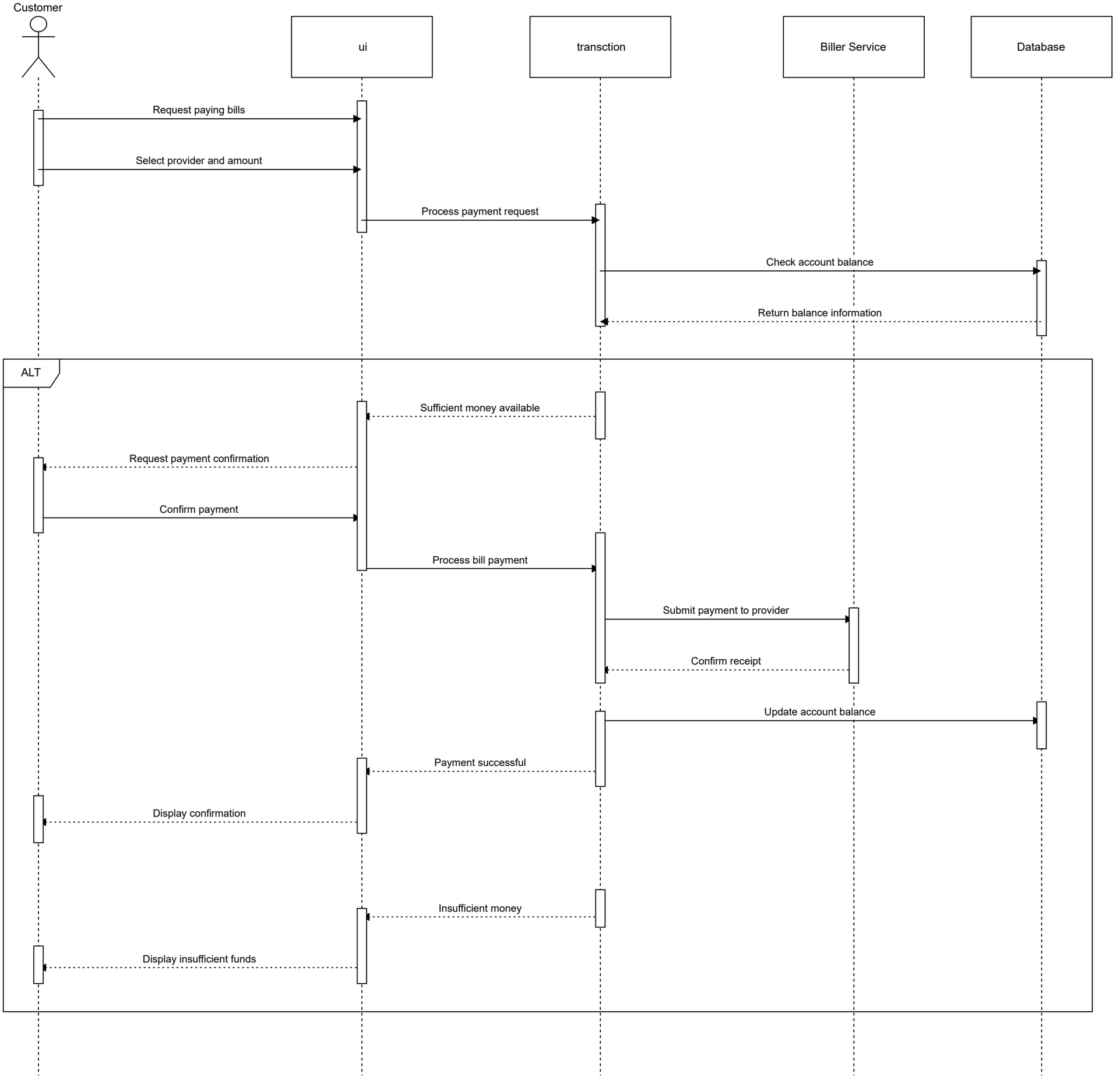

Customer
ui
Authentication Service
Database
Enter PIN
Request PIN validation
Verify PIN
Return verification result
ALT
PIN validated
Choose transaction
PIN rejected
Return card

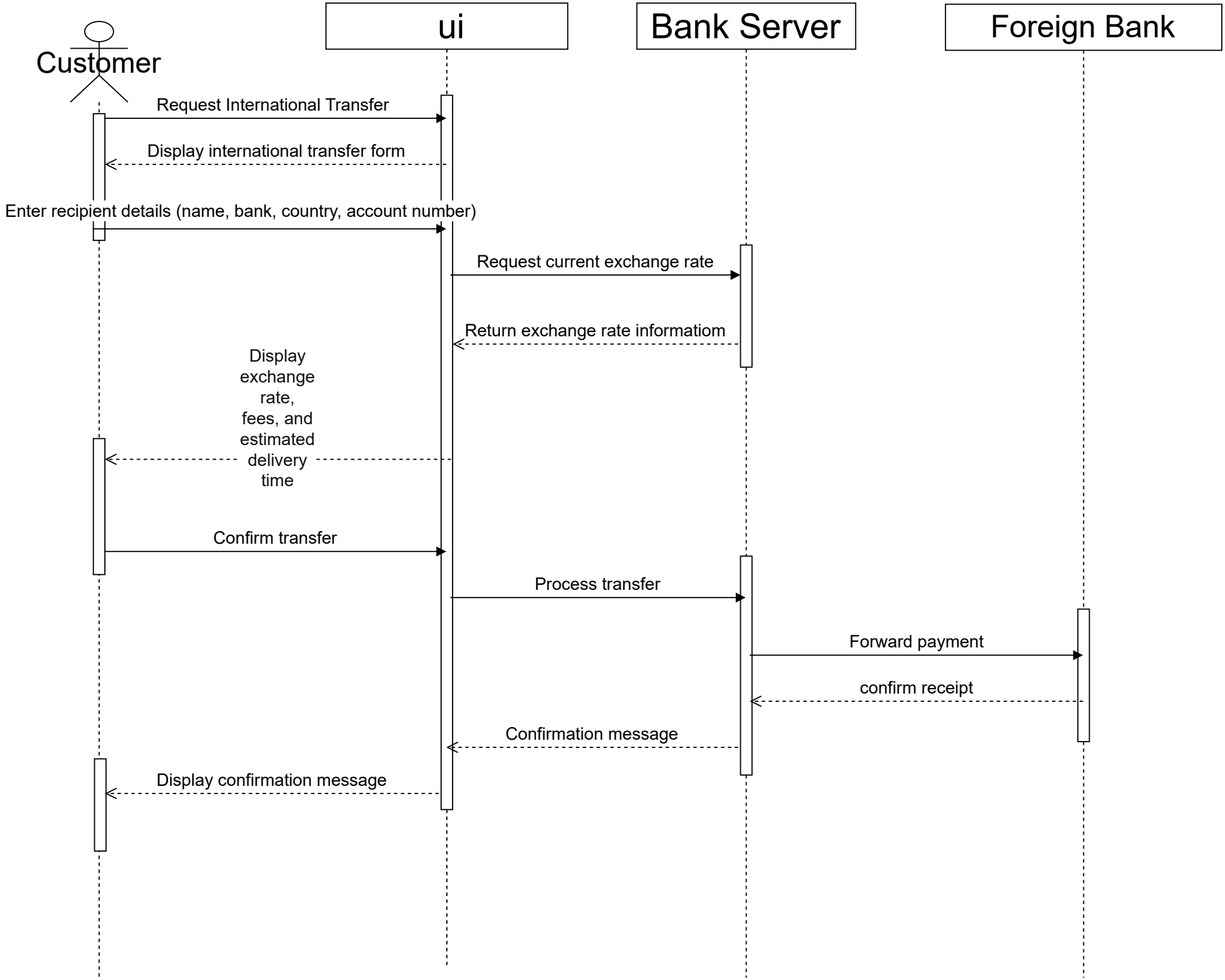

Client
wallet_account
transaction
Database
Request transfer amount
Enter amount
Input amount
Process transfer
Check balance
ALT
Confirm sufficient funds
Update accounts
Confirm update
Approve transfer
Display success message
Report insufficient funds
Decline transfer
Display insufficient funds message

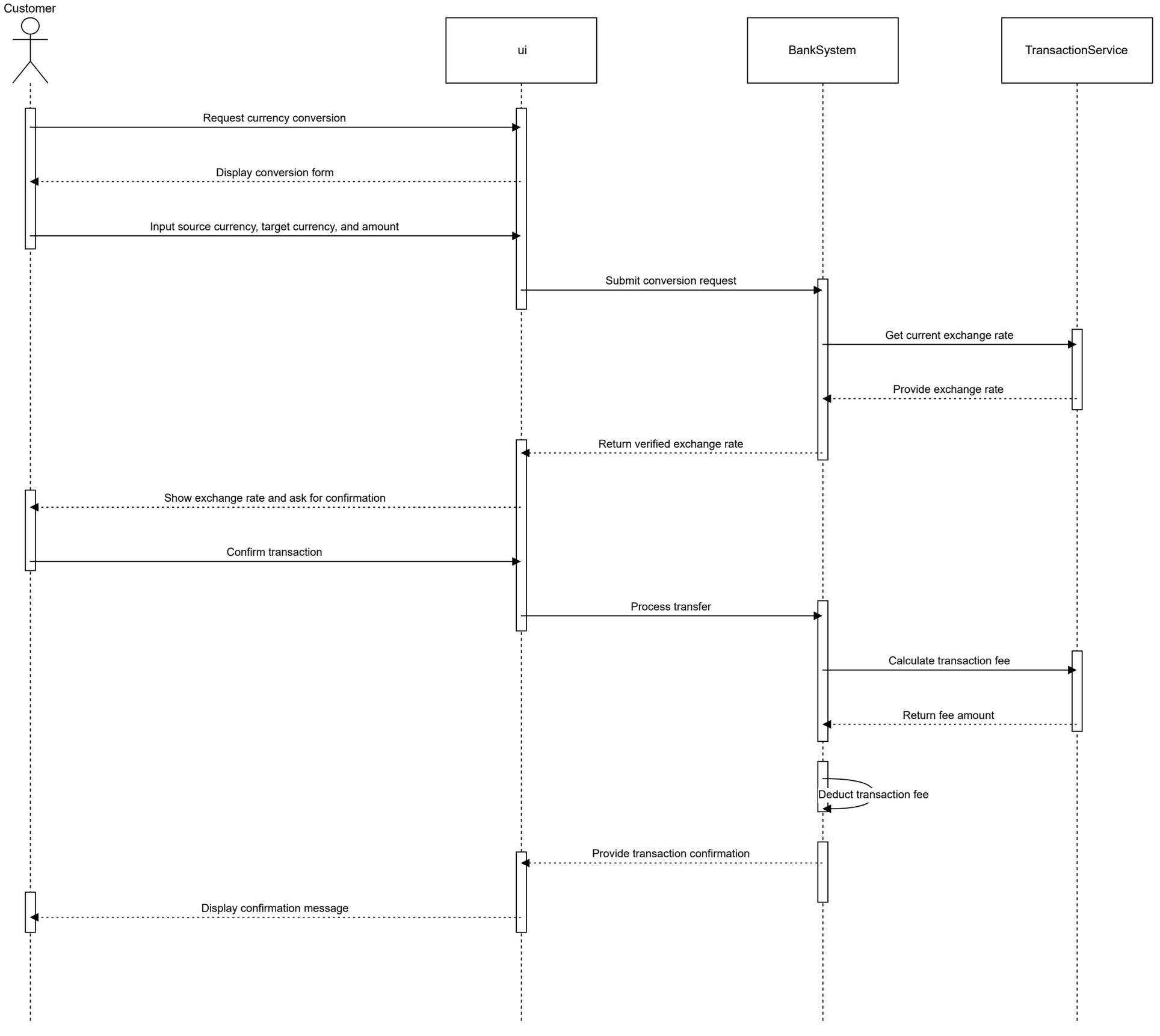

Customer
ui
Transaction
Database
Notification
Request transfer money
Show transfer form
Enter recipient and amount
Submit transfer request
Check available funds
Return balance information
ALT
Process transfer
Transfer processed
Request confirmation notification
Send SMS confirmation
Transfer successful
Show confirmation
Transfer declined
Show insufficient funds

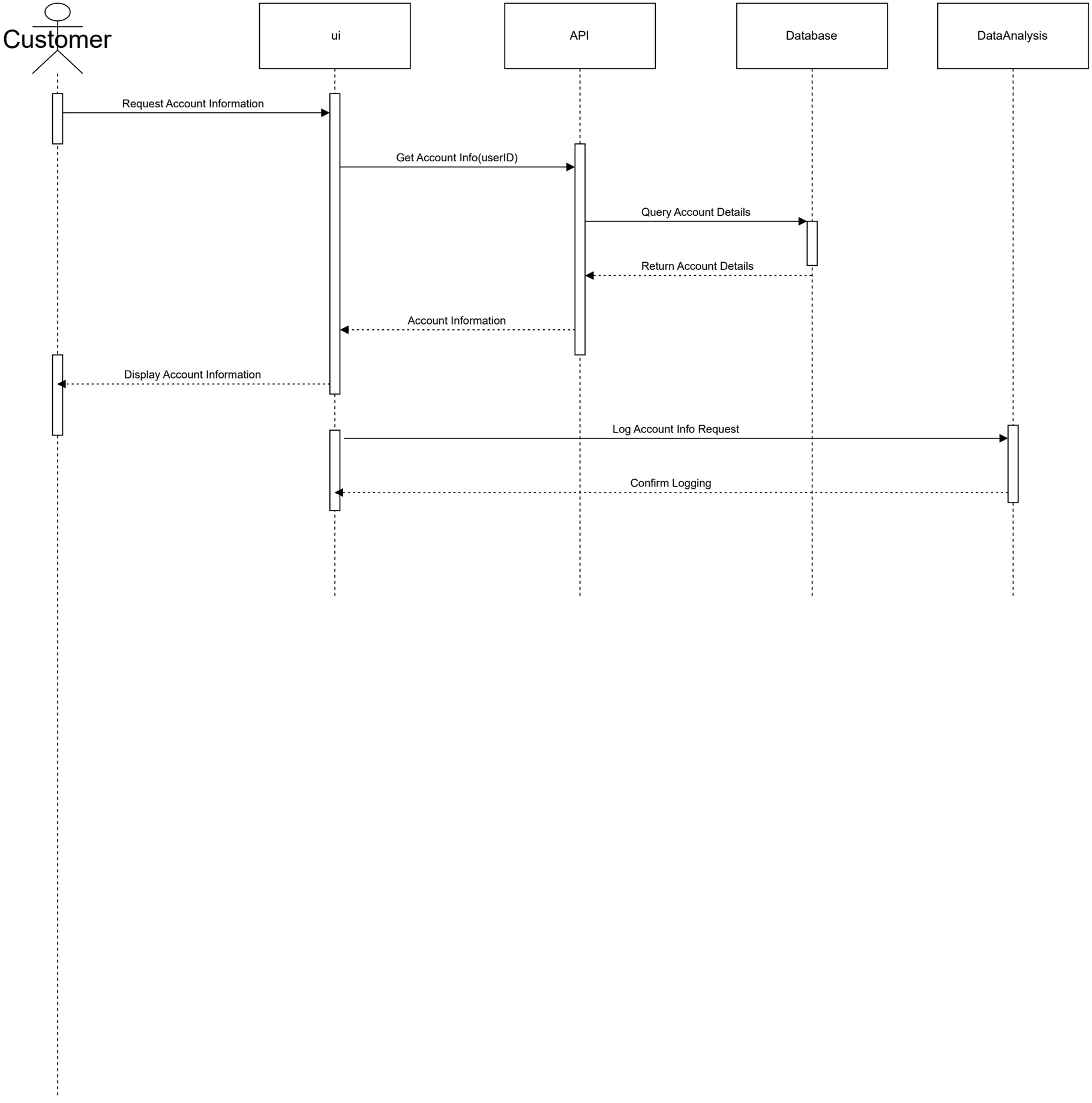

Customer
Transaction
Database
Notification
Receive incoming funds
Update customer balance
Request notification
Send notification SMS/email
View transactions/balance
Request transaction details
Send transaction history
Forward transaction data
Display transfer details

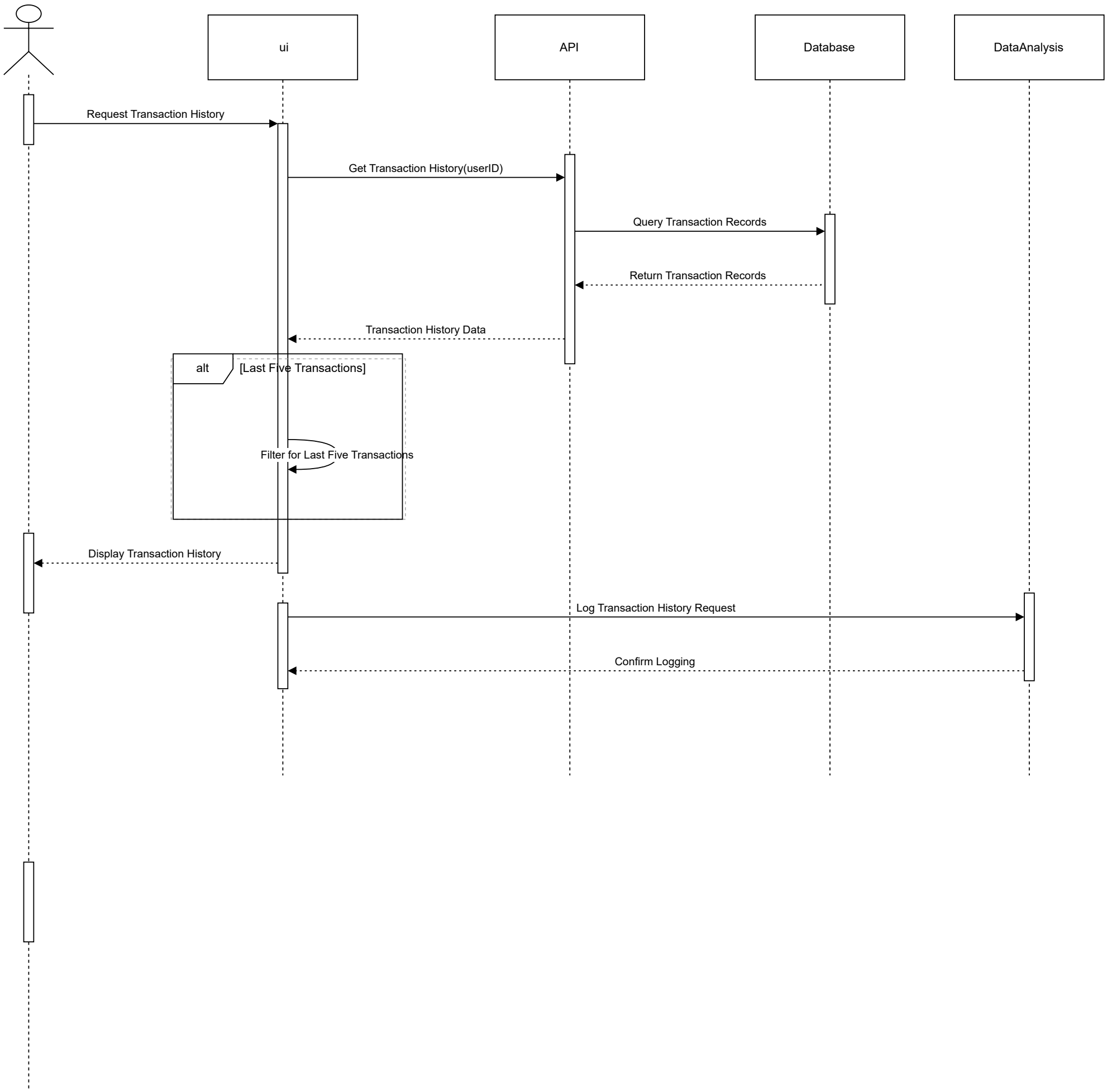

Customer
ui
transction
ui
Audit Logger
Request cancel
Request session termination
ancel active session
Log cancellation event
Request redirection
Display home page

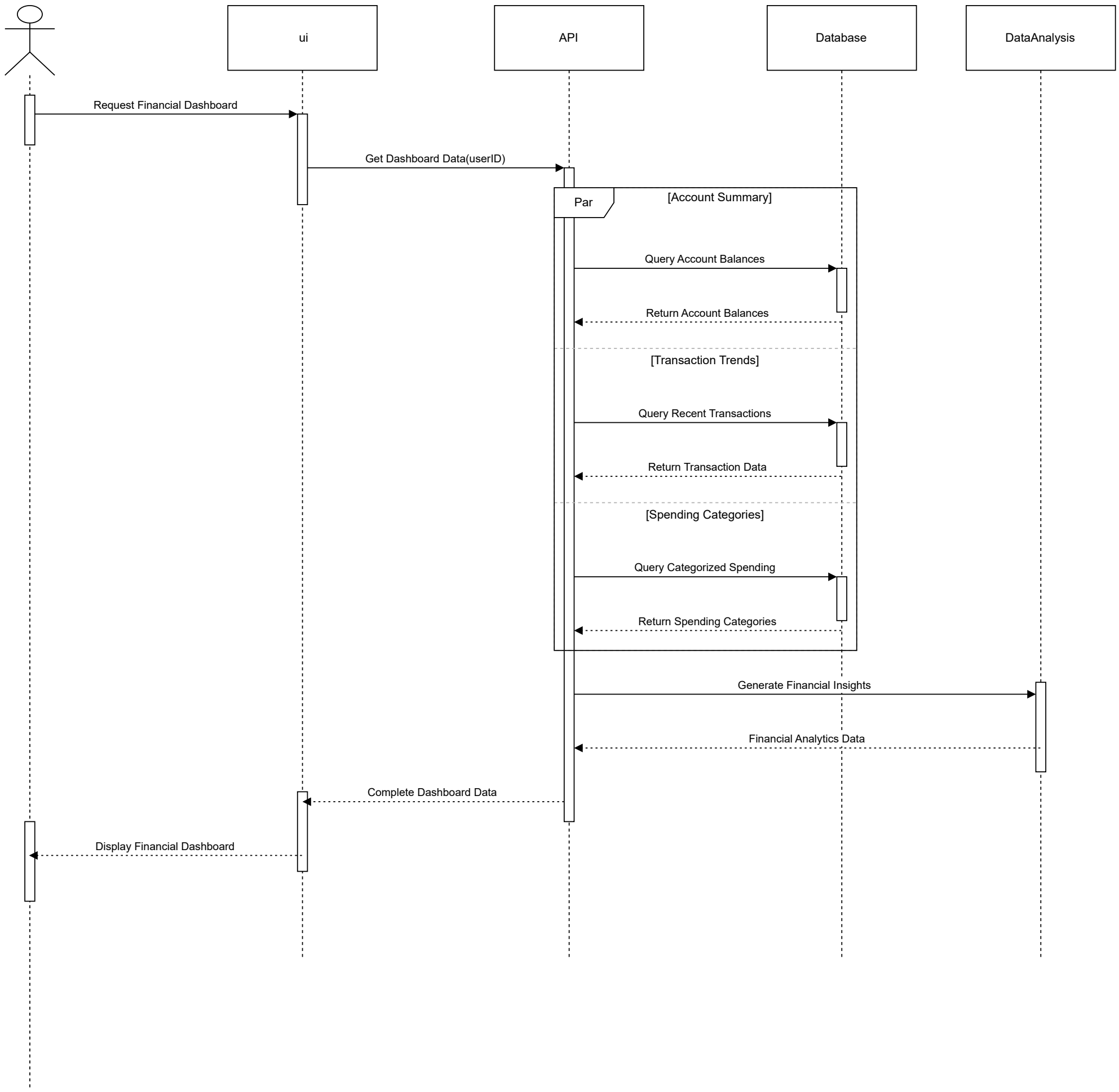

Customer
ui
transction
Biller Service
Database
Request paying bills
Select provider and amount
Process payment request
Check account balance
Return balance information
ALT
Sufficient money available
Request payment confirmation
Confirm payment
Process bill payment
Submit payment to provider
Confirm receipt
Update account balance
Payment successful
Display confirmation
Insufficient money
Display insufficient funds

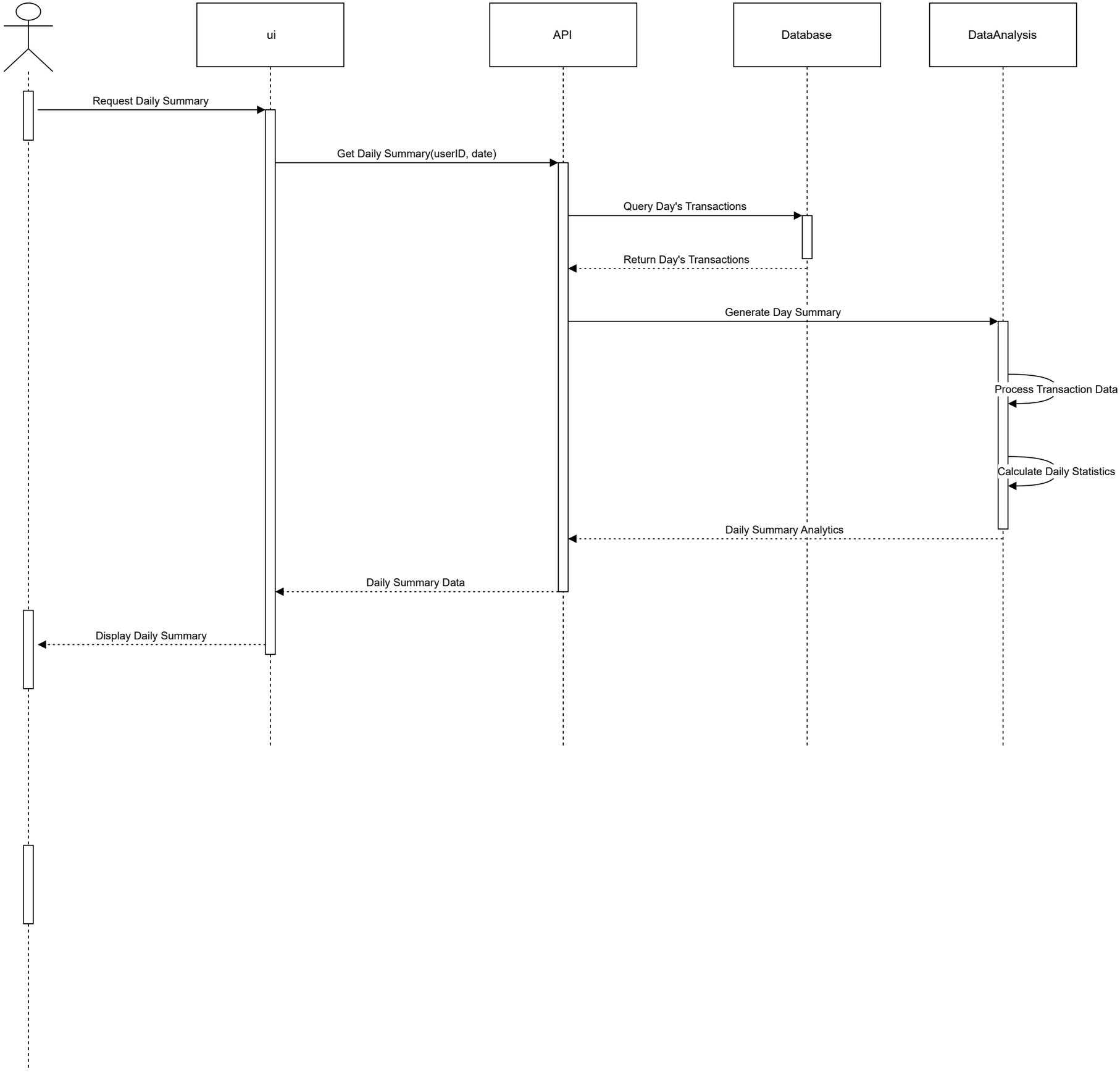

Customer
ui
Bank Server
Foreign Bank
Request International Transfer
Display international transfer form
Enter recipient details (name, bank, country, account number)
Request current exchange rate
Return exchange rate informatiom
Display exchange rate, fees, and estimated delivery time
Confirm transfer
Process transfer
Forward payment
confirm receipt
Confirmation message
Display confirmation message

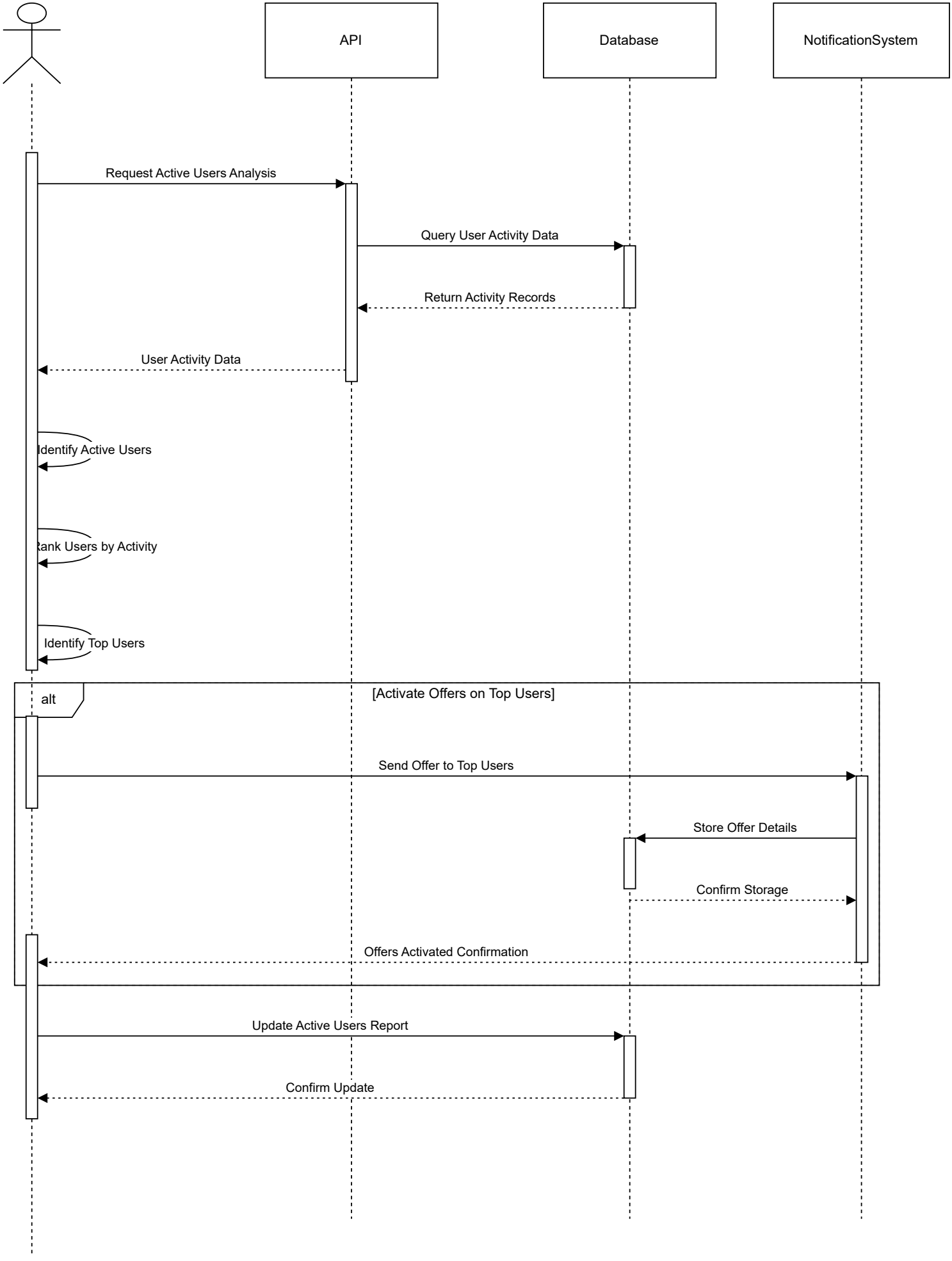

Customer
ui
BankSystem
TransactionService
Request currency conversion
Display conversion form
Input source currency, target currency, and amount
Submit conversion request
Get current exchange rate
Provide exchange rate
Return verified exchange rate
Show exchange rate and ask for confirmation
Confirm transaction
Process transfer
Calculate transaction fee
Return fee amount
Deduct transaction fee
Provide transaction confirmation
Display confirmation message

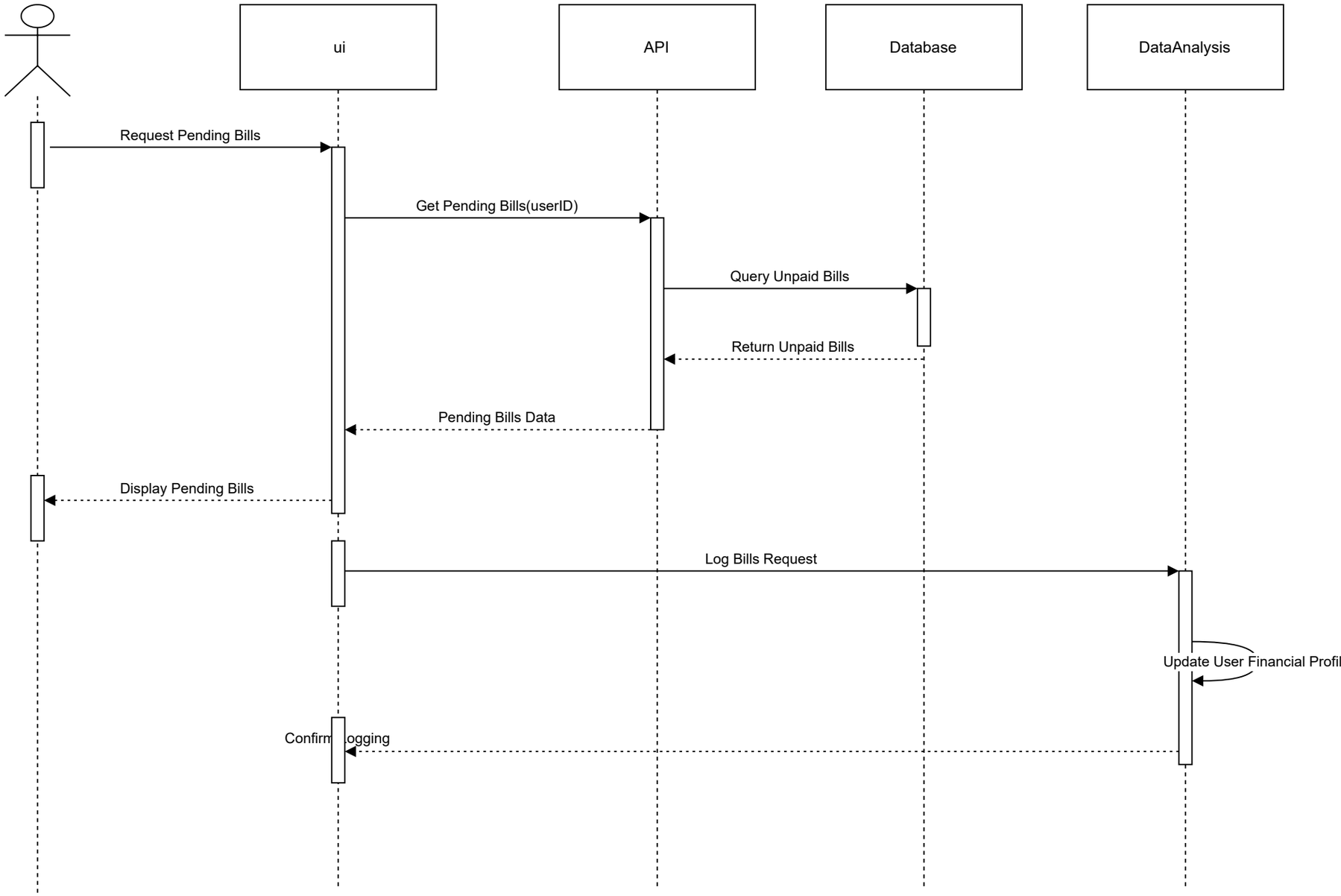

Customer
ui
API
Database
DataAnalysis
Request Account Information
Get Account Info(userID)
Query Account Details
Return Account Details
Account Information
Display Account Information
Log Account Info Request
Confirm Logging

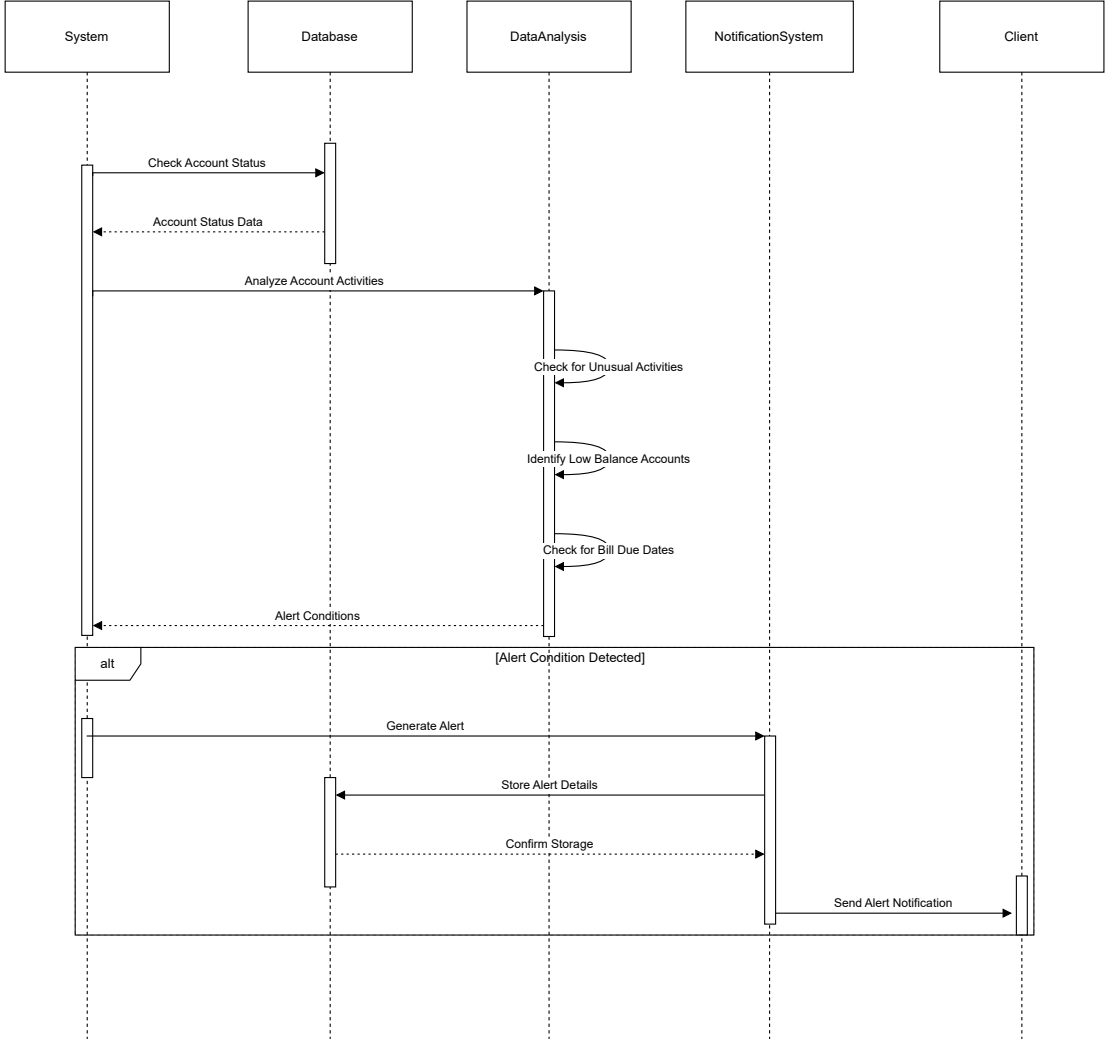

ui
API
Database
DataAnalysis
Request Transaction History
Get Transaction History(userID)
Query Transaction Records
Return Transaction Records
Transaction History Data
alt
[Last Five Transactions]
Filter for Last Five Transactions
Display Transaction History
Log Transaction History Request
Confirm Logging

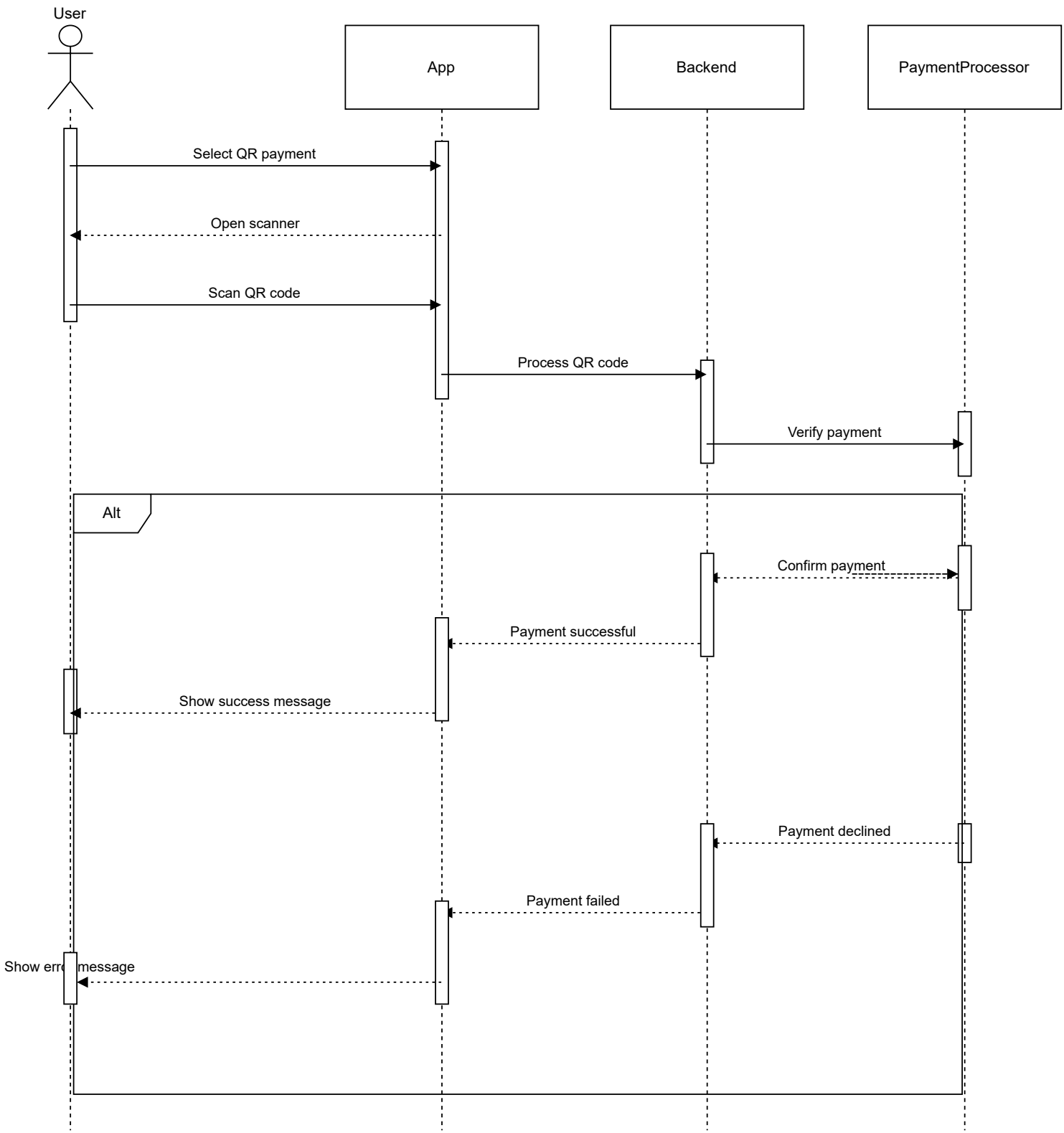

ui
API
Database
DataAnalysis
Request Financial Dashboard
Get Dashboard Data(userID)
Par
[Account Summary]
Query Account Balances
Return Account Balances
[Transaction Trends]
Query Recent Transactions
Return Transaction Data
[Spending Categories]
Query Categorized Spending
Return Spending Categories
Generate Financial Insights
Financial Analytics Data
Complete Dashboard Data
Display Financial Dashboard

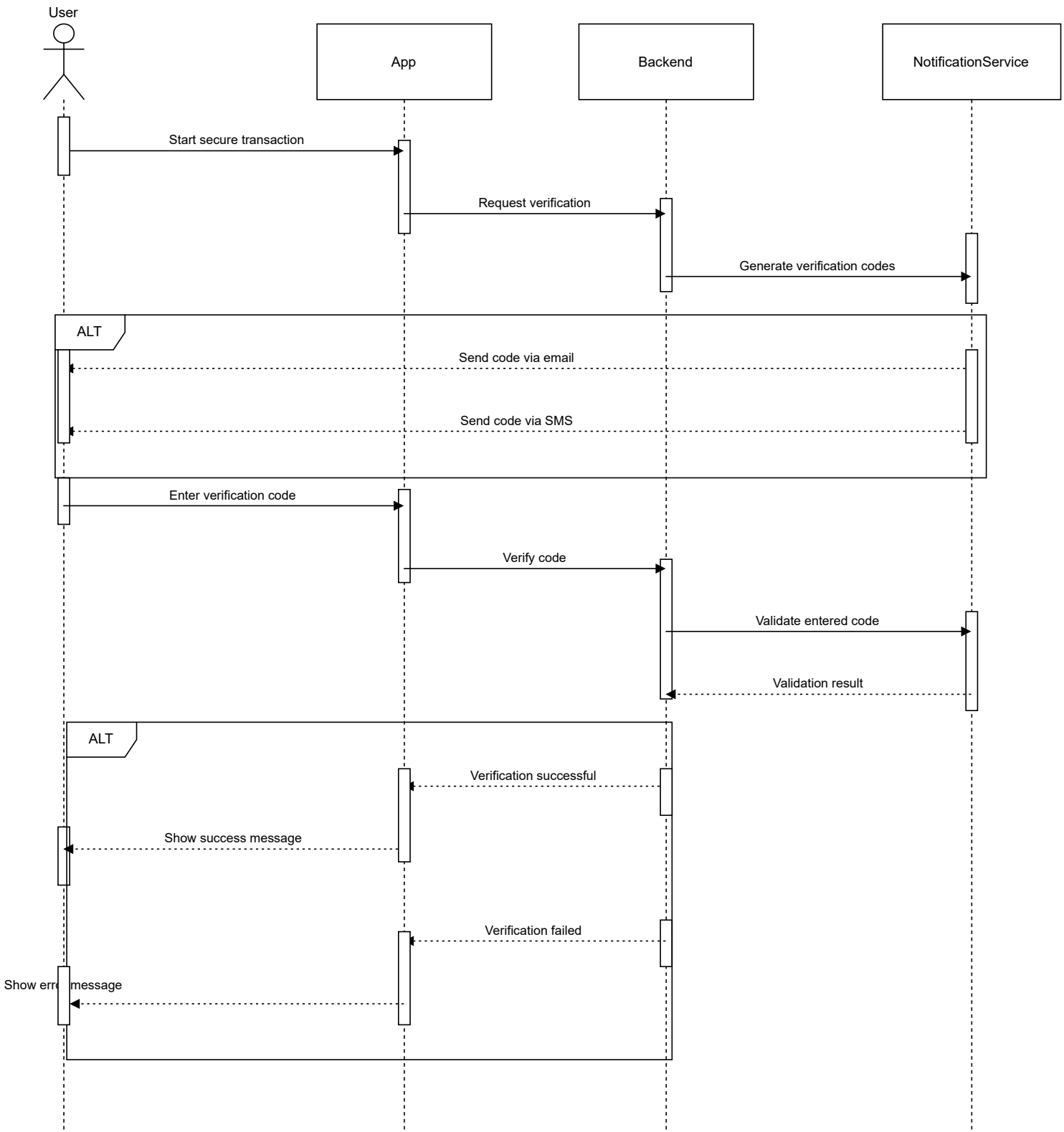

ui
API
Database
DataAnalysis
Request Daily Summary
Get Daily Summary(userID, date)
Query Day's Transactions
Return Day's Transactions
Generate Day Summary
Process Transaction Data
Calculate Daily Statistics
Daily Summary Analytics
Daily Summary Data
Display Daily Summary

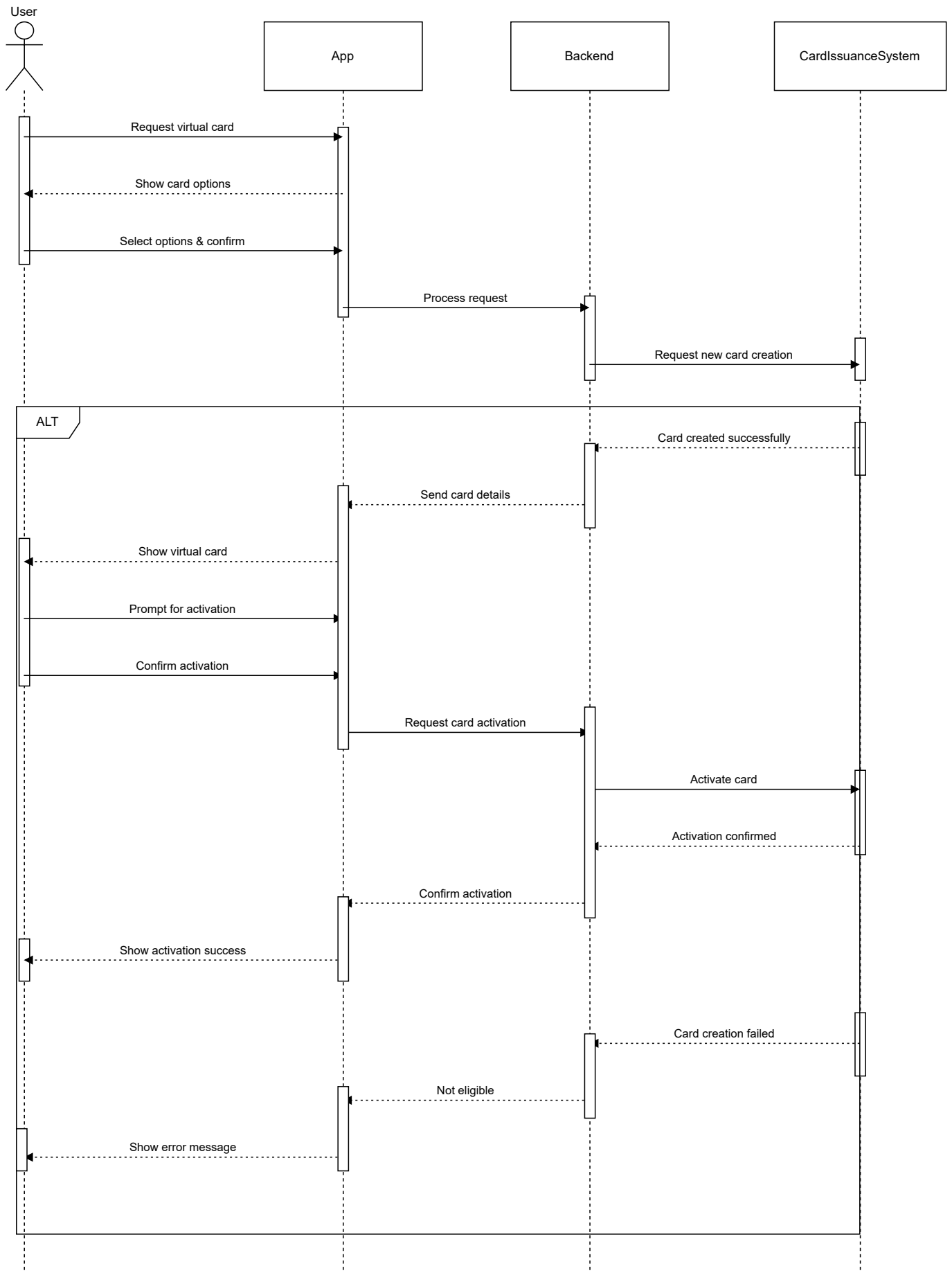

API
Database
NotificationSystem
Request Active Users Analysis
Query User Activity Data
Return Activity Records
User Activity Data
Identify Active Users
Rank Users by Activity
Identify Top Users
alt
[Activate Offers on Top Users]
Send Offer to Top Users
Store Offer Details
Confirm Storage
Offers Activated Confirmation
Update Active Users Report
Confirm Update

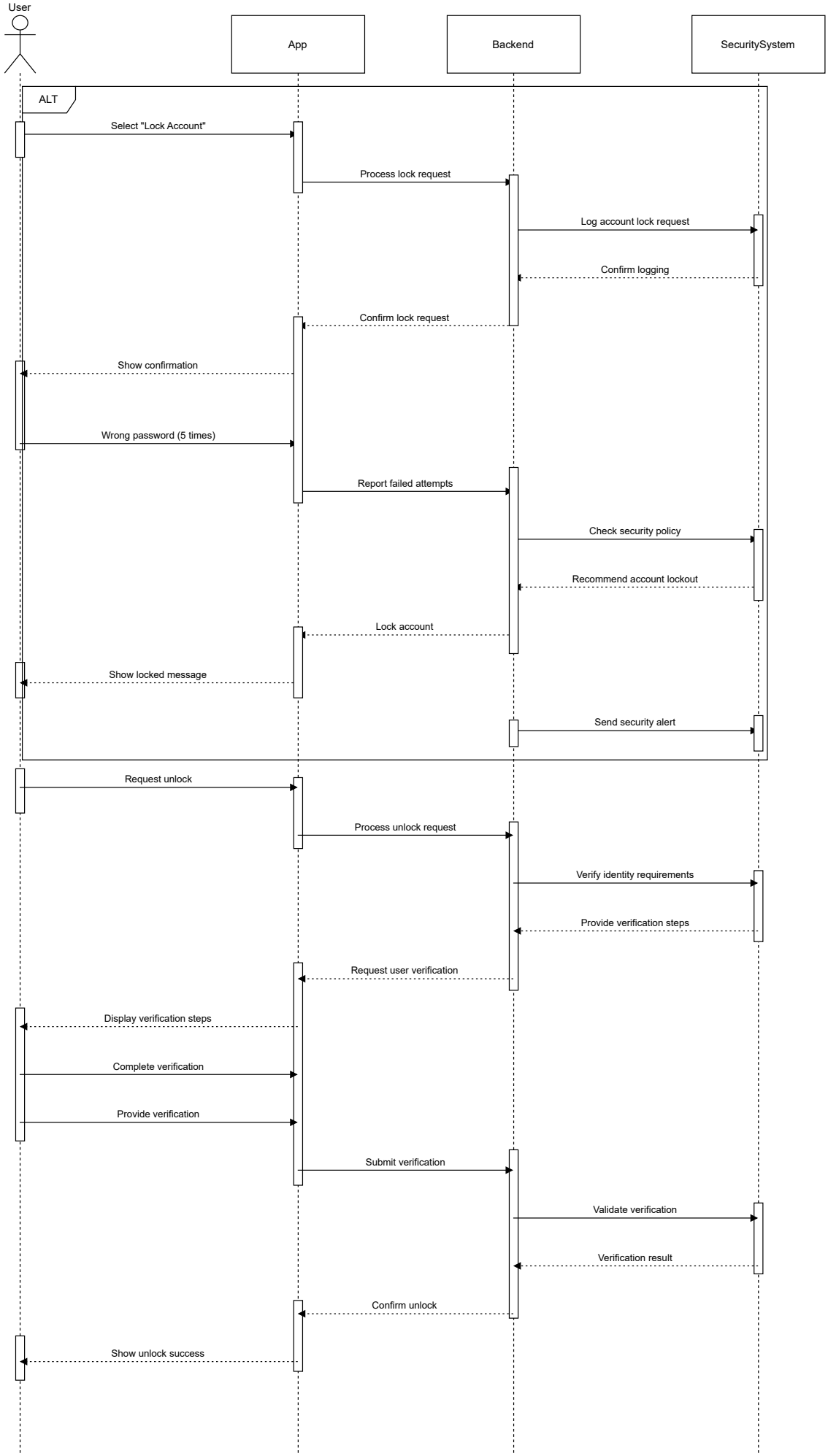

ui
API
Database
DataAnalysis
Request Pending Bills
Get Pending Bills(userID)
Query Unpaid Bills
Return Unpaid Bills
Pending Bills Data
Display Pending Bills
Log Bills Request
Update User Financial Profile
Confirm logging

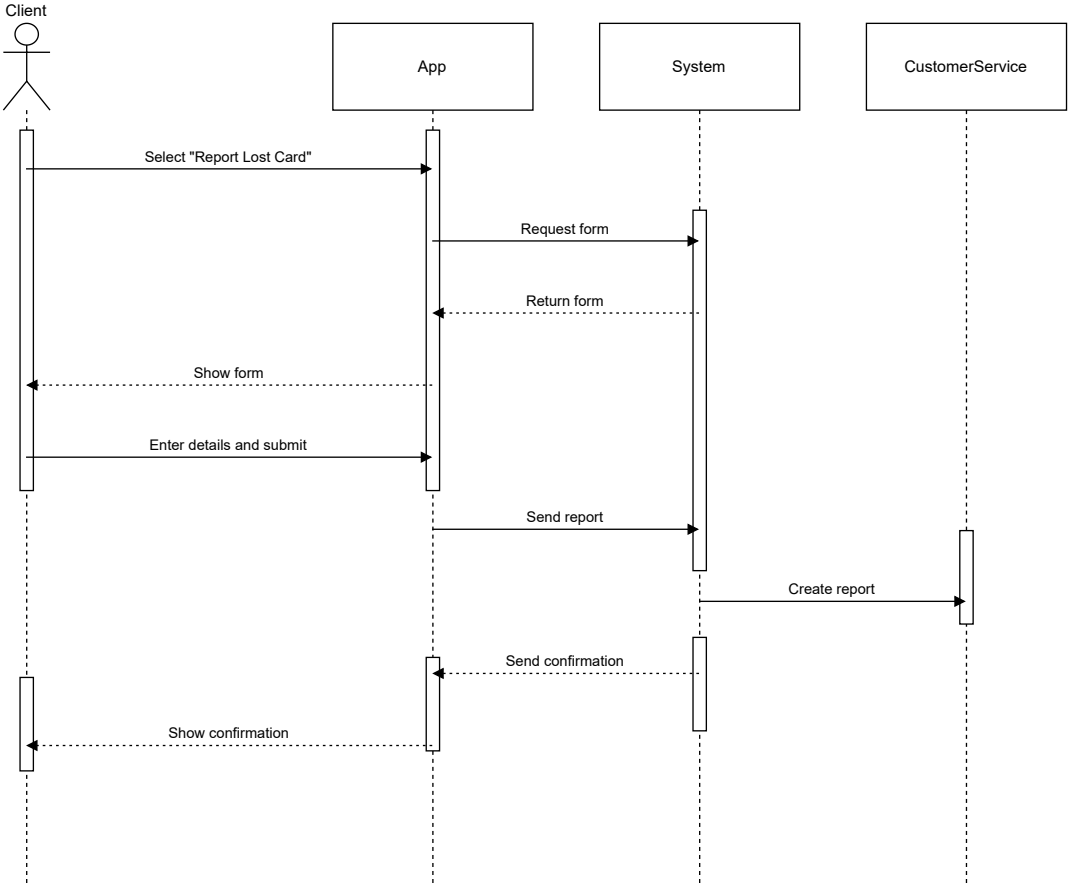

System
Database
DataAnalysis
NotificationSystem
Client
Check Account Status
Account Status Data
Analyze Account Activities
Check for Unusual Activities
Identify Low Balance Accounts
Check for Bill Due Dates
Alert Conditions
alt
[Alert Condition Detected]
Generate Alert
Store Alert Details
Confirm Storage
Send Alert Notification

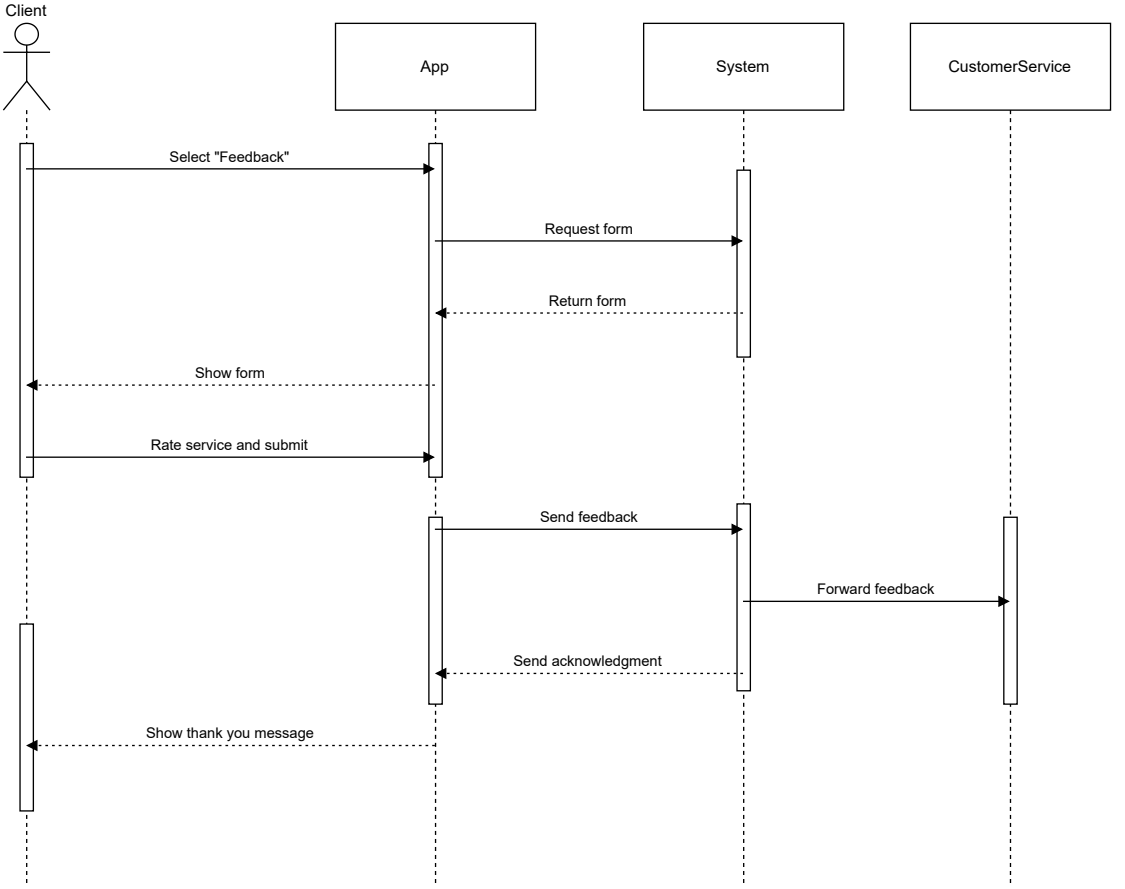

User
App
Backend
PaymentProcessor
Select QR payment
Open scanner
Scan QR code
Process QR code
Verify payment
Alt
Confirm payment
Payment successful
Show success message
Payment declined
Payment failed
Show error message

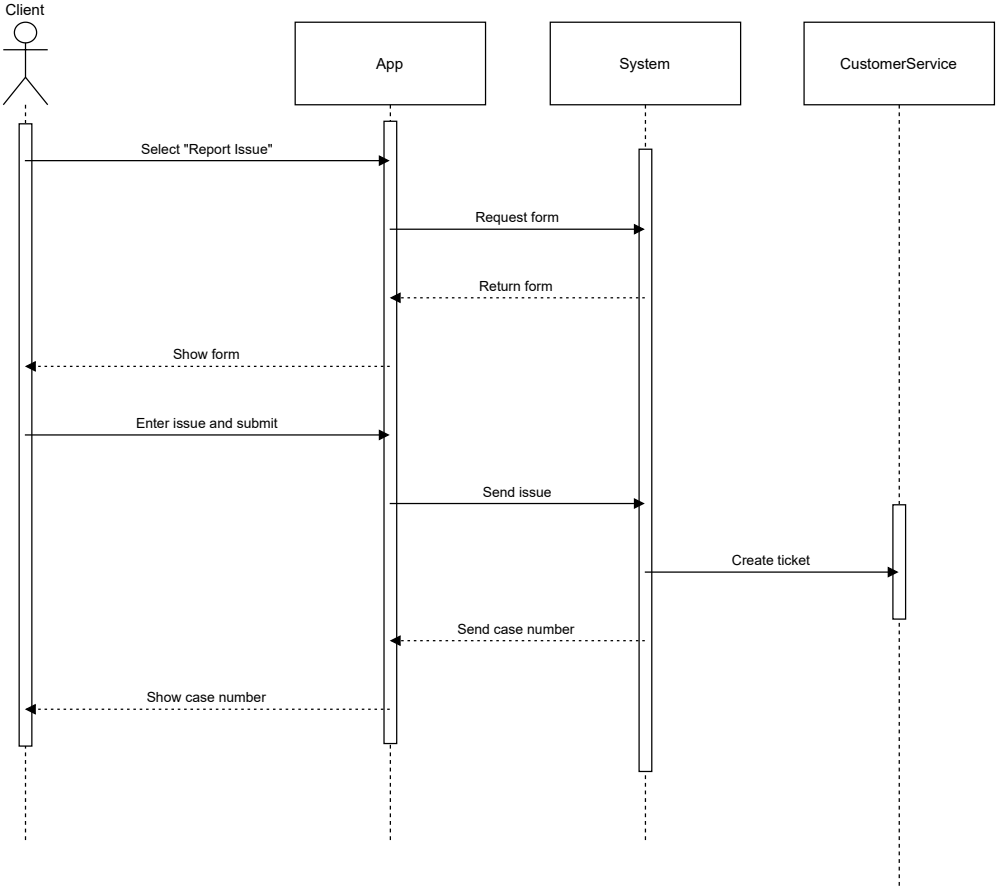

User
App
Backend
NotificationService
Start secure transaction
Request verification
Generate verification codes
ALT
Send code via email
Send code via SMS
Enter verification code
Verify code
Validate entered code
Validation result
ALT
Verification successful
Show success message
Verification failed
Show error message

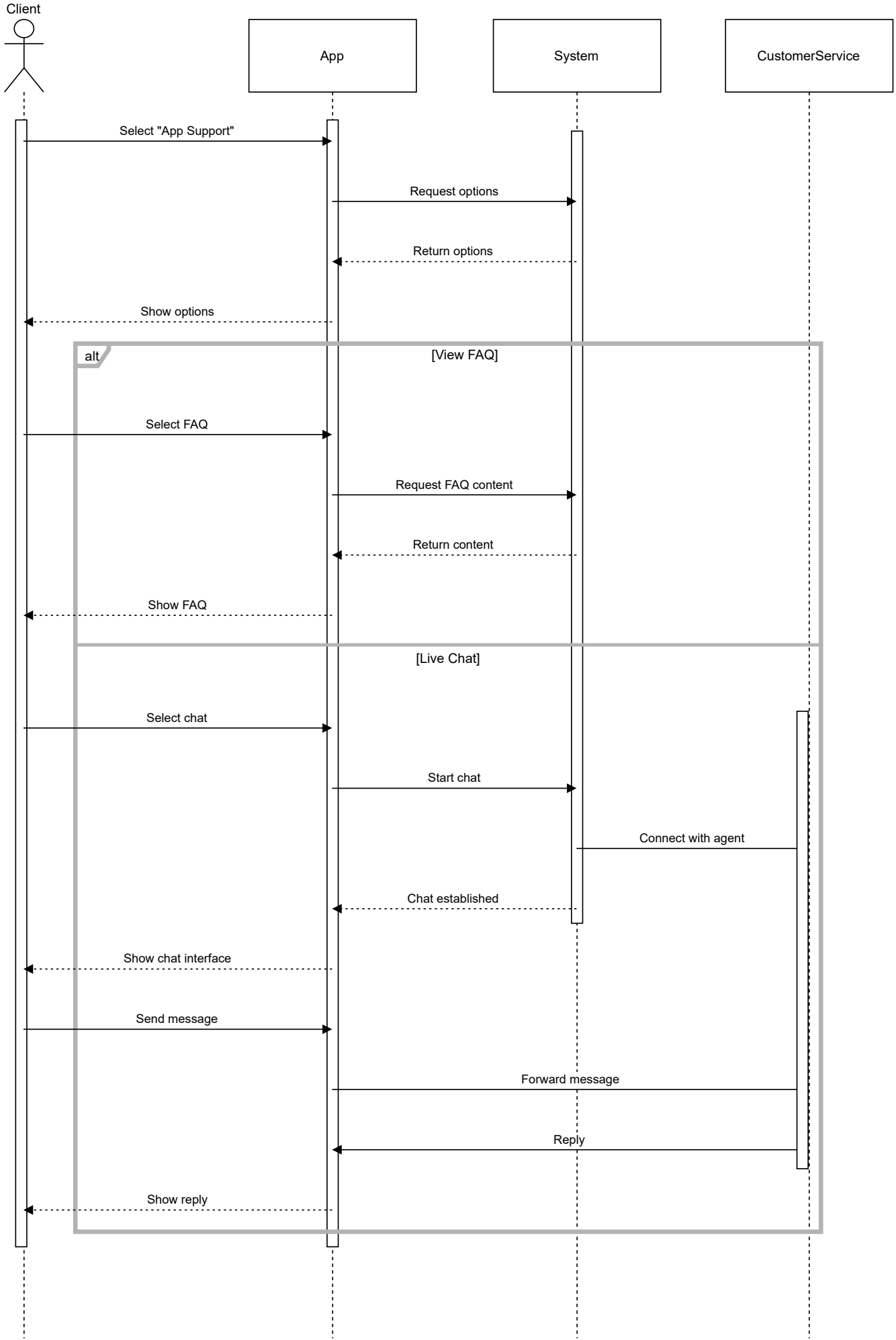

User
App
Backend
CardIssuanceSystem
Request virtual card
Show card options
Select options & confirm
Process request
Request new card creation
ALT
Card created successfully
Send card details
Show virtual card
Prompt for activation
Confirm activation
Request card activation
Activate card
Activation confirmed
Confirm activation
Show activation success
Card creation failed
Not eligible
Show error message

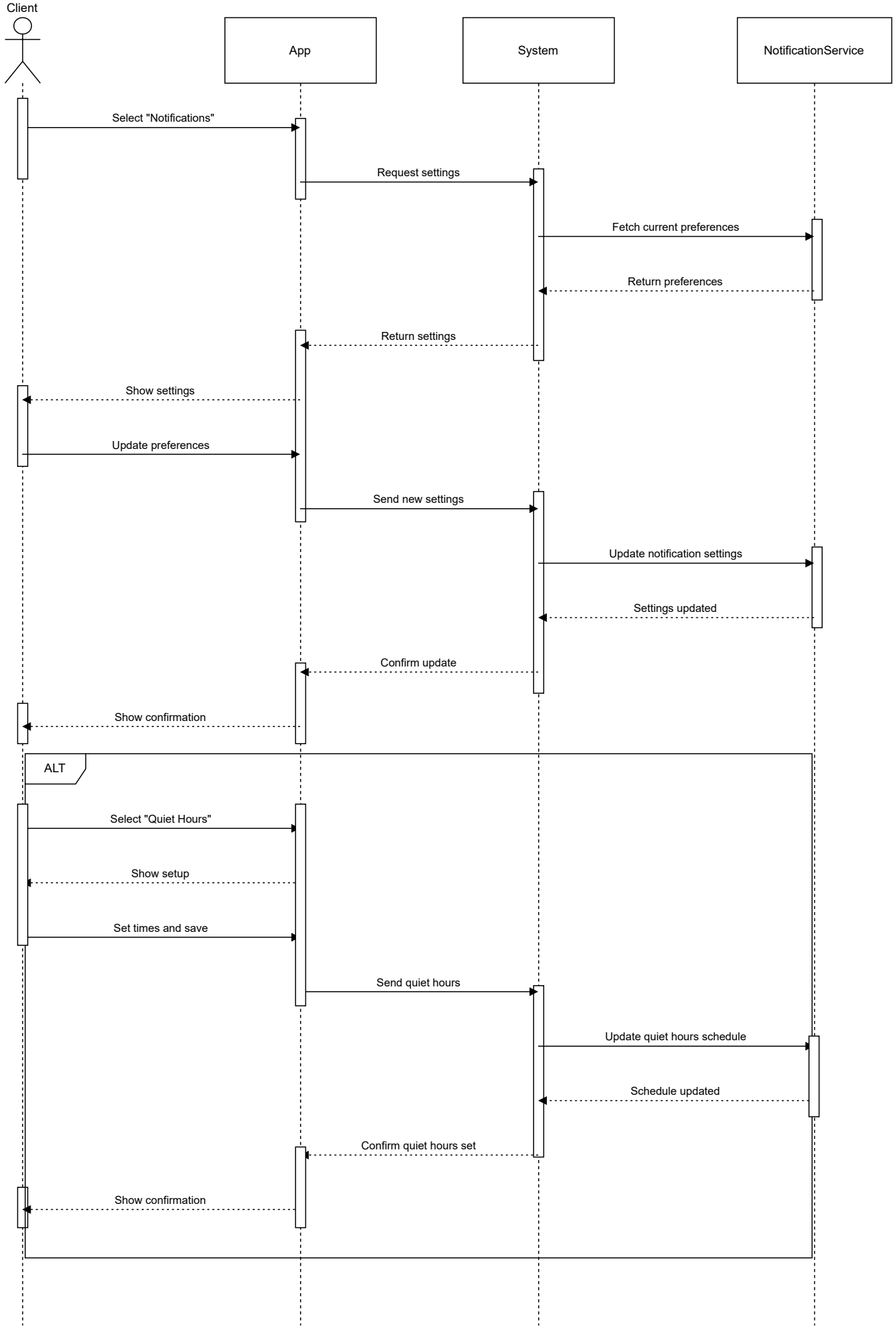

User
App
Backend
SecuritySystem
ALT
Select "Lock Account"
Process lock request
Log account lock request
Confirm logging
Confirm lock request
Show confirmation
Wrong password (5 times)
Report failed attempts
Check security policy
Recommend account lockout
Lock account
Show locked message
Send security alert
Request unlock
Process unlock request
Verify identity requirements
Provide verification steps
Request user verification
Display verification steps
Complete verification
Provide verification
Submit verification
Validate verification
Verification result
Confirm unlock
Show unlock success

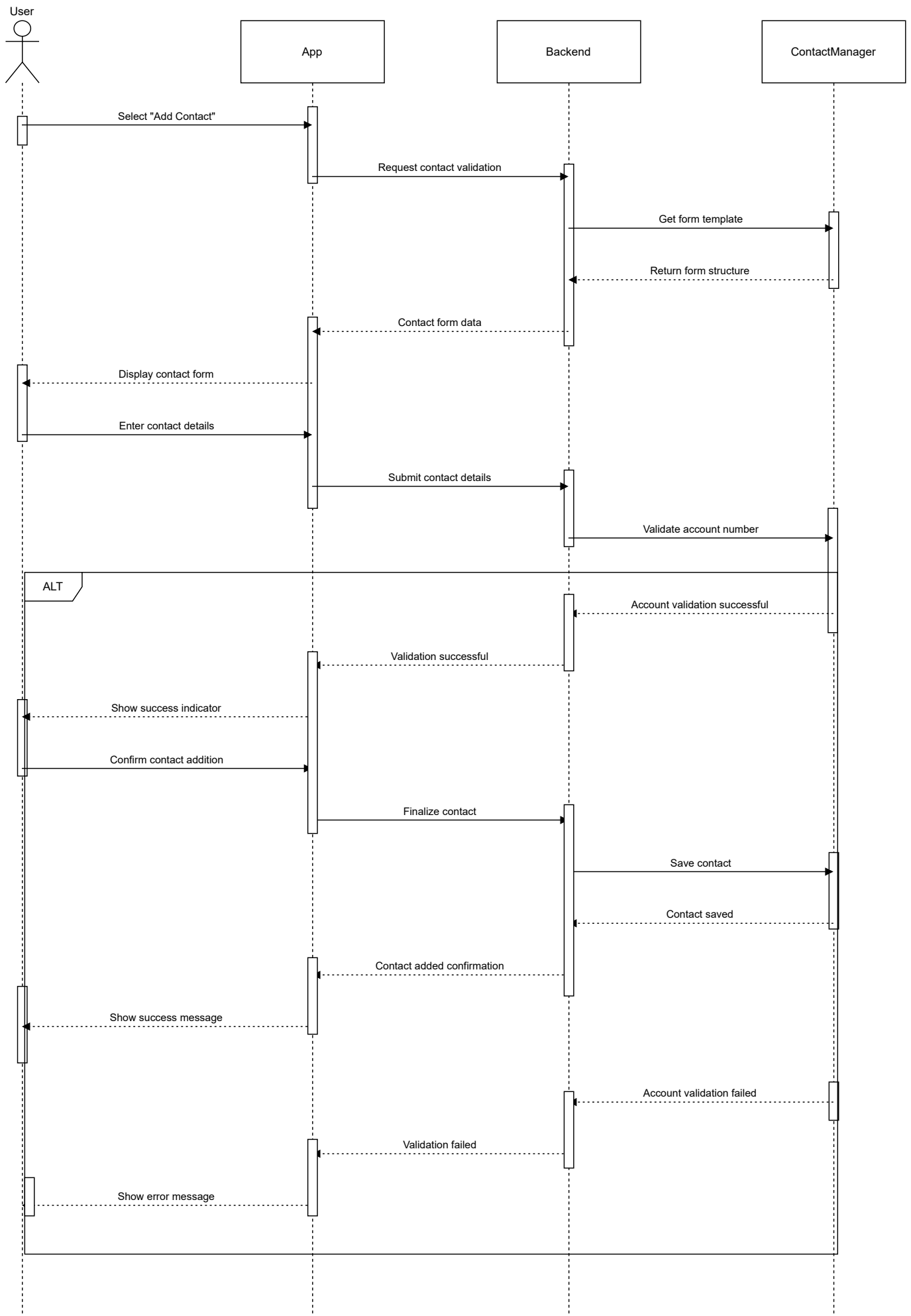

Client
App
System
CustomerService
Select "Report Lost Card"
Request form
Return form
Show form
Enter details and submit
Send report
Create report
Send confirmation
Show confirmation

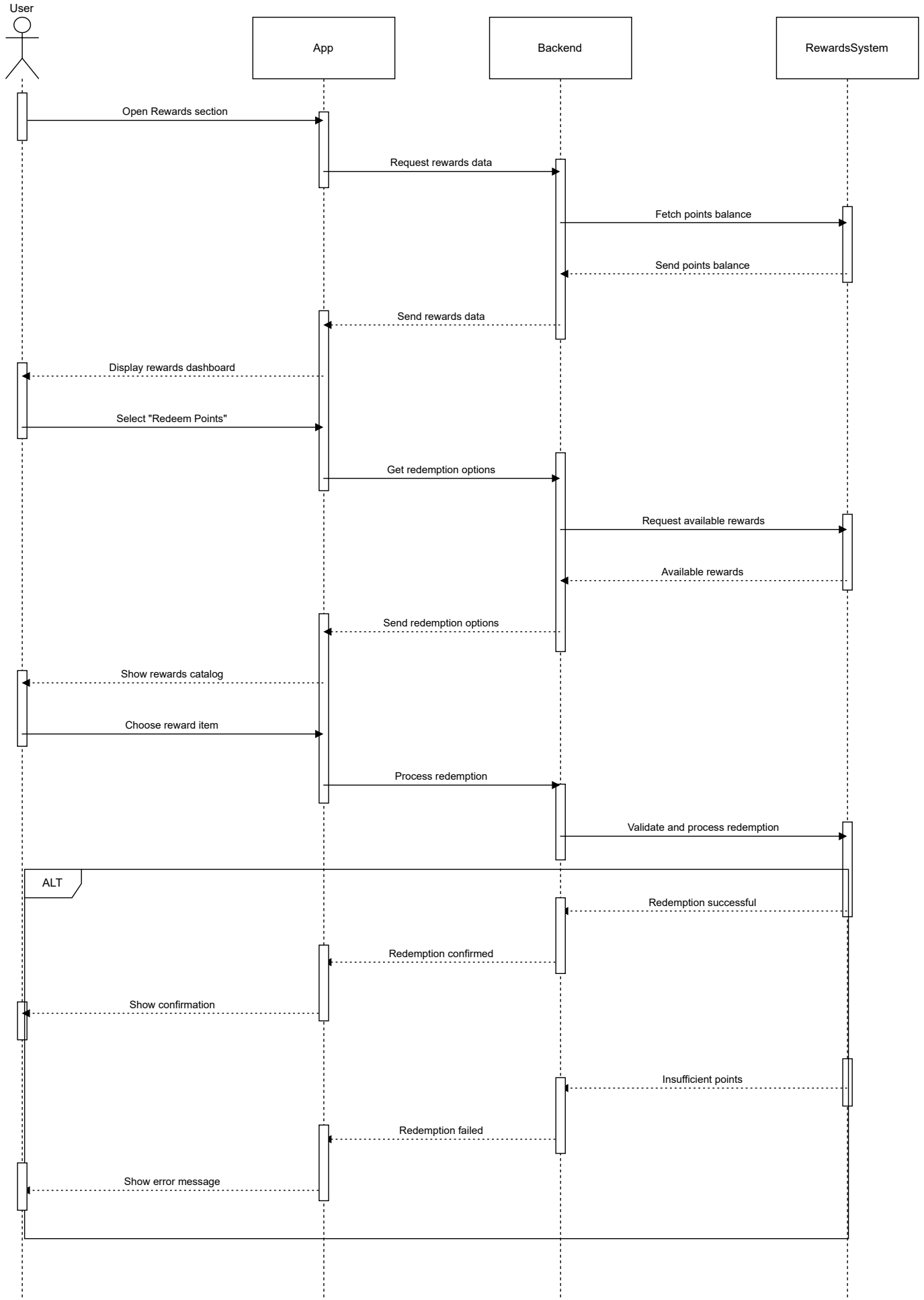

Client
App
System
CustomerService
Select "Feedback"
Request form
Return form
Show form
Rate service and submit
Send feedback
Forward feedback
Send acknowledgment
Show thank you message


Client
App
System
CustomerService
Select "Report Issue"
Request form
Return form
Show form
Enter issue and submit
Send issue
Create ticket
Send case number
Show case number

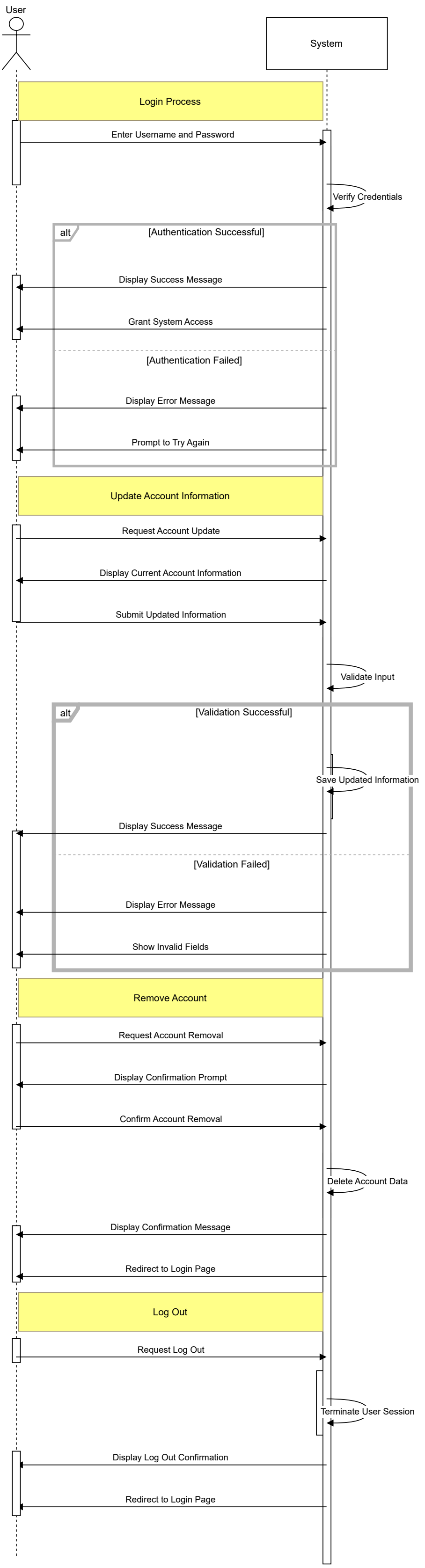

Client
App
System
CustomerService
Select "App Support"
Request options
Return options
Show options
alt
[View FAQ]
Select FAQ
Request FAQ content
Return content
Show FAQ
[Live Chat]
Select chat
Start chat
Connect with agent
Chat established
Show chat interface
Send message
Forward message
Reply
Show reply

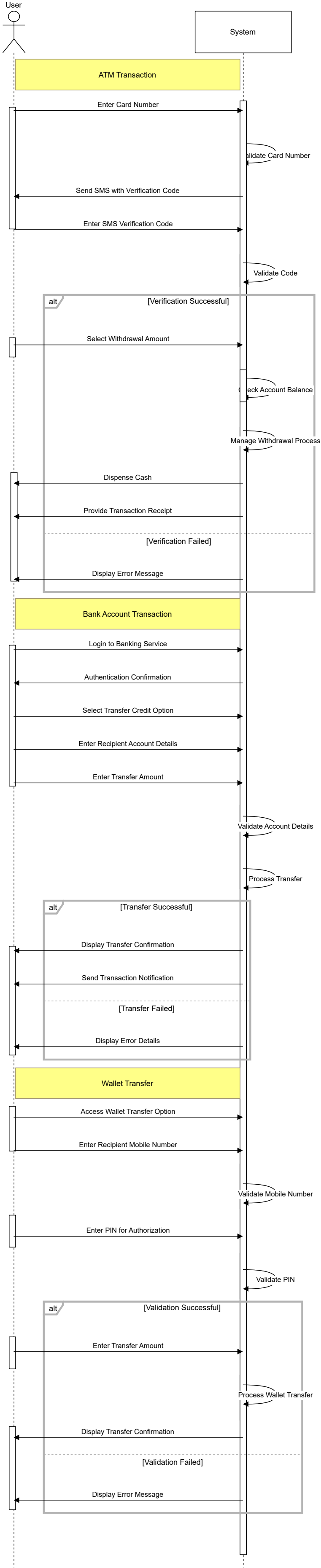

Client
App
System
NotificationService
Select "Notifications"
Request settings
Fetch current preferences
Return preferences
Return settings
Show settings
Update preferences
Send new settings
Update notification settings
Settings updated
Confirm update
Show confirmation
ALT
Select "Quiet Hours"
Show setup
Set times and save
Send quiet hours
Update quiet hours schedule
Schedule updated
Confirm quiet hours set
Show confirmation

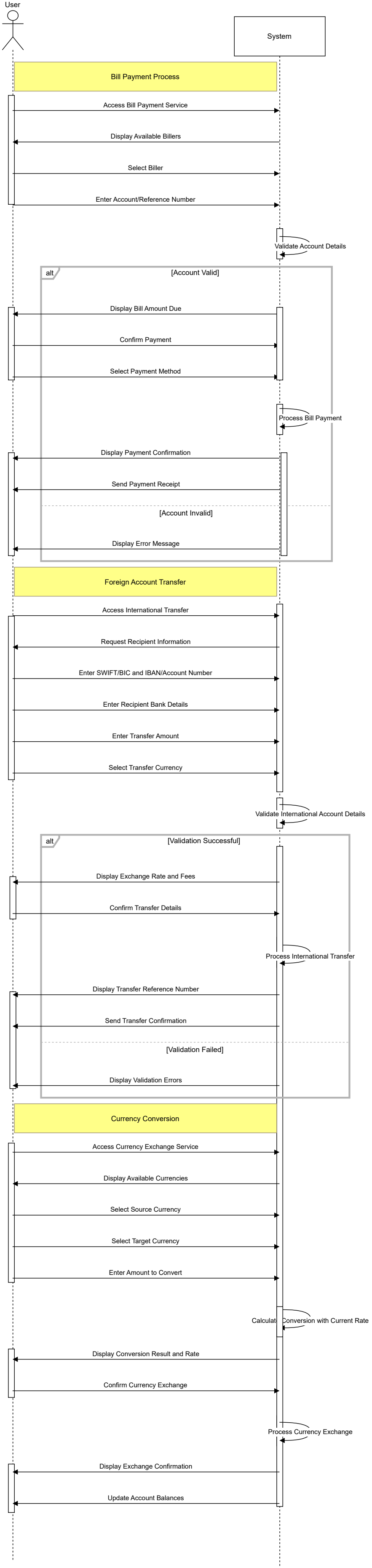

User
App
Backend
ContactManager
Select "Add Contact"
Request contact validation
Get form template
Return form structure
Contact form data
Display contact form
Enter contact details
Submit contact details
Validate account number
ALT
Account validation successful
Validation successful
Show success indicator
Confirm contact addition
Finalize contact
Save contact
Contact saved
Contact added confirmation
Show success message
Account validation failed
Validation failed
Show error message

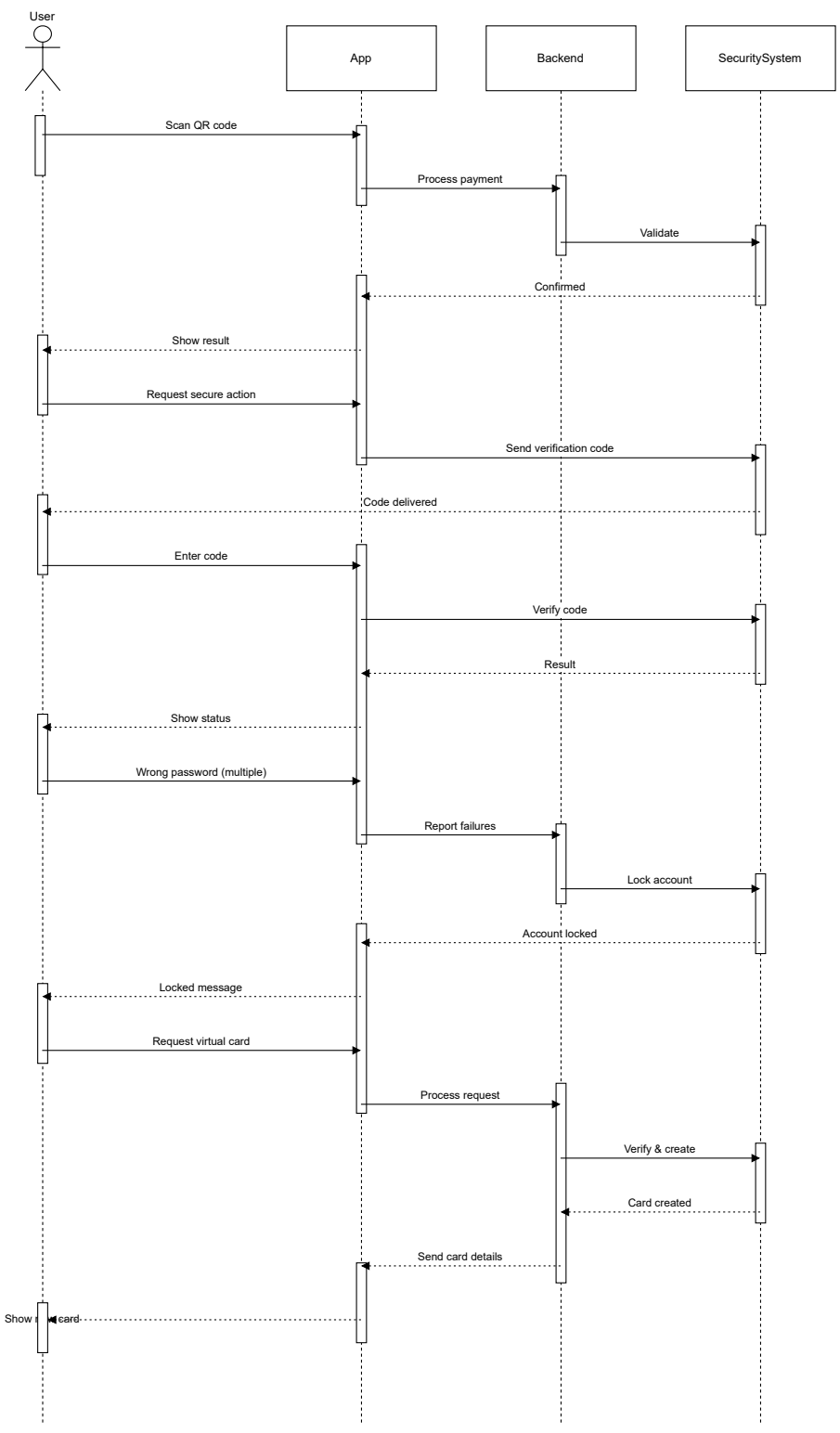

User
App
Backend
RewardsSystem
Open Rewards section
Request rewards data
Fetch points balance
Send points balance
Send rewards data
Display rewards dashboard
Select "Redeem Points"
Get redemption options
Request available rewards
Available rewards
Send redemption options
Show rewards catalog
Choose reward item
Process redemption
Validate and process redemption
ALT
Redemption successful
Redemption confirmed
Show confirmation
Insufficient points
Redemption failed
Show error message

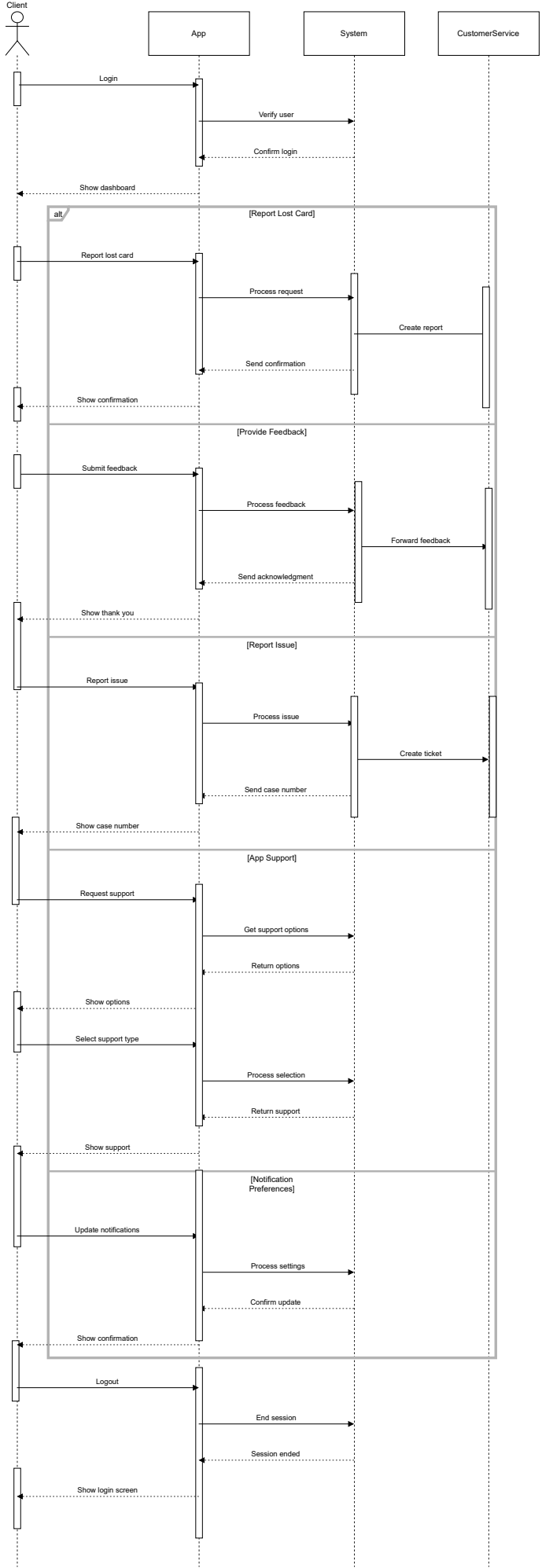
System Sequence Diagrams (SSDs) :



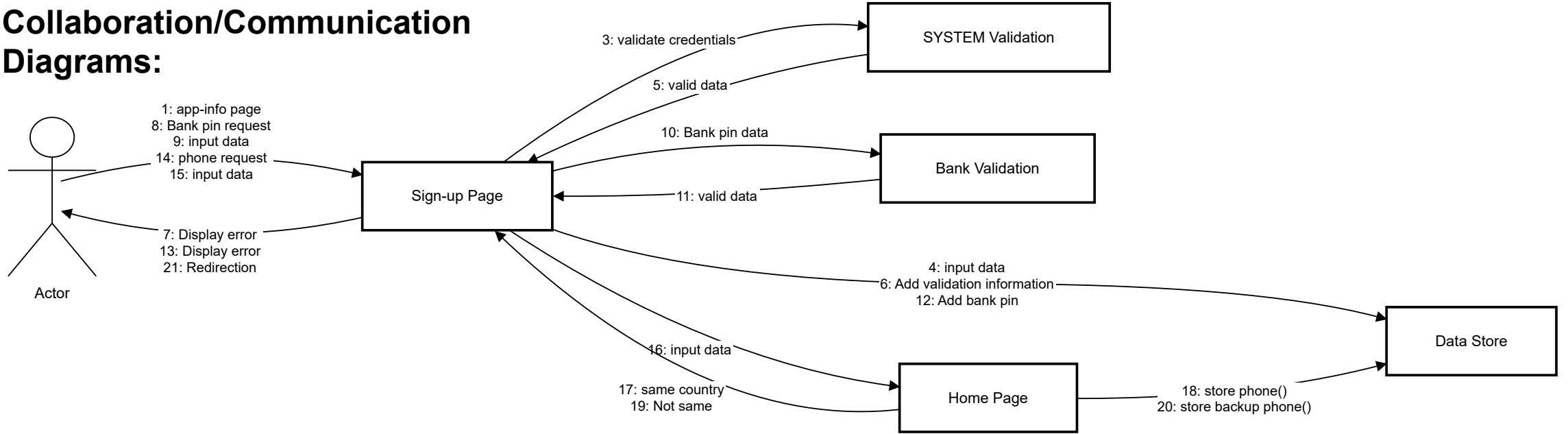

User
System
ATM Transaction
Enter Card Number
alidate Card Number
Send SMS with Verification Code
Enter SMS Verification Code
Validate Code
alt
[Verification Successful]
Select Withdrawal Amount
eck Account Balance
Manage Withdrawal Process
Dispense Cash
Provide Transaction Receipt
[Verification Failed]
Display Error Message
Bank Account Transaction
Login to Banking Service
Authentication Confirmation
Select Transfer Credit Option
Enter Recipient Account Details
Enter Transfer Amount
Validate Account Details
Process Transfer
alt
[Transfer Successful]
Display Transfer Confirmation
Send Transaction Notification
[Transfer Failed]
Display Error Details
Wallet Transfer
Access Wallet Transfer Option
Enter Recipient Mobile Number
Validate Mobile Number
Enter PIN for Authorization
Validate PIN
alt
[Validation Successful]
Enter Transfer Amount
Process Wallet Transfer
Display Transfer Confirmation
[Validation Failed]
Display Error Message

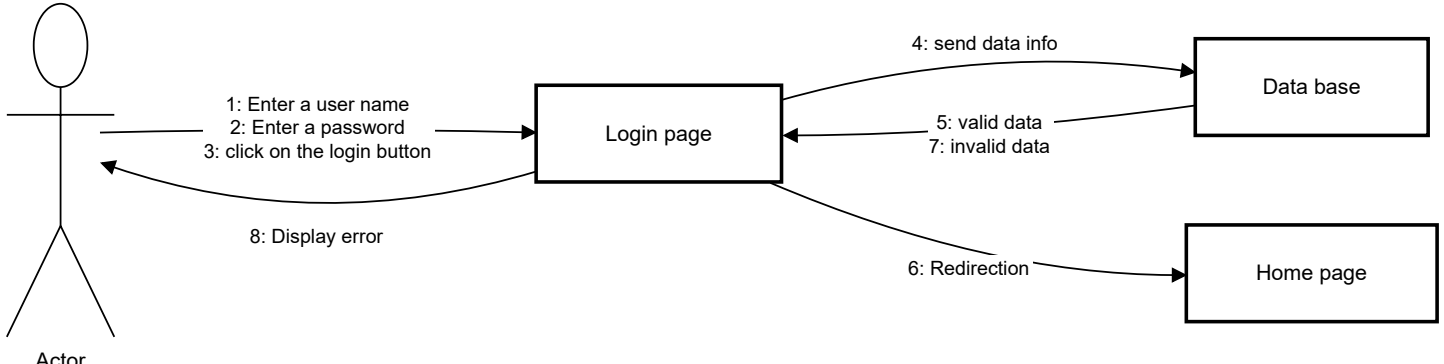

User
System
Bill Payment Process
Access Bill Payment Service
Display Available Billers
Select Biller
Enter Account/Reference Number
Validate Account Details
alt
[Account Valid]
Display Bill Amount Due
Confirm Payment
Select Payment Method
Process Bill Payment
Display Payment Confirmation
Send Payment Receipt
[Account Invalid]
Display Error Message
Foreign Account Transfer
Access International Transfer
Request Recipient Information
Enter SWIFT/BIC and IBAN/Account Number
Enter Recipient Bank Details
Enter Transfer Amount
Select Transfer Currency
Validate International Account Details
alt
[Validation Successful]
Display Exchange Rate and Fees
Confirm Transfer Details
Process International Transfer
Display Transfer Reference Number
Send Transfer Confirmation
[Validation Failed]
Display Validation Errors
Currency Conversion
Access Currency Exchange Service
Display Available Currencies
Select Source Currency
Select Target Currency
Enter Amount to Convert
Calculate Conversion with Current Rate
Display Conversion Result and Rate
Confirm Currency Exchange
Process Currency Exchange
Display Exchange Confirmation
Update Account Balances

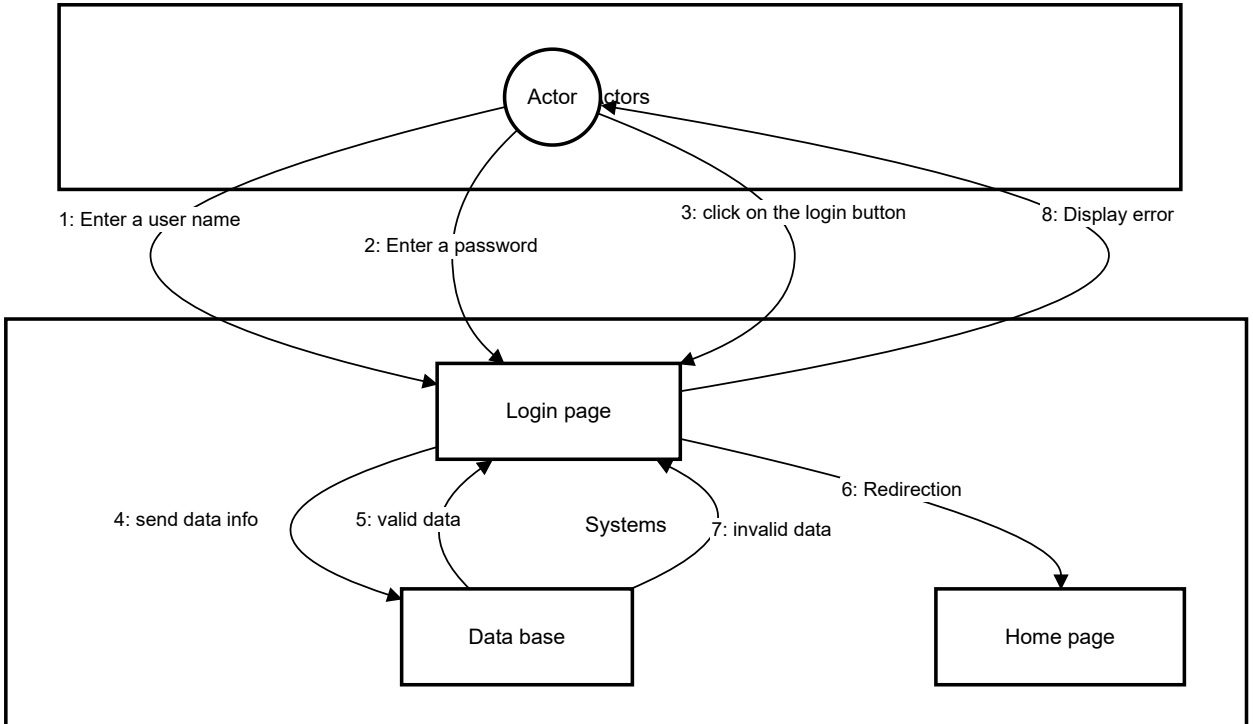

User
App
Backend
SecuritySystem
Scan QR code
Process payment
Validate
Confirmed
Show result
Request secure action
Send verification code
Code delivered
Enter code
Verify code
Result
Show status
Wrong password (multiple)
Report failures
Lock account
Account locked
Locked message
Request virtual card
Process request
Verify & create
Card created
Send card details
Show
card

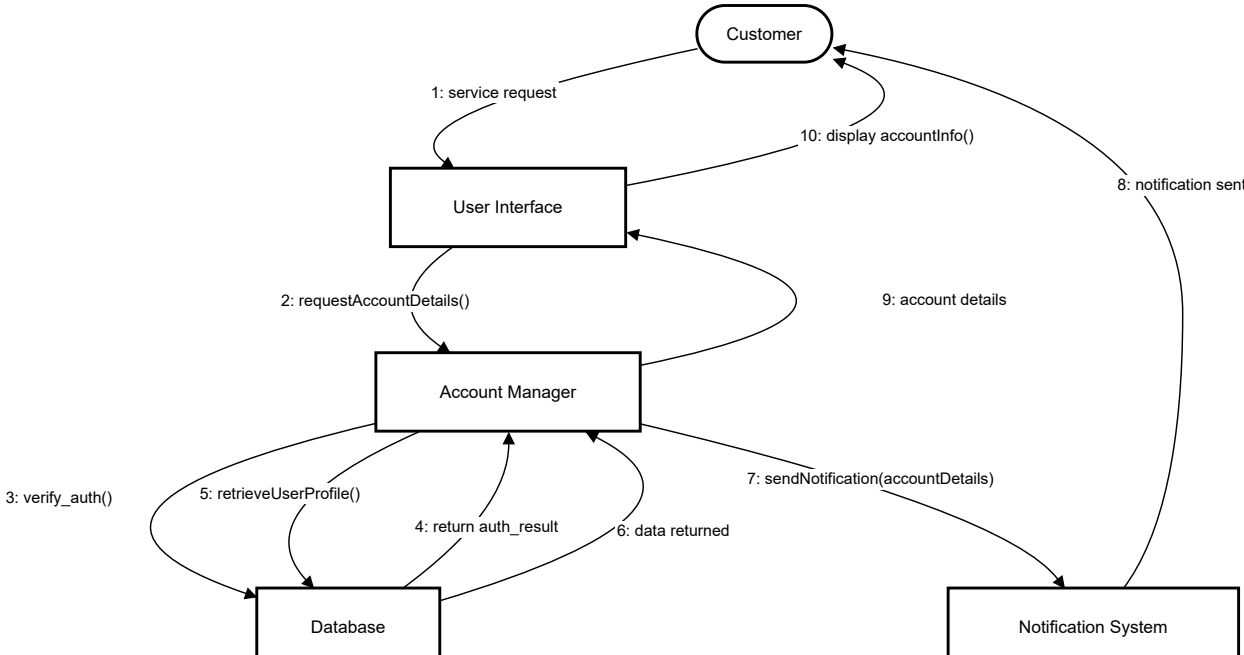

Client
App
System
CustomerService
Login
Verify user
Confirm login
Show dashboard
alt
[Report Lost Card]
Report lost card
Process request
Create report
Send confirmation
Show confirmation
[Provide Feedback]
Submit feedback
Process feedback
Forward feedback
Send acknowledgment
Show thank you
[Report Issue]
Report issue
Process issue
Create ticket
Send case number
Show case number
[App Support]
Request support
Get support options
Return options
Show options
Select support type
Process selection
Return support
Show support
[Notification Preferences]
Update notifications
Process settings
Confirm update
Show confirmation
Logout
End session
Session ended
Show login screen

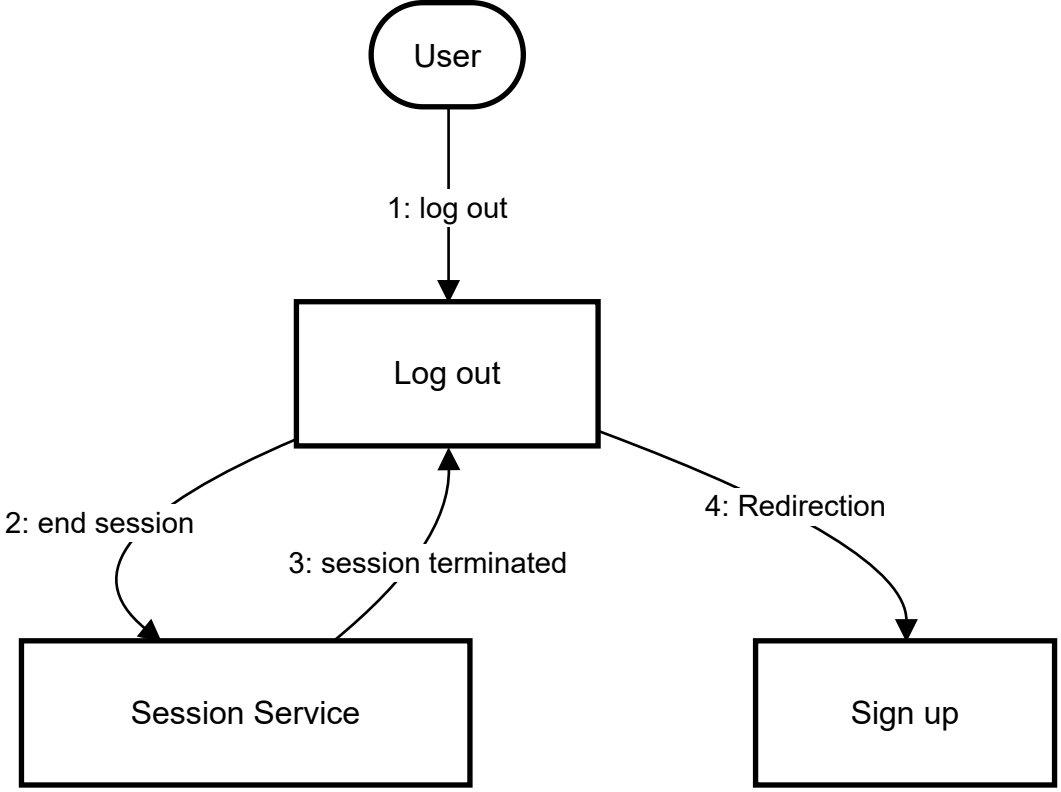
Collaboration/Communication Diagrams:

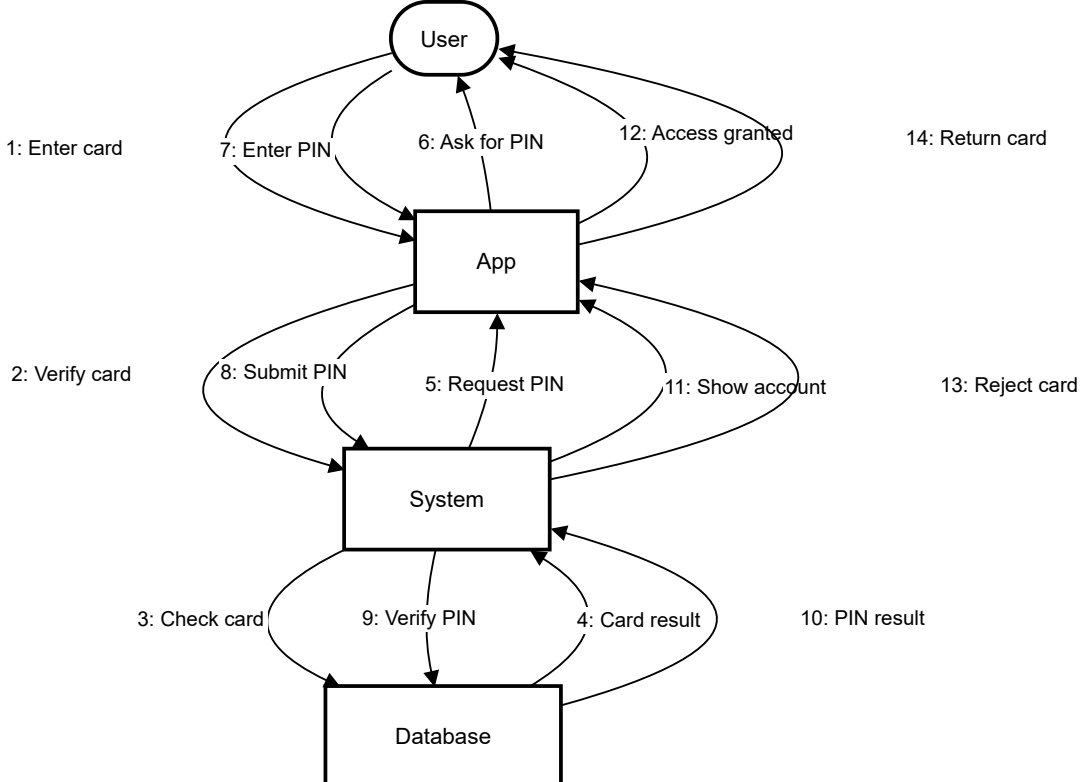


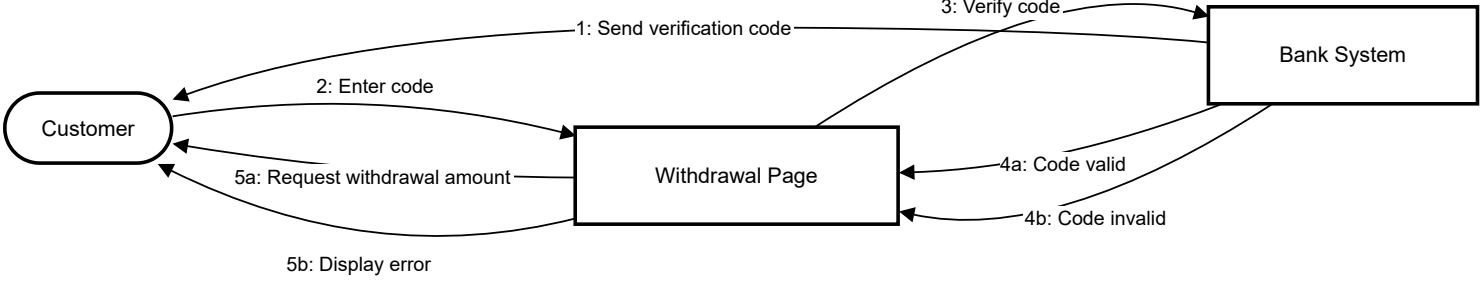

Actor
1: Enter a user name
2: Enter a password
3: click on the login button
Login page
4: send data info
Data base
5: valid data
7: invalid data
8: Display error
6: Redirection
Home page

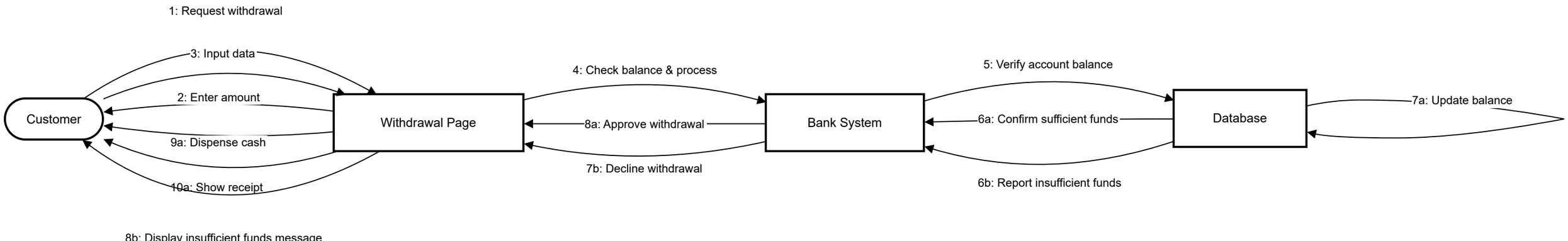

Actor
ctors
1: Enter a user name
2: Enter a password
3: click on the login button
8: Display error
Login page
4: send data info
5: valid data
Systems
7: invalid data
6: Redirection
Data base
Home page

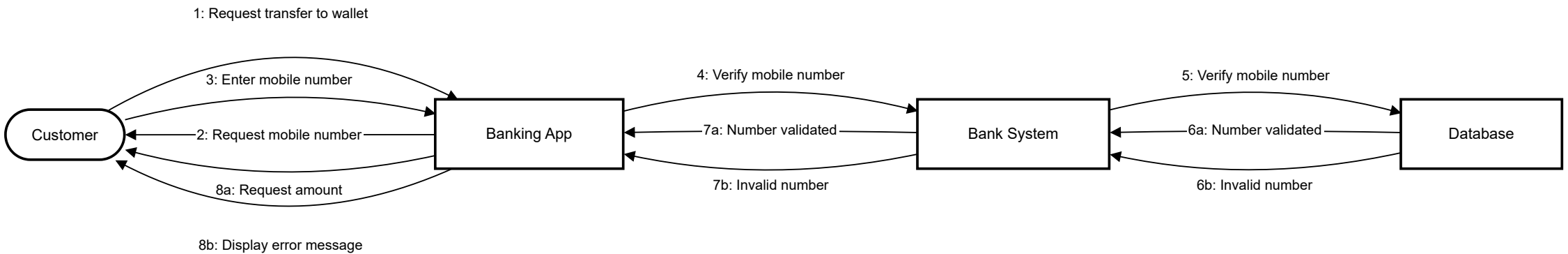

Customer
1: service request
10: display accountInfo()
8: notification sent
User Interface
2: requestAccountDetails()
9: account details
Account Manager
3: verify_auth()
5: retrieveUserProfile()
4: return auth_result
6: data returned
7: sendNotification(accountDetails)
Database
Notification System

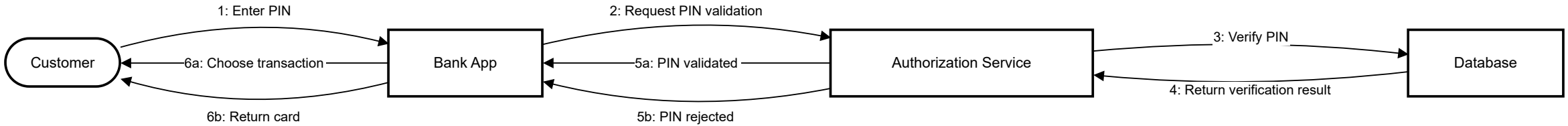

User
1: log out
Log out
2: end session
3: session terminated
4: Redirection
Session Service
Sign up


User
1: Enter card
7: Enter PIN
6: Ask for PIN
12: Access granted
14: Return card
App
2: Verify card
8: Submit PIN
5: Request PIN
11: Show account
13: Reject card
System
3: Check card
9: Verify PIN
4: Card result
10: PIN result
Database


3: Verify code
1: Send verification code
Bank System
2: Enter code
Customer
Withdrawal Page
4a: Code valid
5a: Request withdrawal amount
4b: Code invalid
5b: Display error


1: Request withdrawal
3: Input data
2: Enter amount
9a: Dispense cash
10a: Show receipt
Customer
Withdrawal Page
4: Check balance & process
8a: Approve withdrawal
7b: Decline withdrawal
Bank System
5: Verify account balance
6a: Confirm sufficient funds
6b: Report insufficient funds
Database
7a: Update balance
8b: Display insufficient funds message


1: Request transfer to wallet
3: Enter mobile number
4: Verify mobile number
5: Verify mobile number
Customer
2: Request mobile number
Banking App
7a: Number validated
Bank System
6a: Number validated
Database
8a: Request amount
7b: Invalid number
6b: Invalid number
8b: Display error message


1: Enter PIN
2: Request PIN validation
Customer
6a: Choose transaction
Bank App
5a: PIN validated
Authorization Service
3: Verify PIN
Database
4: Return verification result
6b: Return card
5b: PIN rejected

Use Case Implementation Strategy Analysis

Introduction

This document analyzes the implementation strategy used for the banking system project based on the provided class diagram and use case diagram. The analysis focuses on determining which approach was used to implement the use cases: One Central Class, Actor Class, or Use-Case Class, and discusses the advantages and disadvantages of the chosen approach.

Analysis of Implementation Strategy

After examining the class and use case diagrams, it is evident that the project primarily implements a **Use-Case Class strategy** with elements of the **Actor Class strategy**. This hybrid approach can be observed through several key aspects of the design:

Evidence of Use-Case Class Strategy

1. **Transaction-oriented classes**: Classes like `Transaction`, `Feedback`, and `Notification` are designed around specific use cases rather than actors or a central controller.
2. **Specialized functionality**: Each class encapsulates the behavior required for specific use cases (e.g., the `Wallet` class handles wallet-specific operations, and the `Bank_Account` class manages banking operations).
3. **Distributed responsibility**: Business logic is spread across multiple classes that each handle their specific domain of functionality.

Elements of Actor Class Strategy

1. **Actor-based classes**: Classes like `Client`, `Customer_surver` (Customer Service), and `Security` represent specific actors in the system.
2. **Role-specific behaviors**: The methods and attributes of these classes align with the responsibilities of specific roles within the system.

Advantages of the Chosen Strategy

Advantages of Use-Case Class Strategy

1. **High cohesion**: Each class focuses on a specific set of related functionalities, making the code more maintainable.
2. **Modularity**: Use cases can be developed, tested, and modified independently of each other.

3. **Extensibility**: New use cases can be added by introducing new classes without significantly modifying existing code.
4. **Clear responsibilities**: Each class has a well-defined purpose that aligns with business requirements.
5. **Reduced coupling**: Classes interact through well-defined interfaces, reducing dependencies between components.

Advantages of Actor Class Elements

1. **User-centered design**: The inclusion of actor classes ensures that user roles and permissions are clearly represented in the system architecture.
2. **Simplified access control**: Actor classes can encapsulate role-specific behaviors and permissions.
3. **Intuitive mapping**: The system structure maps naturally to the organizational structure of users.

Disadvantages of the Chosen Strategy

Disadvantages of Use-Case Class Strategy

1. **Potential for duplication**: Similar functionalities might be implemented across different use case classes, leading to code duplication.
2. **Complex interactions**: As the number of classes increases, managing interactions between them can become complex.
3. **Overhead**: Creating a new class for each use case can lead to a large number of classes, increasing the cognitive load for developers.
4. **Integration challenges**: Ensuring consistent behavior across use case classes may require additional coordination effort.

Disadvantages of Actor Class Elements

1. **Changing user roles**: If the business requirements change and user roles are redefined, the class structure may need significant refactoring.
2. **Blurred responsibilities**: Actor classes may take on too many responsibilities if not carefully designed, leading to lower cohesion.

Alternative Approaches

One Central Class Approach (Not Used)

The One Central Class approach would have centered all functionality around a single controlling class. This approach was not chosen for this project, which is beneficial because:

1. **Avoids "god class" problems**: The current design avoids creating a single overly complex class that would be difficult to maintain.
2. **Better distribution of responsibilities**: The current approach allows for better separation of concerns and more focused classes.
3. **Improved scalability**: Distributed functionality allows for easier scaling and maintenance of individual components.

Conclusion

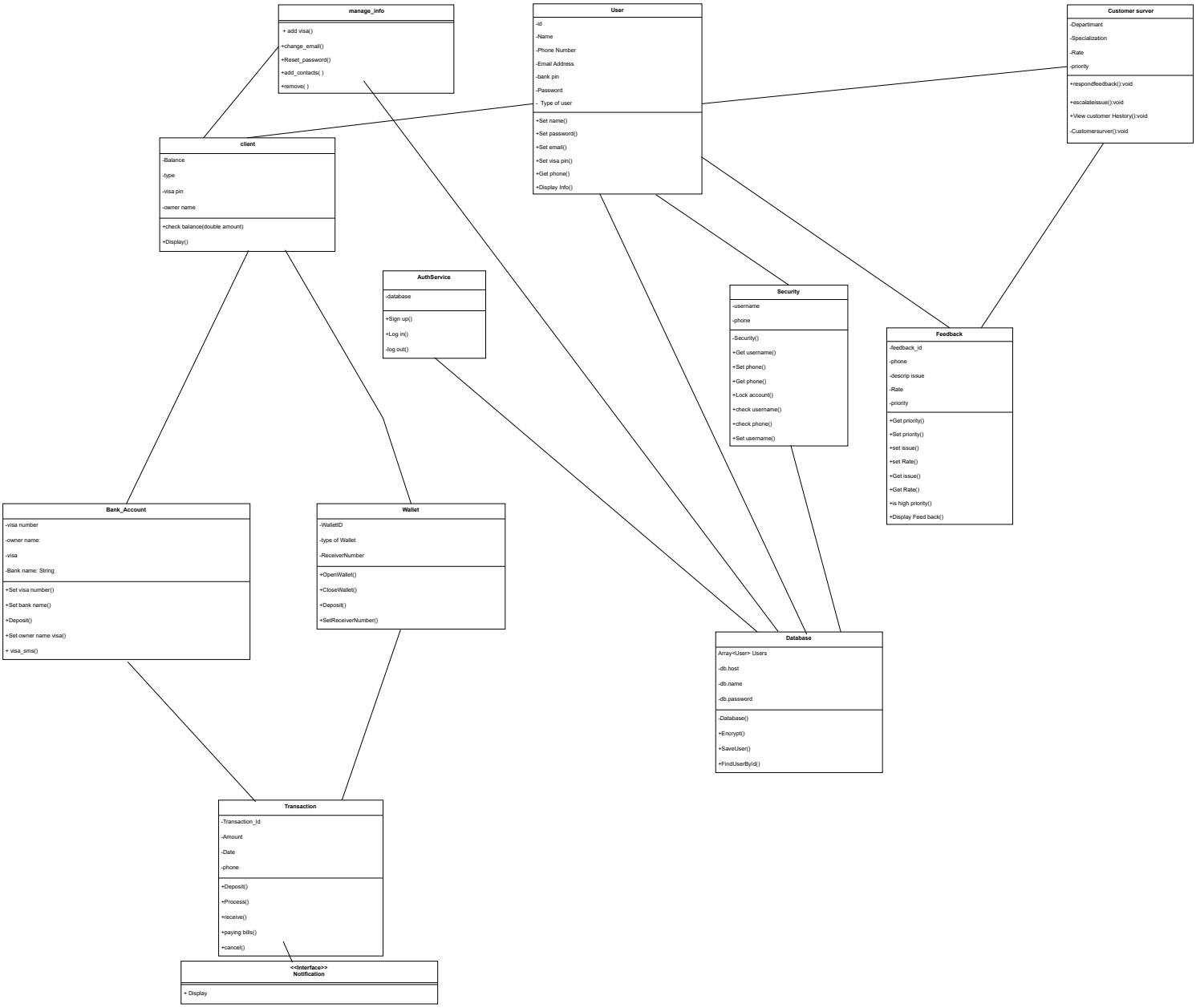
The banking system project implements a hybrid strategy that primarily follows the Use-Case Class approach with elements of the Actor Class strategy. This combination provides a balance between user-centered design and functional organization.

The approach offers advantages in terms of modularity, cohesion, and clear separation of concerns, which are particularly valuable for a banking system where security, transaction integrity, and user role management are critical. However, it also presents challenges in terms of potential code duplication and managing complex interactions between classes.

As the system evolves, maintaining clear boundaries between classes and ensuring consistent behavior across use cases will be important for long-term maintainability. Special attention should be paid to cross-cutting concerns like security and notification handling to ensure they are implemented consistently across all relevant use cases.

Overall, the chosen strategy aligns well with the complex requirements of a banking system, providing a flexible and maintainable architecture that can accommodate future changes and extensions.

Class Diagram 2 :


manage_info
+ add visa()
+change_email()
+Reset_password()
+add_contacts( )
+remove( )
User
-Id
-Name
-Phone Number
-Email Address
-bank pin
-Password
- Type of user
+Set name()
+Set password()
+Set email()
+Set visa pin()
+Get phone()
+Display Info()
Customer surver
-Departimant
-Specialization
-Rate
-priority
+respondfeedback():void
+escalateissue():void
+View customer Hestory():void
-Customersurver():void
client
-Balance
-type
-visa pin
-owner name
+check balance(double amount)
+Display()
AuthService
-database
+Sign up()
+Log in()
-log out()
Security
-username
-phone
-Security()
+Get username()
+Set phone()
+Get phone()
+Lock account()
+check username()
+check phone()
+Set username()
Feedback
-feedback_id
-phone
-descrip issue
-Rate
-priority
+Get priority()
+Set priority()
+set issue()
+set Rate()
+Get issue()
+Get Rate()
+is high priority()
+Display Feed back()
Bank_Account
-visa number
-owner name:
-visa
-Bank name: String
+Set visa number()
+Set bank name()
+Deposit()
+Set owner name visa()
+ visa_sms()
Wallet
-WalletID
-type of Wallet
-ReceiverNumber
+OpenWallet()
+CloseWallet()
+Deposit()
+SetReceiverNumber()
Database
Array<User> Users
-db.host
-db.name
-db.password
-Database()
+Encrypt()
+SaveUser()
+FindUserById()
Transaction
-Transaction_Id
-Amount
-Date
-phone
+Deposit()
+Process()
+receive()
+paying bills()
+cancel()
<<Interface>>
Notification
+ Display

Design Patterns Description

Name : Facade Design Pattern.

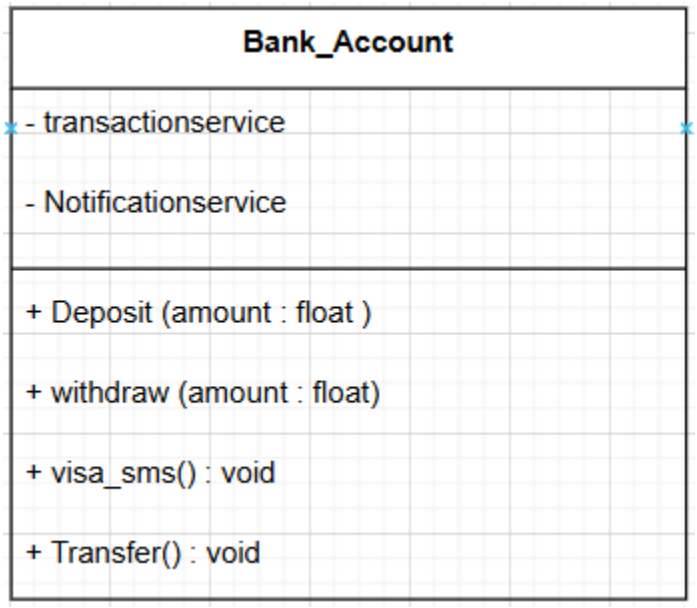
Context : A Bank_Account class that provides a simplified interface to complex banking operations.

Problem : How do you provide a simple, unified interface to a complex subsystem of banking operations without exposing implementation details?

Forces : There must be a way to hide the complexities of the banking system while providing users with easy-to-use methods for common banking operations.

Solution : Create a Bank_Account facade that provides simplified methods like Deposit() and visa_sms() that internally handle the complex interactions with various banking subsystems (transaction processing, notification services, etc.).

References : The Facade pattern is one of the structural patterns introduced by the "Gang of Four" in their influential work on design patterns.



Design Patterns Description

Name : Facade Design Pattern.

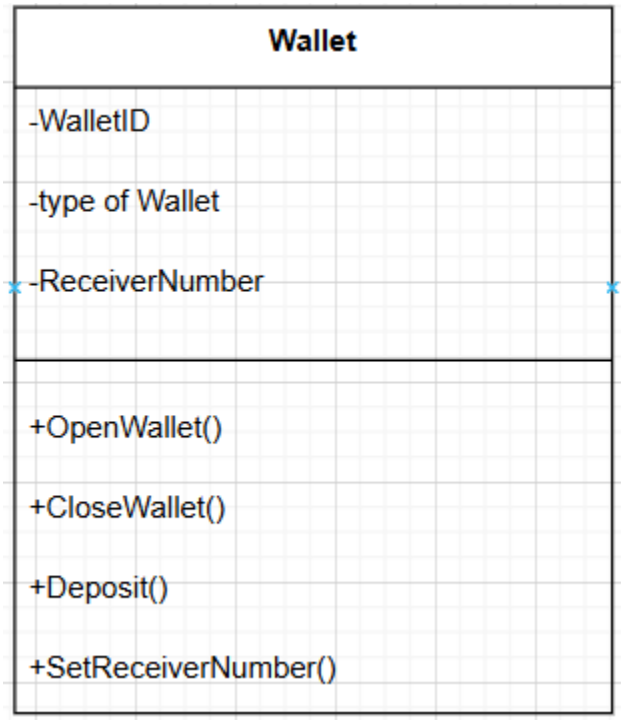
Context : A Bank_Account class that provides a simplified interface to complex banking operations.

Problem : How do you provide a simple, unified interface to a complex subsystem of banking operations without exposing implementation details?

Forces : There must be a way to hide the complexities of the banking system while providing users with easy-to-use methods for common banking operations.

Solution : Create a Bank_Account facade that provides simplified methods like Deposit() and visa_sms() that internally handle the complex interactions with various banking subsystems.

References : The Facade pattern is one of the structural patterns introduced by the "Gang of Four" in their influential work on design patterns.



Name : Strategy Design Pattern.

Context :

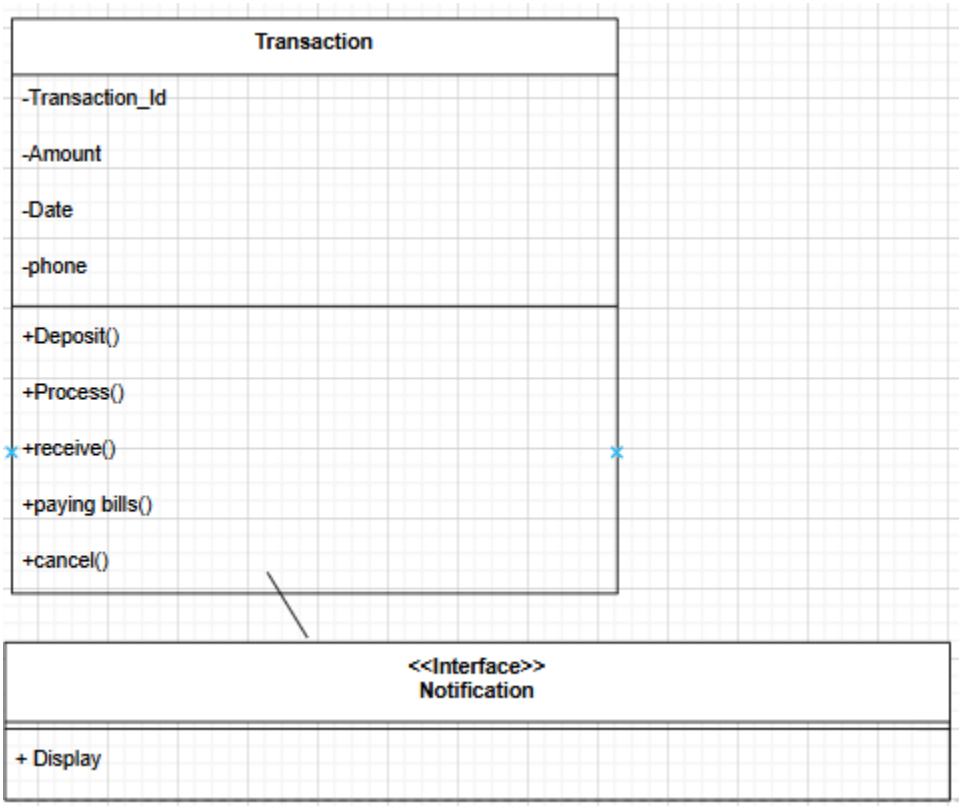
-You need to process different types of transactions. -Each transaction type may require different processing algorithms.

Problem : How can you encapsulate various transaction processing behaviors and make them interchangeable without modifying the Transaction class?

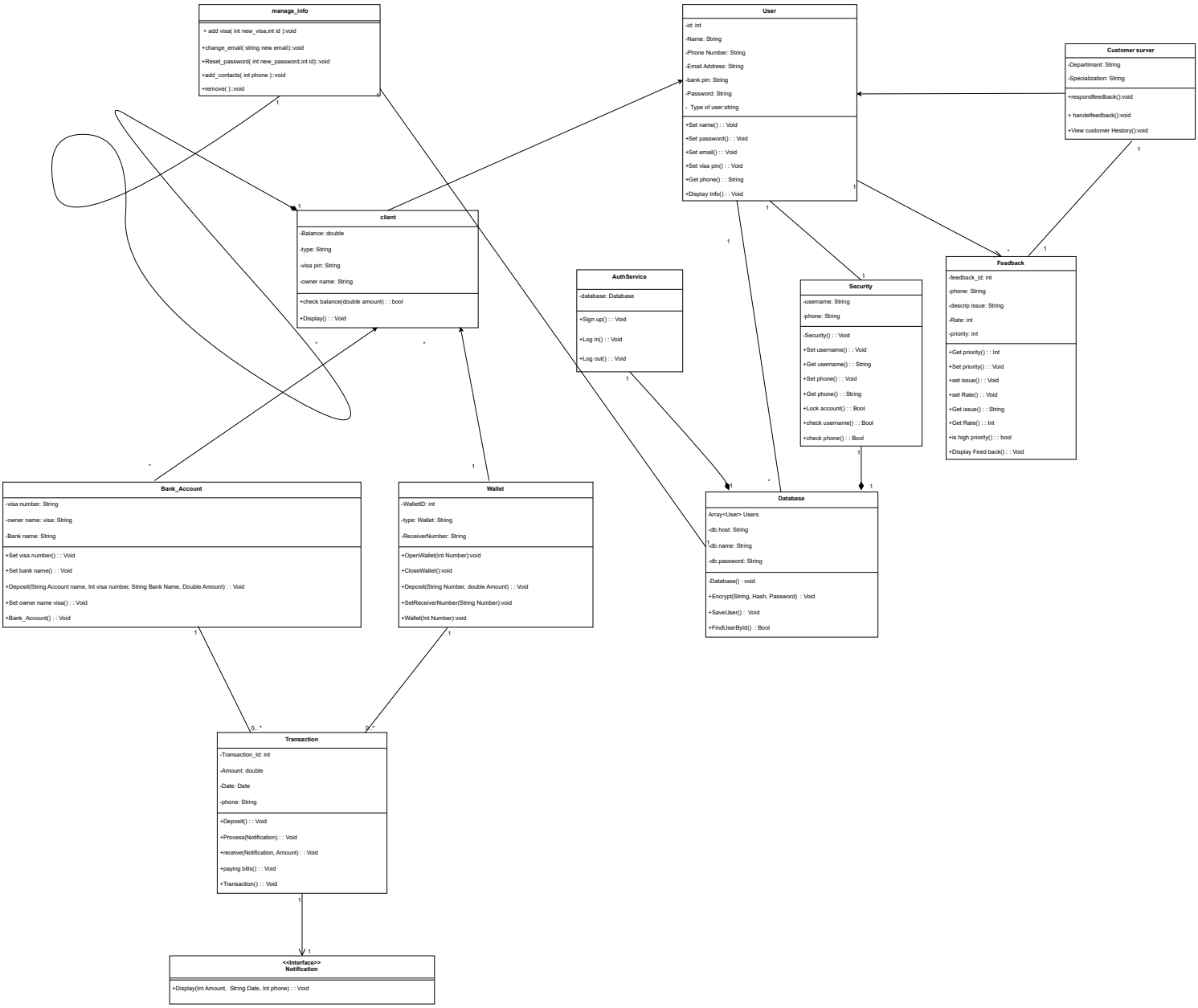
Forces : You want to switch between different transaction processing strategies during runtime without complicating the main Transaction class.

Solution : Define a family of notification algorithms, encapsulate each in the Notification interface, and make Transaction use this interface. This allows the Transaction class to vary its notification behavior independently from clients that use it.

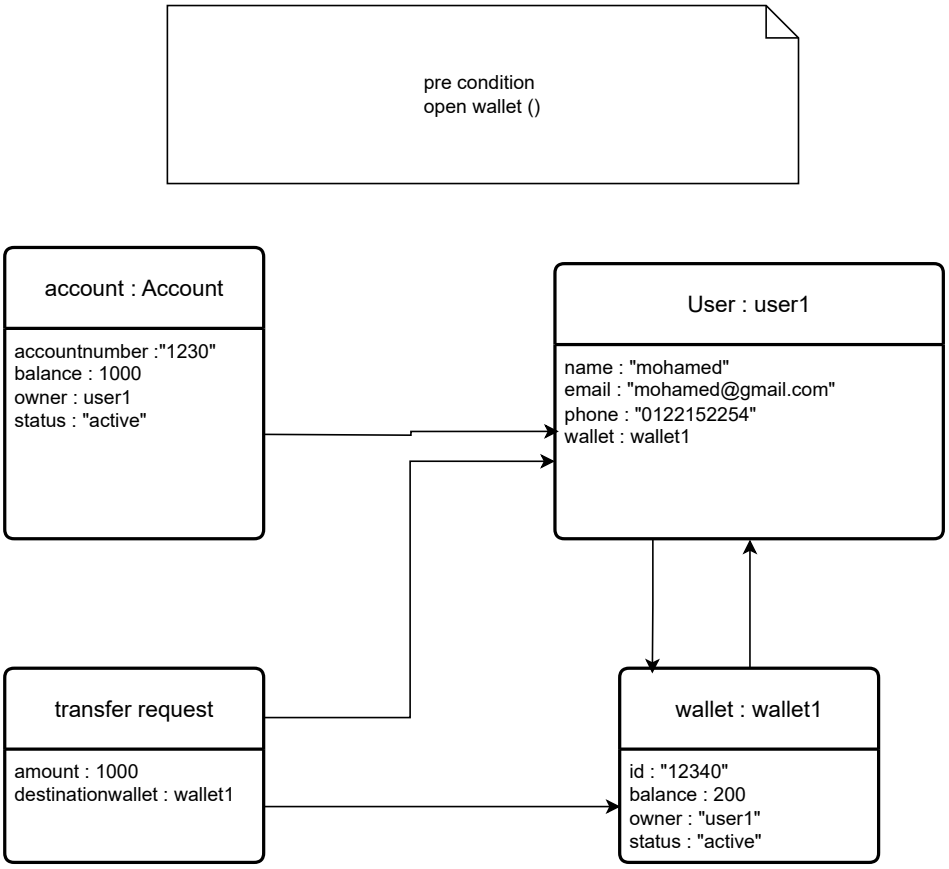
References : The Strategy pattern is one of the behavioral patterns described in the book "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (Gang of Four), published in 1994

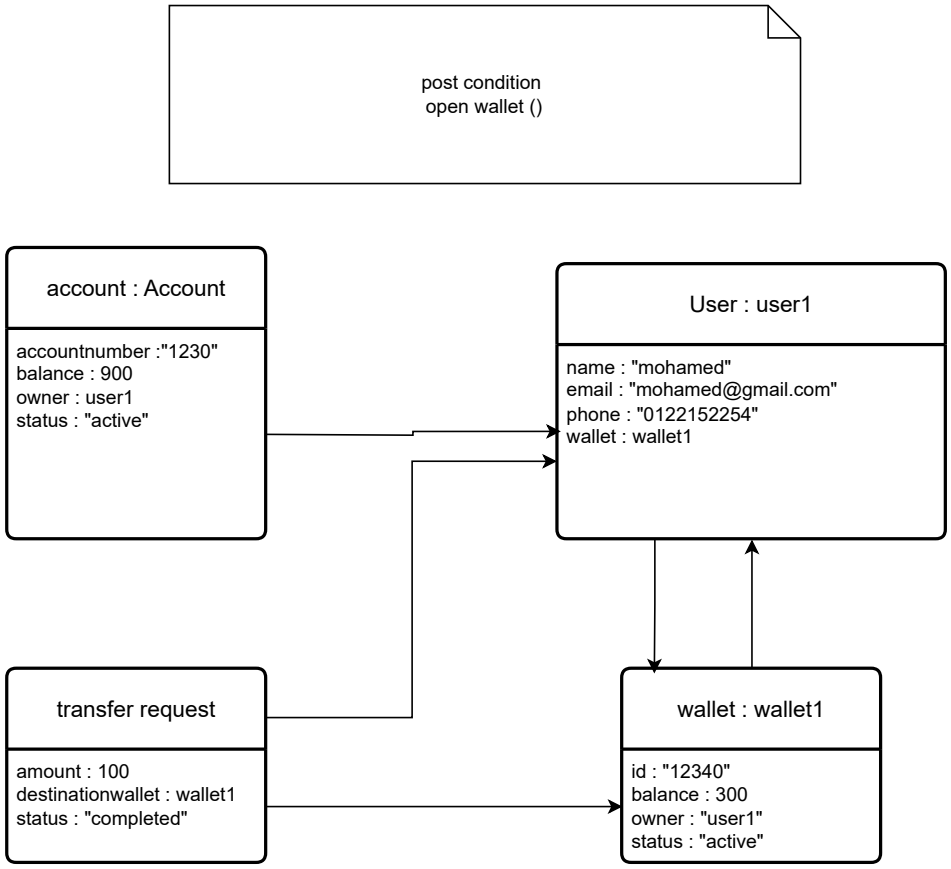


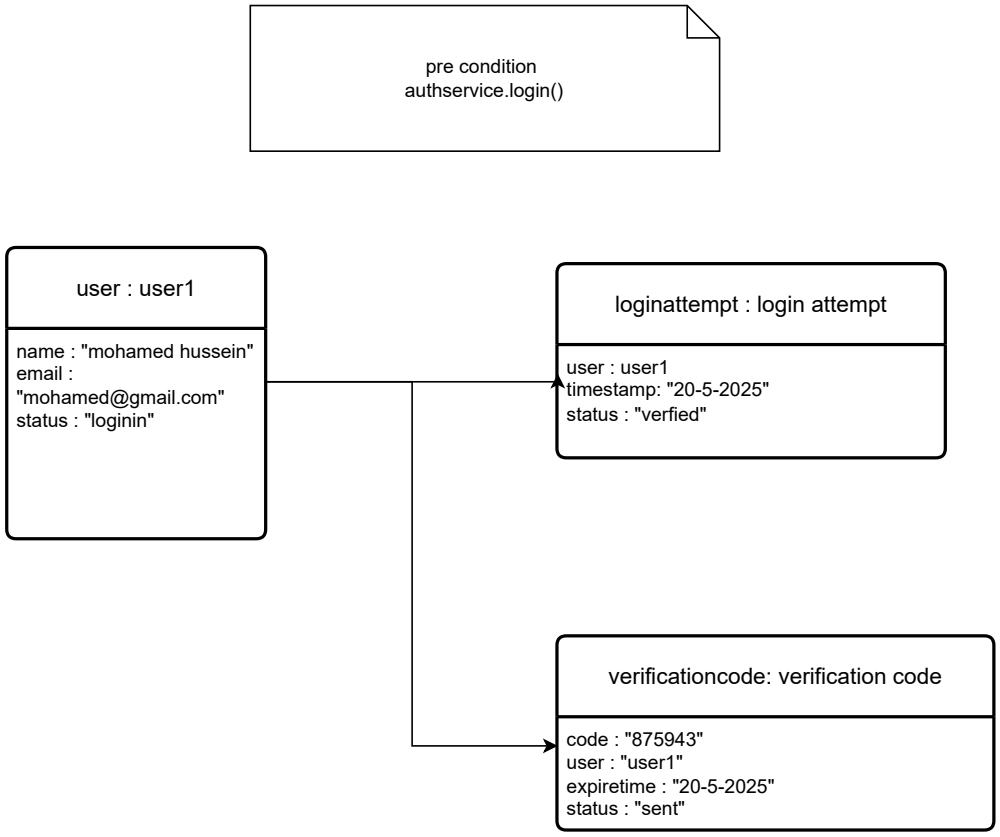
Class Diagram 3 :

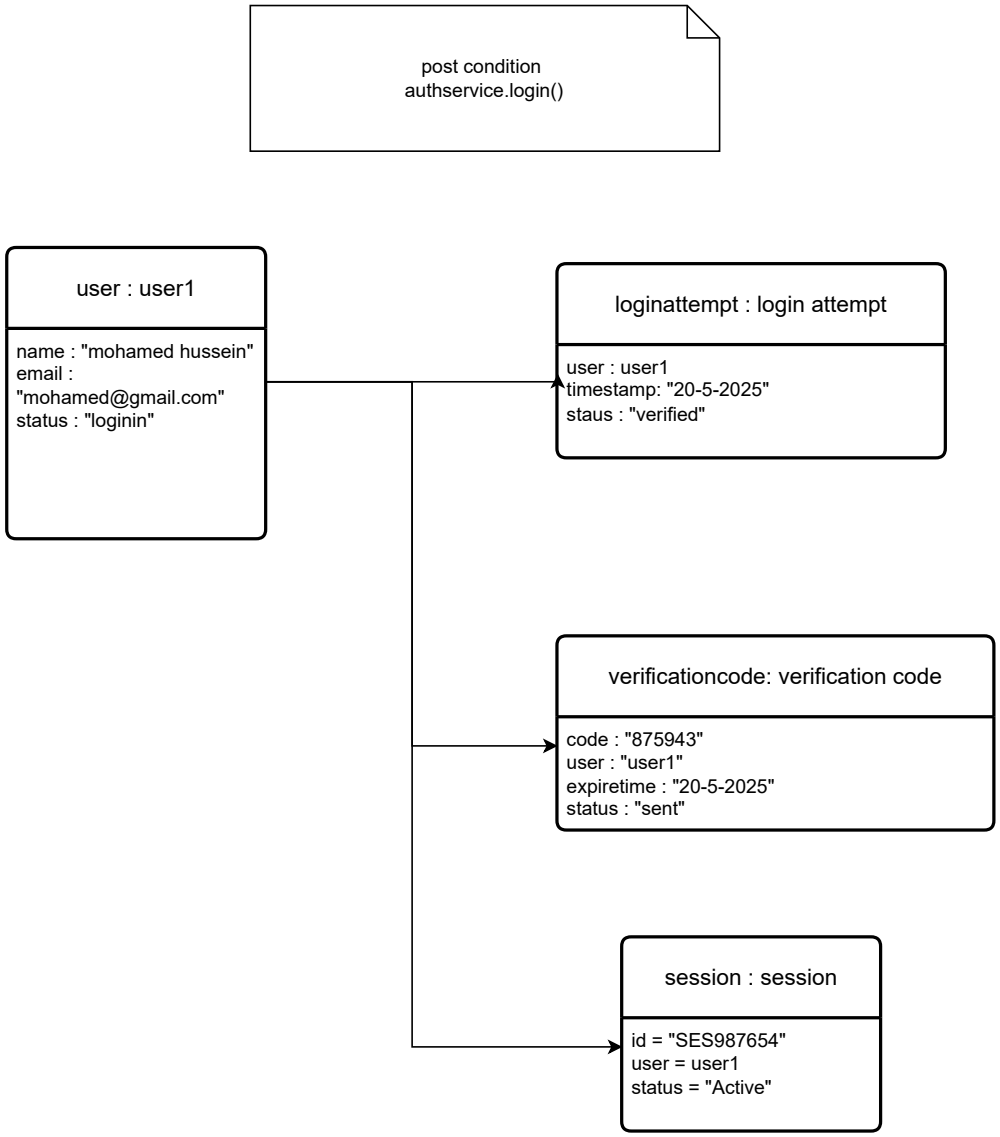

manage_info
+ add visa(int new_visa,int id):void
+change_email(string new email):void
+Reset_password(int new_password,int id):void
+add_contacts(int phone):void
+remove( ):void
User
-id: int
-Name: String
-Phone Number: String
-Email Address: String
-bank pin: String
-Password: String
- Type of user:string
+Set name() : : Void
+Set password() : : Void
+Set email() : : Void
+Set visa pin() : : Void
+Get phone() : : String
+Display Info() : : Void
Customer surver
-Departimant: String
-Specialization: String
+respondfeedback():void
+ handelfeedback():void
+View customer Hestory():void
client
-Balance: double
-type: String
-visa pin: String
-owner name: String
+check balance(double amount) : : bool
+Display() : : Void
AuthService
-database: Database
+Sign up() : : Void
+Log in() : : Void
+Log out() : : Void
Security
-username: String
-phone: String
-Security() : : Void
+Set username() : : Void
+Get username() : : String
+Set phone() : : Void
+Get phone() : : String
+Lock account() : : Bool
+check username() : : Bool
+check phone() : : Bool
Feedback
-feedback_id: int
-phone: String
-descrip issue: String
-Rate: int
-priority: int
+Get priority() : : Int
+Set priority() : : Void
+set issue() : : Void
+set Rate() : : Void
+Get issue() : : String
+Get Rate() : : Int
+Is high priority() : : bool
+Display Feed back() : : Void
Bank_Account
-visa number: String
-owner name: visa: String
-Bank name: String
+Set visa number() : : Void
+Set bank name() : : Void
+Deposit(String Account name, Int visa number, String Bank Name, Double Amount) : : Void
+Set owner name visa() : : Void
+Bank_Account() : : Void
Wallet
-WalletID: int
-type: Wallet: String
-ReceiverNumber: String
+OpenWallet(Int Number):void
+CloseWallet():void
+Deposit(String Number, double Amount) : : Void
+SetReceiverNumber(String Number):void
+Wallet(Int Number):void
Database
Array<User> Users
-db.host: String
-db.name: String
-db.password: String
-Database() : void
+Encrypt(String, Hash, Password) : Void
+SaveUser() : Void
+FindUserById() : Bool
Transaction
-Transaction_Id: int
-Amount: double
-Date: Date
-phone: String
+Deposit() : : Void
+Process(Notification) : : Void
+receive(Notification, Amount) : : Void
+paying bills() : : Void
+Transaction() : : Void
<<Interface>>
Notification
+Display(Int Amount, String Date, Int phone) : : Void

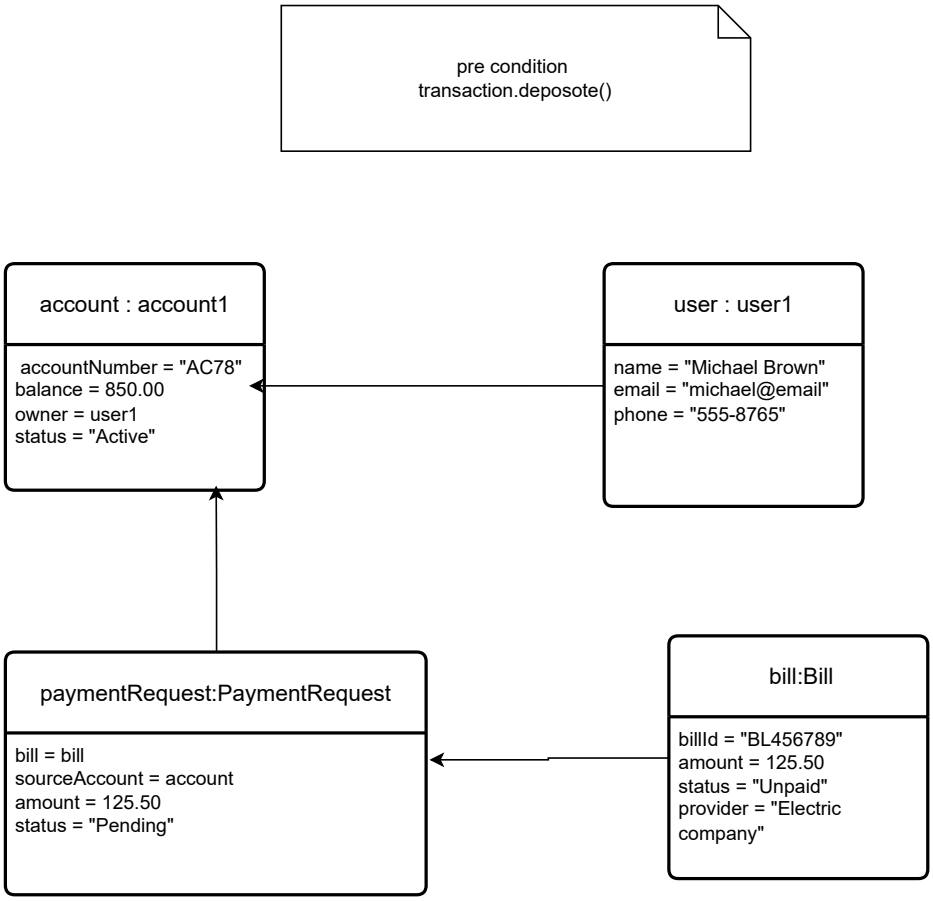
Object Diagrams :

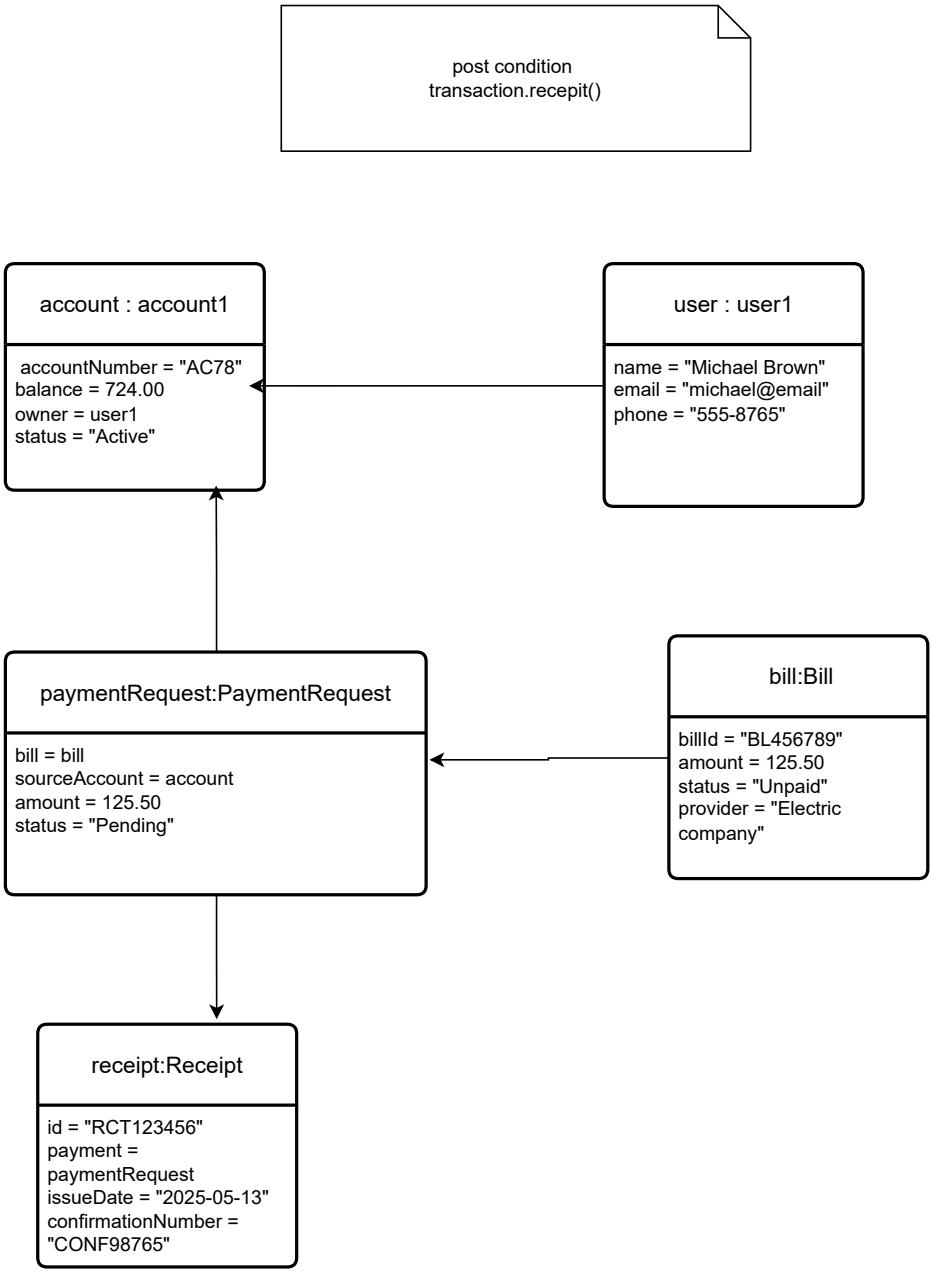


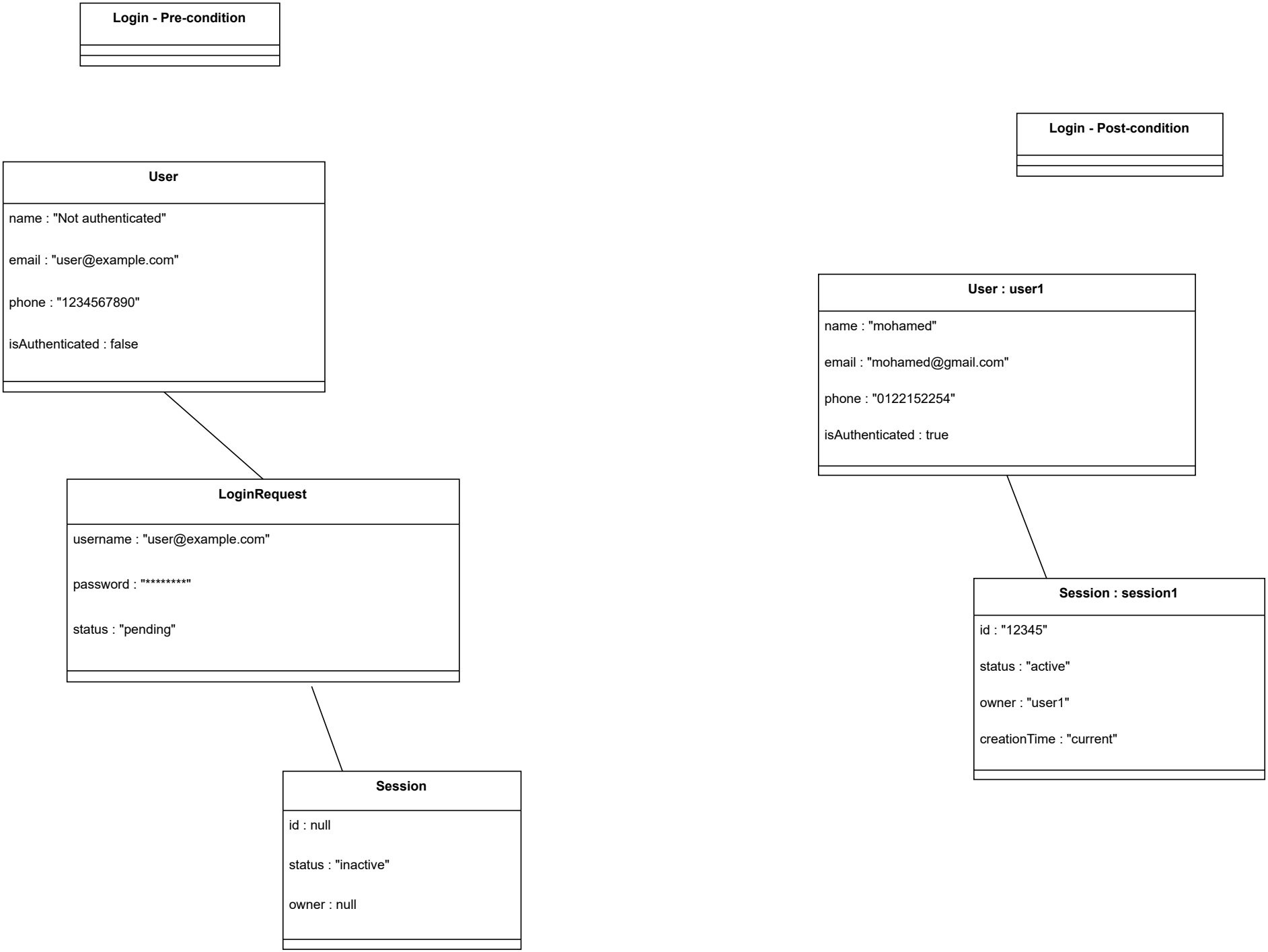

post condition
open wallet ()
account : Account
accountnumber :"1230"
balance : 900
owner : user1
status : "active"
User : user1
name : "mohamed"
email : "mohamed@gmail.com"
phone : "0122152254"
wallet : wallet1
transfer request
amount : 100
destinationwallet : wallet1
status : "completed"
wallet : wallet1
id : "12340"
balance : 300
owner : "user1"
status : "active"

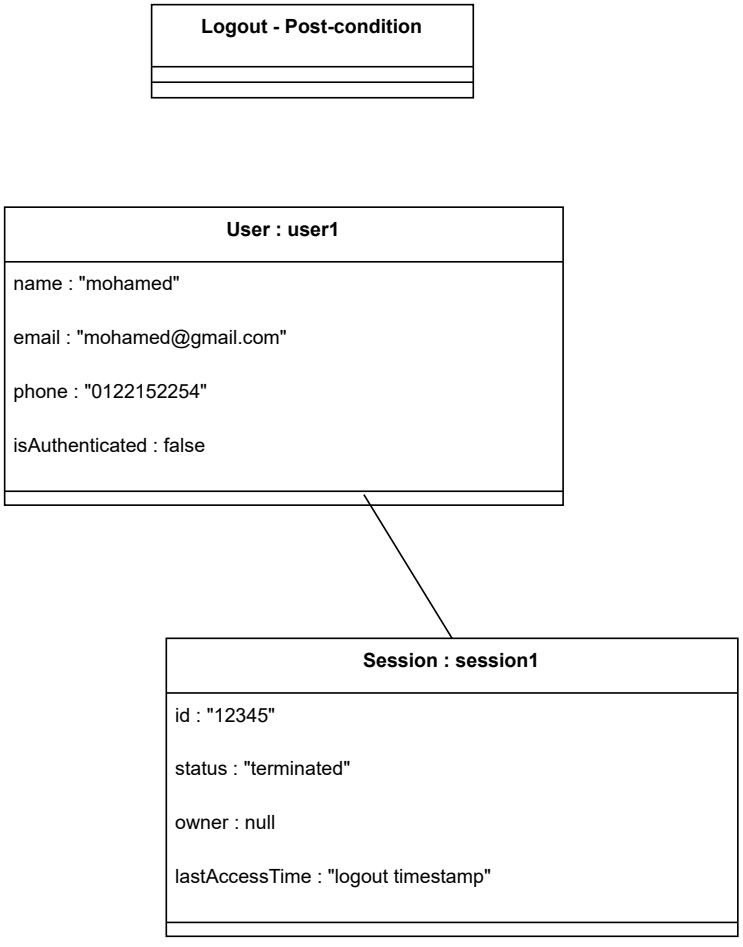

pre condition
authservice.login()
user : user1
name : "mohamed hussein"
email :
"mohamed@gmail.com"
status : "loginin"
loginattempt : login attempt
user : user1
timestamp: "20-5-2025"
status : "verfied"
verificationcode: verification code
code : "875943"
user : "user1"
expiretime : "20-5-2025"
status : "sent"

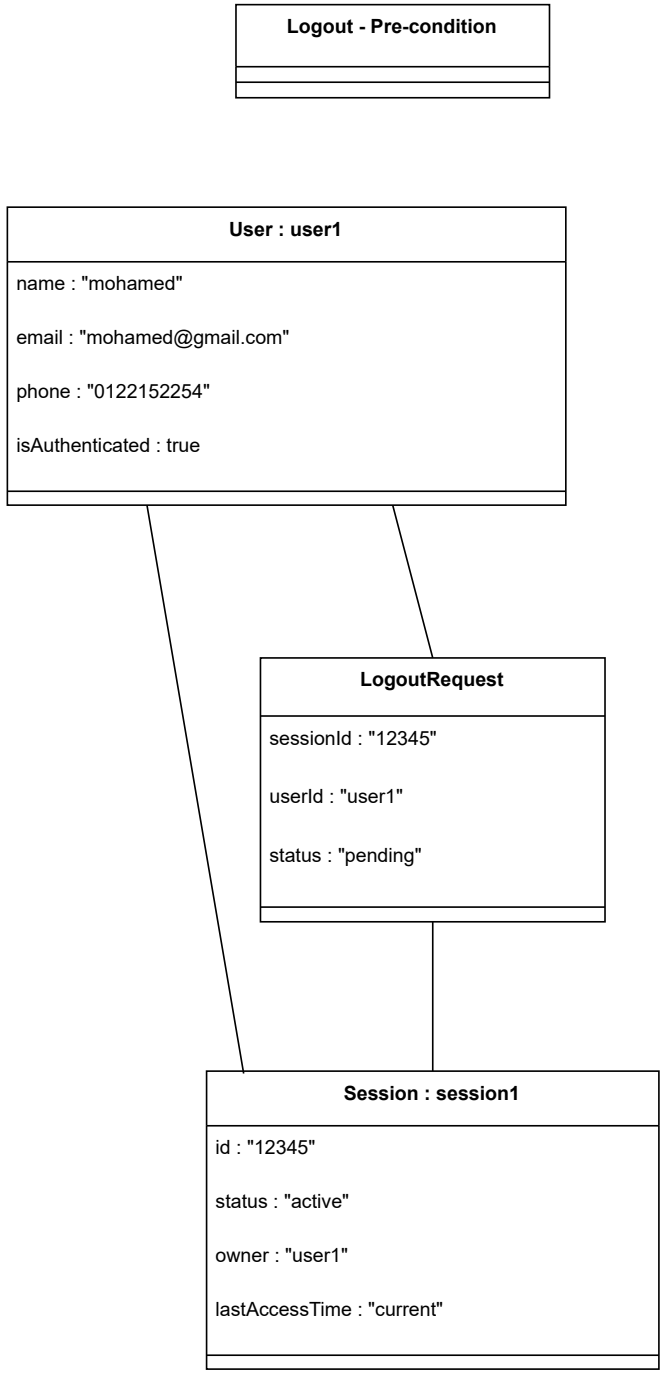

post condition
authservice.login()
user : user1
name : "mohamed hussein"
email :
"mohamed@gmail.com"
status : "loginin"
loginattempt : login attempt
user : user1
timestamp: "20-5-2025"
staus : "verified"
verificationcode: verification code
code : "875943"
user : "user1"
expiretime : "20-5-2025"
status : "sent"
session : session
id = "SES987654"
user = user1
status = "Active"

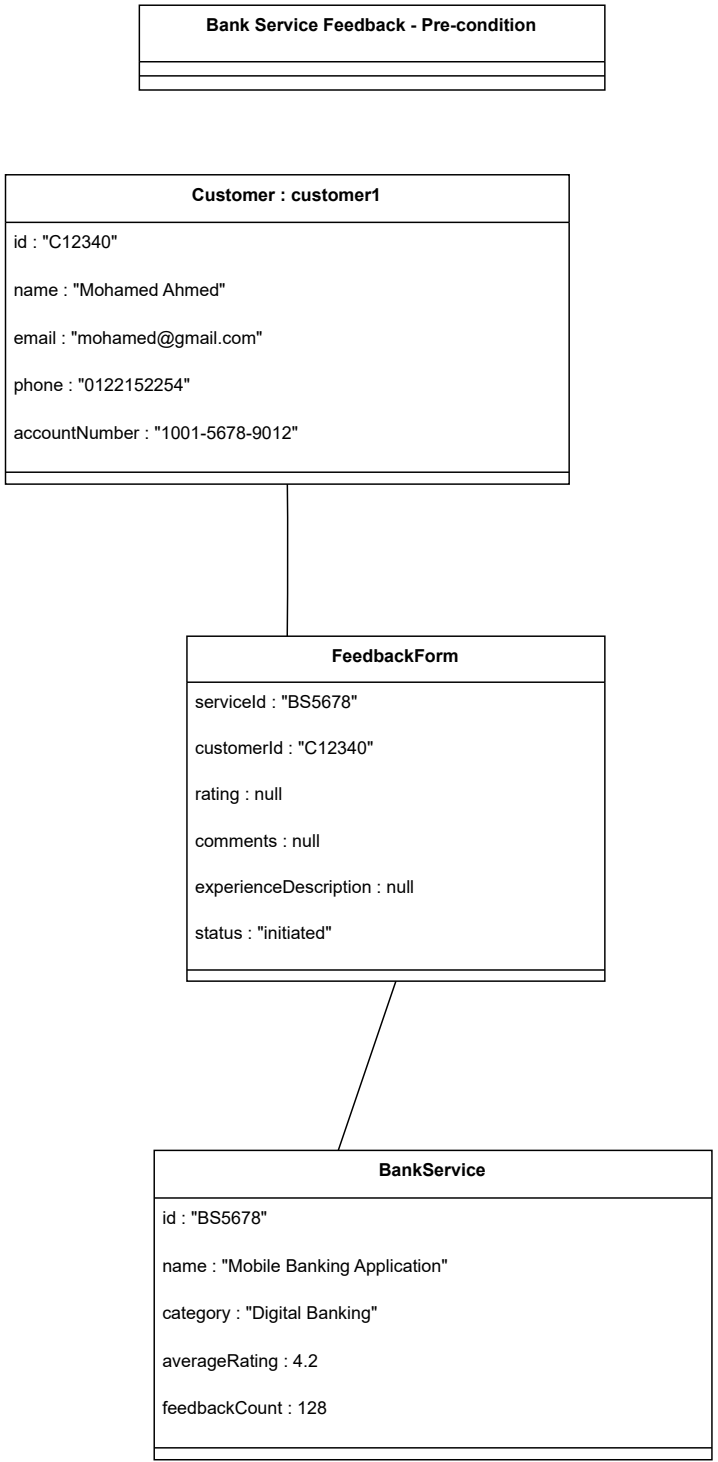

pre condition
transaction.deposote()
account : account1
accountNumber = "AC78"
balance = 850.00
owner = user1
status = "Active"
user : user1
name = "Michael Brown"
email = "michael@email"
phone = "555-8765"
paymentRequest:PaymentRequest
bill = bill
sourceAccount = account
amount = 125.50
status = "Pending"
bill:Bill
billId = "BL456789"
amount = 125.50
status = "Unpaid"
provider = "Electric company"

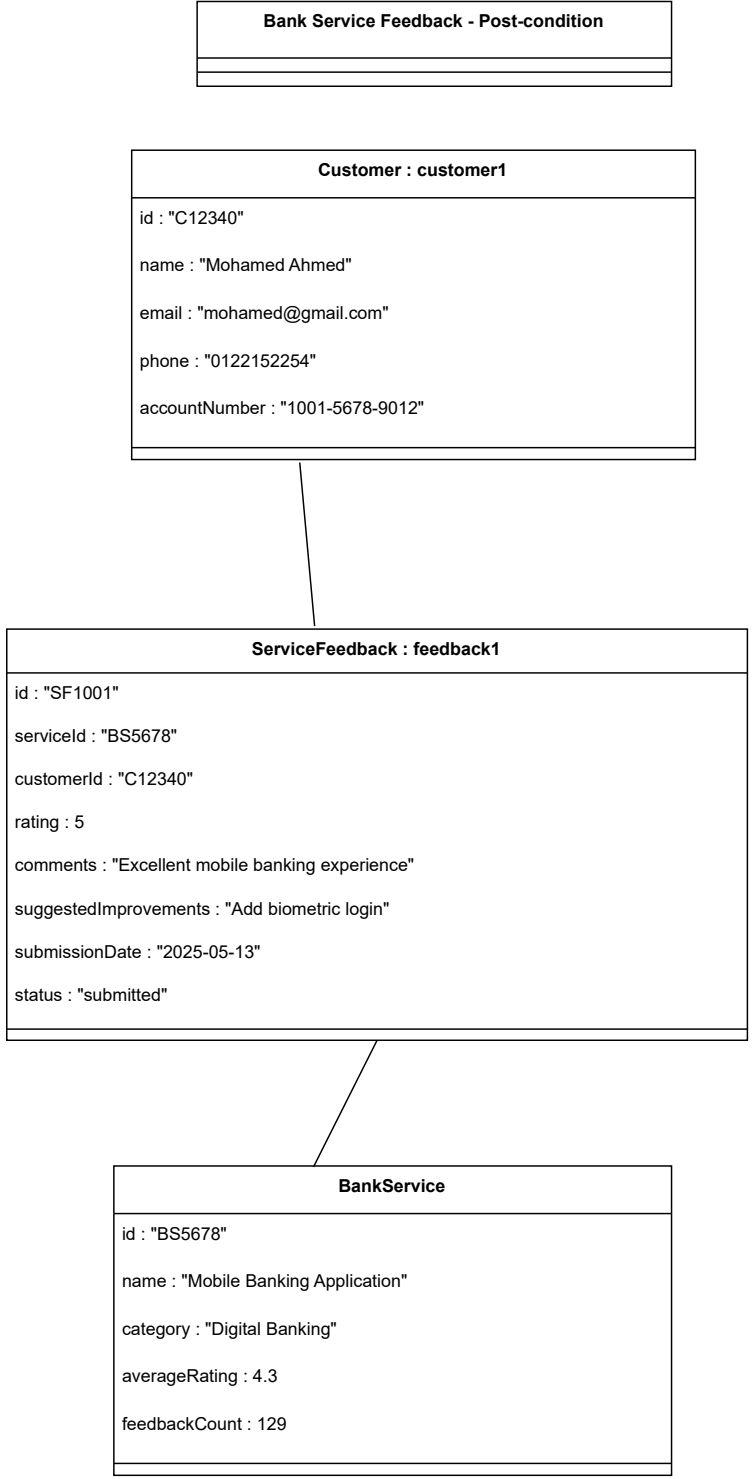

post condition
transaction.recepit()
account : account1
accountNumber = "AC78"
balance = 724.00
owner = user1
status = "Active"
user : user1
name = "Michael Brown"
email = "michael@email"
phone = "555-8765"
paymentRequest:PaymentRequest
bill = bill
sourceAccount = account
amount = 125.50
status = "Pending"
bill:Bill
billId = "BL456789"
amount = 125.50
status = "Unpaid"
provider = "Electric company"
receipt:Receipt
id = "RCT123456"
payment = paymentRequest
issueDate = "2025-05-13"
confirmationNumber = "CONF98765"


Login - Pre-condition
User
name : "Not authenticated"
email : "user@example.com"
phone : "1234567890"
isAuthenticated : false
LoginRequest
username : "user@example.com"
password : "********"
status : "pending"
Session
id : null
status : "inactive"
owner : null
Login - Post-condition
User : user1
name : "mohamed"
email : "mohamed@gmail.com"
phone : "0122152254"
isAuthenticated : true
Session : session1
id : "12345"
status : "active"
owner : "user1"
creationTime : "current"

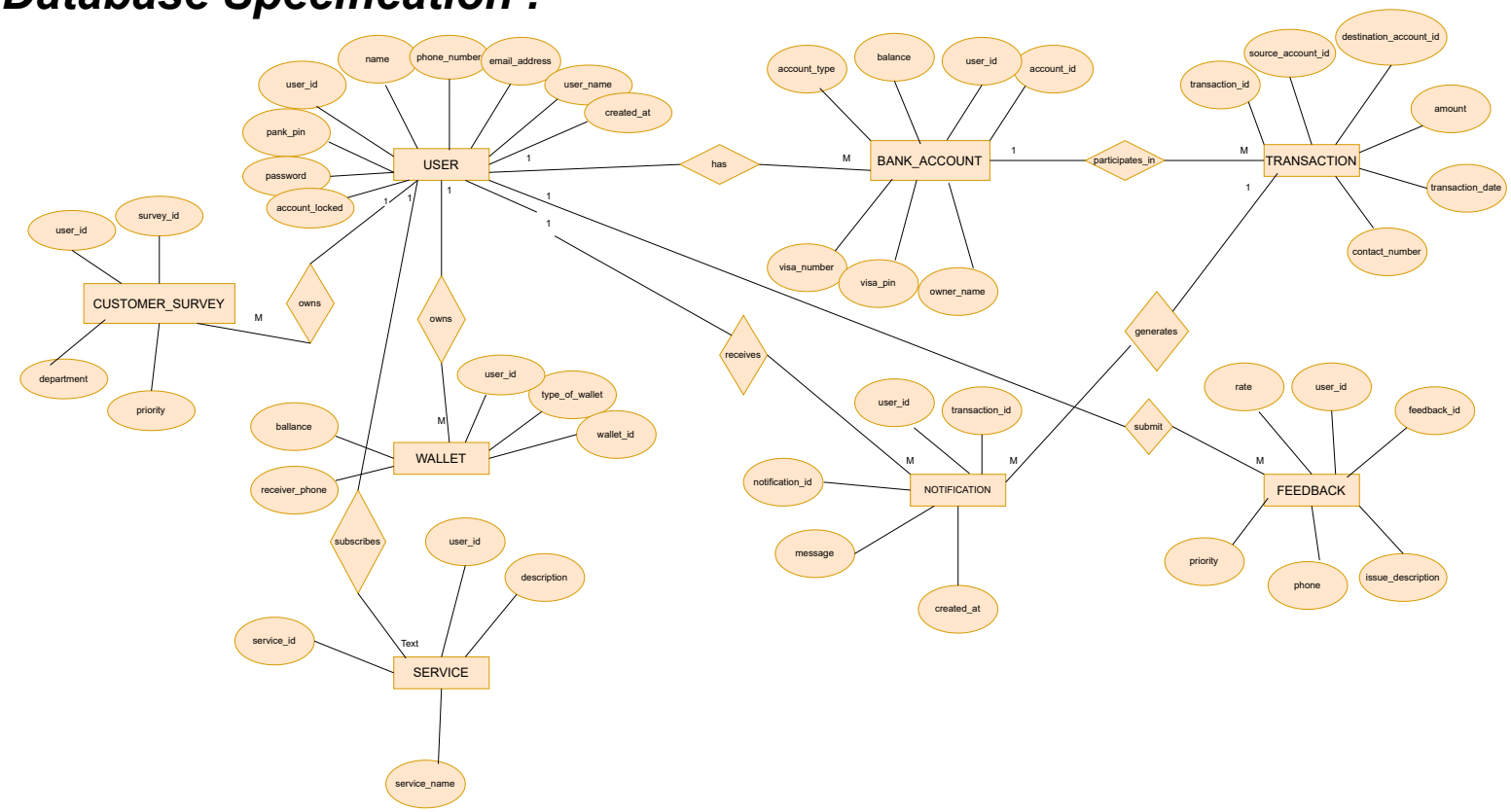

Logout - Post-condition
User : user1
name : "mohamed"
email : "mohamed@gmail.com"
phone : "0122152254"
isAuthenticated : false
Session : session1
id : "12345"
status : "terminated"
owner : null
lastAccessTime : "logout timestamp"


Logout - Pre-condition
User : user1
name : "mohamed"
email : "mohamed@gmail.com"
phone : "0122152254"
isAuthenticated : true
LogoutRequest
sessionId : "12345"
userId : "user1"
status : "pending"
Session : session1
id : "12345"
status : "active"
owner : "user1"
lastAccessTime : "current"


Bank Service Feedback - Pre-condition
Customer : customer1
id : "C12340"
name : "Mohamed Ahmed"
email : "mohamed@gmail.com"
phone : "0122152254"
accountNumber : "1001-5678-9012"
FeedbackForm
serviceId : "BS5678"
customerId : "C12340"
rating : null
comments : null
experienceDescription : null
status : "initiated"
BankService
id : "BS5678"
name : "Mobile Banking Application"
category : "Digital Banking"
averageRating : 4.2
feedbackCount : 128

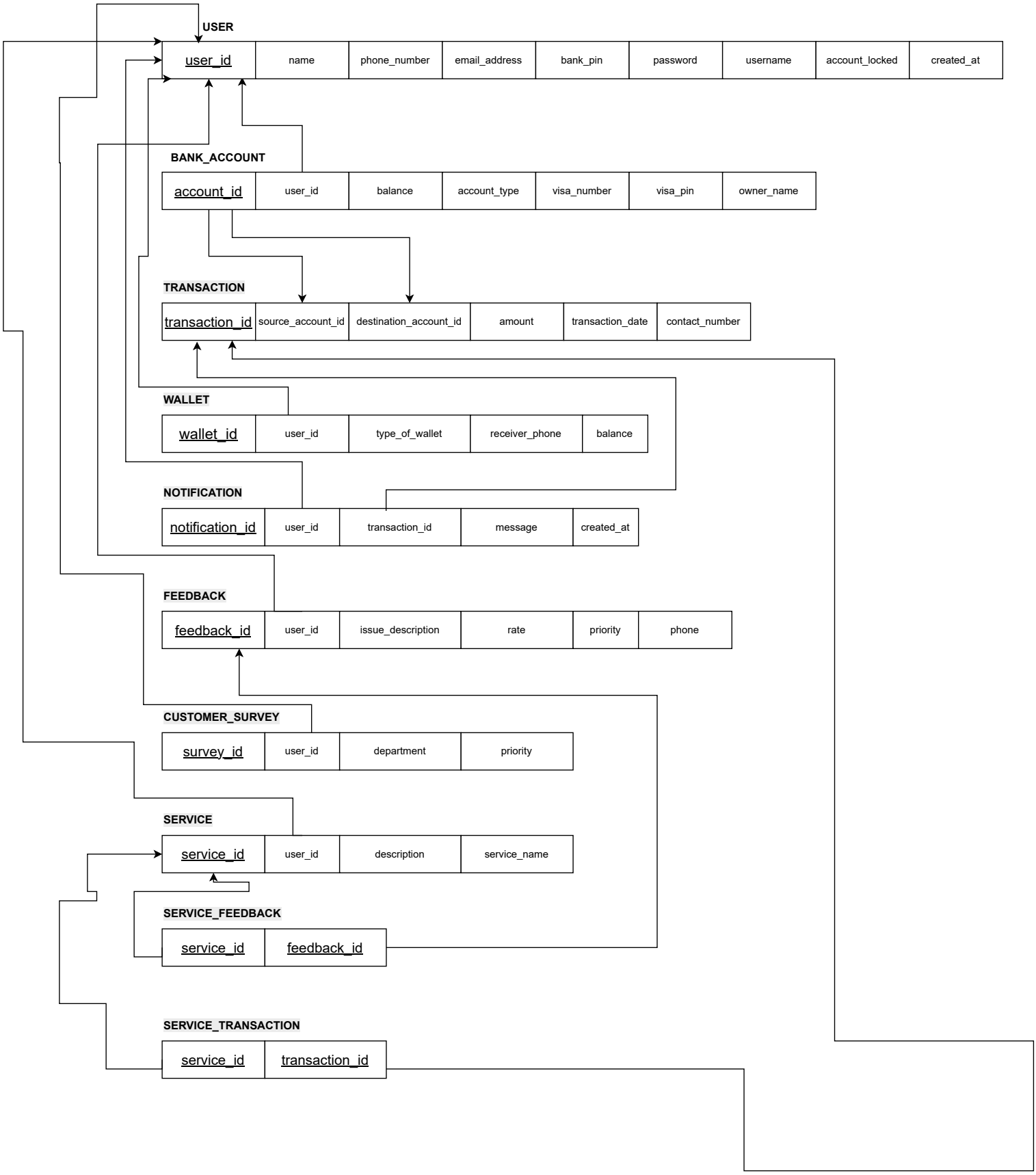

Bank Service Feedback - Post-condition
Customer : customer1
id : "C12340"
name : "Mohamed Ahmed"
email : "mohamed@gmail.com"
phone : "0122152254"
accountNumber : "1001-5678-9012"
ServiceFeedback : feedback1
id : "SF1001"
serviceId : "BS5678"
customerId : "C12340"
rating : 5
comments : "Excellent mobile banking experience"
suggestedImprovements : "Add biometric login"
submissionDate : "2025-05-13"
status : "submitted"
BankService
id : "BS5678"
name : "Mobile Banking Application"
category : "Digital Banking"
averageRating : 4.3
feedbackCount : 129

Database Specification :



ERD diagram :

Tables :


USER
user_id
name
phone_number
email_address
bank_pin
password
username
account_locked
created_at
BANK_ACCOUNT
account_id
user_id
balance
account_type
visa_number
visa_pin
owner_name
TRANSACTION
transaction_id
source_account_id
destination_account_id
amount
transaction_date
contact_number
WALLET
wallet_id
user_id
type_of_wallet
receiver_phone
balance
NOTIFICATION
notification_id
user_id
transaction_id
message
created_at
FEEDBACK
feedback_id
user_id
issue_description
rate
priority
phone
CUSTOMER_SURVEY
survey_id
user_id
department
priority
SERVICE
service_id
user_id
description
service_name
SERVICE_FEEDBACK
service_id
feedback_id
SERVICE_TRANSACTION
service_id
transaction_id