THIS FORM IS FOR BOTH THE GENERAL & MEDICAL INFORMATICS PROGRAMMES

SE-I COURSE PROJECT (COVER SHEET)

Discussions are Scheduled for Week 12/13 *(Details TBA Later)*.

- Print 1 copy of this cover sheet and attach it to a printed copy of the documentation *(SRS, … etc.)*. You must also submit soft copies of all your documents *(as PDFs)*; details will be announced later.
- Please write all your names in Arabic.
- Please make sure that your students' IDs are correct.
- Handwritten Signatures for the attendance of all team members should be filled in before the discussion.
- Please attend the discussion on time *(announced separately)*; late teams will lose 5 grades.

Project Name: _____

Team Information ***(typed, not handwritten, except for the attendance signature)*:**

	ID [Ordered by ID]	Full Name [In Arabic]	Attendance [Handwritten Signature]	Final Grade
1	20230066	ادهم محمد صبري زين الدين		
2	20230112	أيمن عبدالعزيز إبراهيم الفقي		
3	20230508	محمد منصور السيد محمد		
4	20230106	إياد جمال عبداللطيف حسن		
5	20230493	محمد علي محمد عبد العليم		
6	20230033	احمد مجدي حسني السيد		

Grading Criteria:

14 Items		Grade	Notes
1. Functional Requirements	**1**		
2. Non-Functional Requirements	**1**		
3. Use-Case Diagram(s) *including general use-cases for the system and the detailed use-cases description*	**2**		
4. System Architecture – *including applied Architectural Pattern(s)*	**1**		
5. Activity Diagrams	**2**		
6. Object Diagrams *(Including object diagrams that illustrate the preconditions and the post-conditions of selected functions)*	**2**		
7. Package Diagram(s)	**1**		
8. Sequence Diagrams *including* ***System Sequence Diagrams (SSDs)***	**2**		
9. Database Specification *(ERD, Tables)*	**2**		
10. Collaboration/Communication Diagrams	**2**		
11. Class Diagram *(3 versions)* **1) An initial version based on the requirements and Use-Case/Activity diagrams.** **2) An intermediate version based on the interaction diagrams.** **3) A final version after applying the design patterns and other modifications.**	**5**		
12. Three Mandatory Design Patterns Applied *(Including a typed description)*	**3**		
13. Front End Design for all Functions *(HTML, Bootstrap)*	**1**		

14. Implementation based on the submitted Requirements & Design. Should include at least 4 of the following modules (in addition, of course, to modules specific to your projects): **1) User Role Management Module.** **2) User manipulation Module** *(Login, Add / Delete / Update / Search, List).* **3) Controlling Resources Module** *(Rooms, Orders, Products, … etc.).* **4) Reservation and Rescheduling Module.** **5) Generating Reports Module** *(PDFs, … etc.).* **6) Sending Emails or Notifications Module.**	**5**		

Teaching-Assistant's Signature: _____

_____
30

Helwan University – Faculty of Computing & Artificial Intelligence

Module: CS251 Software Engineering 1 – Spring "Semester 2" 2024-2025

Falsopay - Modern Banking Platform

Software Engineering Documentation

By: Adham Zineldin, Mohamed Ali, Eyad Gamal, Ayman Abdelaziz, Ahmed Magdy, Mohamed Mansour

5/10/2025

Table of Contents

PART 1: Overview & Software Requirements Specification

1. Introduction

a) Purpose

Falsopay is a modern banking platform designed to provide secure, efficient, and user-friendly financial services. The system aims to revolutionize digital banking by offering instant payments, real-time transaction updates, and comprehensive bank account management.

b) Project Scope

The project encompasses:

Feature	Description
User Authentication	Secure login, registration, and password management
Account Management	Create, view, and manage bank accounts
Card Management	Virtual and physical card management
Payment Processing	Instant money transfers and bill payments
Transaction History	Detailed transaction records and statements
Support System	Ticket-based customer support

c) Glossary and Abbreviations

Term	Definition

JWT	JSON Web Token - A secure way to transmit information between parties
API	Application Programming Interface - A set of rules for building and interacting with software applications
REST	Representational State Transfer - An architectural style for distributed hypermedia systems
UI/UX	User Interface/User Experience - The design of user interactions and experiences

d) List of the System Stakeholders

Stakeholder Type	Description
End Users	Bank customers, system administrators, and support staff
External Stakeholders	Banks, payment processors, and regulatory bodies
Development Team	Software developers, QA engineers, and DevOps engineers

e) References

Reference	URL
PHP Documentation	https://www.php.net/docs.php
React Documentation	https://reactjs.org/docs
MySQL Documentation	https://dev.mysql.com/doc/
JWT Documentation	https://jwt.io/introduction

2. Functional Requirements

a) User Requirements Specification

Authentication & Authorization:

1. Users must be able to register with email and password
2. Users must be able to login using JWT authentication
3. Users must be able to reset their password
4. Users must be able to manage their profile

b) System Requirements Specification

Backend Requirements:

1. RESTful API implementation
2. WebSocket server for real-time updates
3. Database management system
4. Security implementation
5. Error handling and logging

c) Requirements' Priorities

Using the MoSCoW Scheme:

Total: 30 functions | Implementation: 24 functions

#	Function Description	Priority	Implemented
1	Register and log in via email	Must have	☑
2	Log in via Google	Should have	☑
3	Send money to a user by phone number	Must have	☑
4	View transaction history	Must have	☑
5	Check account balance	Must have	☑
6	Link a debit card securely	Must have	☑
7	Link a prepaid card	Must have	☑
8	Receive success/failure confirmation for transfers	Must have	☑
9	Edit profile and update phone number	Should have	☑
10	Reset password via email	Should have	☑
11	Block/unblock a linked card	Should have	☑
12	Report suspicious activity	Should have	☑
13	Scan QR code to pay another user	Should have	☑
14	View and download monthly account statements	Should have	☑
15	Add money to prepaid card from bank	Should have	☑

#	Function Description	Priority	Implemented
16	Set transfer limits per day/week	Could have	☑
17	Enable/disable notifications	Could have	☑
18	Access customer support chat	Could have	☑
19	Get in-app help/tutorials	Could have	☑
20	View app usage statistics (e.g. total amount sent)	Could have	☑
21	Add recipients to favorites for quick transfer	Should have	☑
22	Set a PIN for confirming transfers	Must have	☑
23	Schedule recurring transfers	Could have	✗
24	Export transactions as CSV/PDF	Could have	☑
25	Use biometric authentication (fingerprint/face)	Should have	✗
26	Multi-language support (English/Arabic toggle)	Won't have now	✗
27	Push notifications for incoming transfers	Should have	☑
28	Search/filter transactions by date or amount	Should have	☑
29	Request money from other users	Should have	☑
30	Create payment links to share externally	Could have	☑

Total: 6 functions | Implementation: 6 functions

#	Function Description	Priority	Implemented
31	Process transfer requests in < 1 second	Must have	☑
32	Store passwords securely using hashing (e.g., bcrypt)	Must have	☑
33	Maintain a secure audit trail of all actions	Should have	☑
34	Auto-backup all data every 24 hours	Could have	☑
35	Support high concurrency without data loss	Must have	☑
36	Encrypt all sensitive data at rest and in transit	Must have	☑

3. Non-functional Requirements

a) Categories of Non-Functional Requirements

1. Performance
2. Security
3. Reliability
4. Usability
5. Maintainability
6. Scalability

b) Non-functional Requirements Specification

Performance

- **Web pages must load within 5 seconds under normal network conditions.**
- **All API endpoints must respond in under 1 seconds.**

- **The system must efficiently handle 1,000+ concurrent users without degradation in performance.**
- **Real-time data updates (e.g., via WebSockets or polling) must propagate within 100 milliseconds.**

Security

- **All authentication must use JWT (JSON Web Tokens) for secure, stateless sessions.**
- **All communications must be secured using HTTPS (TLS 1.2 or higher).**
- **Strict input validation must be enforced across all endpoints.**
- **The system must be resilient against common web threats including:**
  - **SQL Injection through parameterized queries and ORM usage.**
  - **Cross-Site Scripting (XSS) via output encoding and CSP headers.**
  - **Cross-Site Request Forgery (CSRF) through tokens and origin check**

c) Fit Criteria

1. **Performance Testing**
   - **Open the website in a browser and use Chrome DevTools (F12 → Network tab) to check that:**
     - **Pages load in under 2 seconds**
     - **API responses are under 500ms**
2. **Security Testing**
   - **Use simple online tools like:**
     - **SecurityHeaders.com to check HTTPS and header security**
     - **XSS Game (for learning and checking basic XSS issues)**
   - **Manually test for input validation by entering special characters (like <script>, ' OR 1=1, etc.) into forms.**
3. **Load Testing**
   - **Test your app with multiple tabs or devices at the same time (try with friends/classmates).**
   - **Ensure it works correctly when many users are using it at once (aim for at least 10–20 as a small-scale test).**
4. **Accessibility Testing**
   - **Use the built-in Lighthouse tool in Chrome (Right-click → "Inspect" → "Lighthouse" tab → Run Accessibility report).**

- **Make sure important buttons and inputs are clearly labeled and can be used with the keyboard.**

d) Architecture Impact

- Microservices architecture for scalability
- Caching layer for performance
- Load balancing for reliability

4. Design & Implementation Constraints

Technical Constraints:

1. PHP 8.2+ requirement
2. MySQL 8.0+ database
3. Node.js 18+ for frontend
4. Modern browser support

Business Constraints:

1. Compliance with banking regulations
2. Data privacy requirements
3. Security standards
4. Performance requirements

5. System Evolution

a) Anticipated Changes:

1. Mobile app development
2. Additional payment methods
3. Enhanced security features
4. Integration with more banks

b) Future Impact:

1. Scalable architecture design
2. Modular code structure
3. API versioning
4. Database migration support

6. Requirements Discovery & Validation

Discovery Approaches:

1. User interviews
2. Market research
3. Competitor analysis
4. Prototype testing

Validation Techniques:

1. User acceptance testing
2. Security testing
3. Performance testing
4. Usability testing

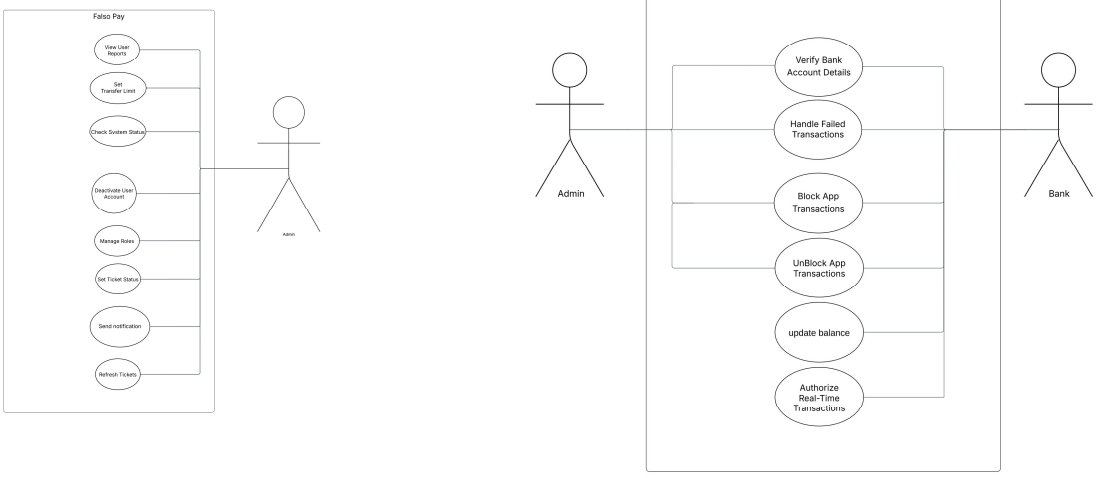
PART 2: System Design & Models

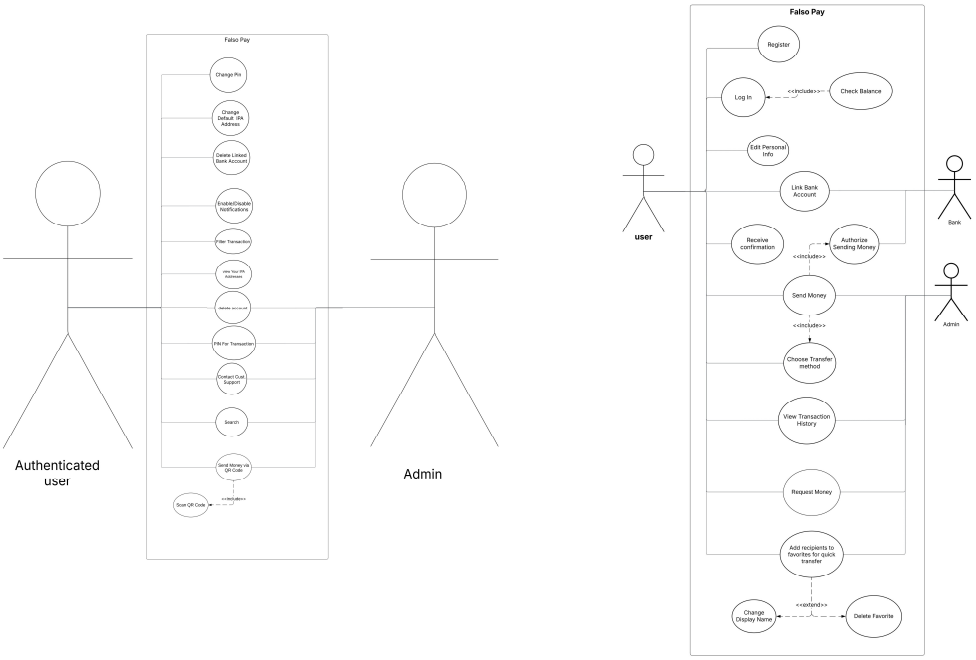
8. Functional Diagrams (ALL DIAGRAMS ARE AT THE END)

a) Use-Case Diagrams

The following use-case diagram illustrates the main interactions between users and the system:

Use Case Diagram:





b) System Architecture

1. **Modified MVC (Model-View-Controller) Pattern**
   - **Why Used?**
   - **Separates concerns between data, presentation, and business logic**
   - **Makes the codebase more maintainable and testable**
   - **Enables parallel development of frontend and backend**

2. **Repository Pattern**
   - **Why Used?**
   - **Abstracts data access logic from business logic**
   - **Makes the system more testable through mocking**
   - **Provides a consistent interface for data operations**

3. **Service Layer Pattern**

- **Why Used?**
- **Encapsulates complex business logic**
- **Promotes code reuse across controllers**
- **Makes the system more maintainable**

c) Design Patterns

1. **Repository Pattern**
2. **Observer Pattern**
3. **Strategy Pattern**

d) Class specifications

Entity Classes (Models)

1. **User**
2. **BankAccount**
3. **Card**
4. **Transaction**
5. **SupportTicket**
6. **MoneyRequest**
7. **SystemSettings**
8. **InstantPaymentAddress**
9. **Bank**
10. **BankUser**
11. **Favorite**

Logic Classes (Controllers)

1. **AuthController**
2. **UserController**
3. **UserAdminController**
4. **BankAccountController**

5. **CardController**
6. **TransactionController**
7. **SupportController**
8. **MoneyRequestController**
9. **SystemController**
10. **InstantPaymentAddressController**
11. **BankController**
12. **BankUserController**
13. **FavoriteController**

Service Classes

1. **EmailService**
2. **SocketService**
3. **WhatsAppAPI**

e) Design Smell

In the FalsoPay system, a design smell can be identified in the TransactionService class, which exhibits characteristics of a "Bloated Service." This class handles multiple responsibilities related to transactions:

1. **Transaction creation and execution**
2. **Balance verification**
3. **Transaction validation**
4. **Notification triggering**
5. **Account balance updates**
6. **Transaction history management**
7. **Money request processing**

This violates the Single Responsibility Principle (SRP), one of the SOLID principles of object-oriented design. The TransactionService has too many responsibilities and becomes a central point of coupling in the system.

Evidence from the Design

In the sequence diagram for UC9-SendMoney, we can see that the TransactionService is responsible for:

1. **Preparing transactions**

2. **Checking balances (through AccountService)**
3. **Creating transaction drafts**
4. **Setting transfer methods**
5. **Executing transactions**
6. **Updating accounts**
7. **Triggering notifications**

This creates a highly coupled design where changes to any transaction-related functionality might affect the entire service, making it harder to maintain and test.

Impact

This design smell leads to several problems:

1. **Maintainability issues: Changes to one aspect of transaction processing might affect unrelated functionality**
2. **Testing difficulties: The service becomes hard to test due to its many responsibilities**
3. **Decreased cohesion: The class lacks a focused, single purpose**
4. **Increased coupling: Many other components depend on this service**
5. **Scalability challenges: The service becomes a bottleneck in the system**

Suggested Solution

To address this design smell, we should refactor the TransactionService by applying the Single Responsibility Principle and splitting it into more focused services:

1. **TransactionCreationService: Responsible for creating and preparing transactions**
2. **TransactionValidationService: Handles validation of transactions**
3. **TransactionExecutionService: Manages the execution of transactions**
4. **BalanceVerificationService: Focuses on balance checks and verification**
5. **TransactionNotificationService: Handles notification aspects of transactions**
6. **MoneyRequestService: Manages money request functionality**

Additionally, we should:

1. **Implement a Facade pattern if clients need a simplified interface to these services**
2. **Use the Mediator pattern to handle complex interactions between these services**
3. **Implement proper dependency injection to reduce coupling**
4. **Create well-defined interfaces for each service**

This refactoring would lead to a more maintainable, testable, and scalable design that better follows the principles of object-oriented design.

Benefits of the Solution

1. **Improved maintainability: Changes to one aspect of transaction processing won't affect others**
2. **Better testability: Each service can be tested in isolation**
3. **Higher cohesion: Each class has a clear, single responsibility**
4. **Reduced coupling: Dependencies are more explicit and manageable**
5. **Better scalability: Services can be scaled independently based on load**
6. **Easier to extend: New transaction-related functionality can be added without modifying existing code**

PART 3: Development Phase

9. Implementation Modules

Module	Description
User Role Management	Handles user authentication, authorization, and role-based access control
User Manipulation	Manages user profiles, settings, and preferences
Resource Control	Manages system resources and access permissions
Payment Processing	Handles money transfers, bill payments, and transaction processing
Reporting	Generates financial reports and transaction statements
Notifications	Sends email, SMS, and push notifications

PART 4: Complexity & Testing

10. Complexity Metrics

Metric	Value	Description
Lines of Code (LOC)	25,000	Total number of lines in the codebase

Cyclomatic Complexity (CCM)	25	Average complexity per method
Weighted Methods per Class (WMC)	8	Average number of methods per class
Depth of Inheritance (DIT)	3	Maximum inheritance depth
Number of Children (NOC)	5	Average number of child classes
Coupling Between Objects (CBO)	12	Average number of coupled classes
Response for Class (RFC)	15	Average number of methods called
Lack of Cohesion (LCOM)	0.8	Measure of class cohesion

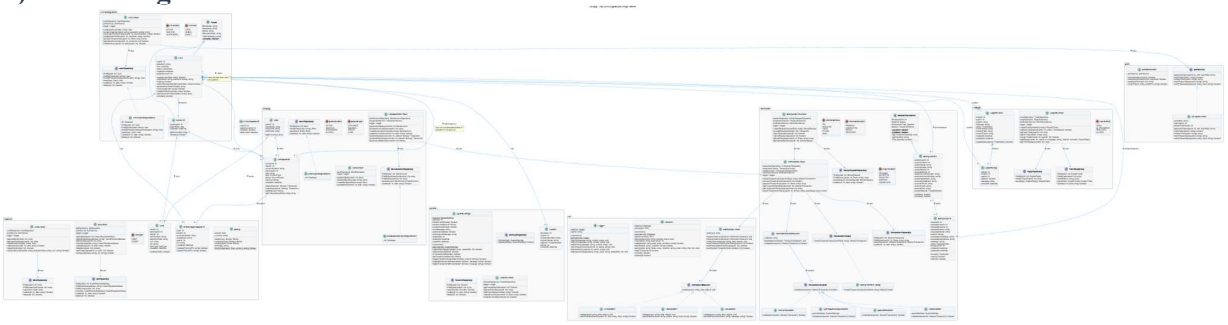
11. Testing Reports

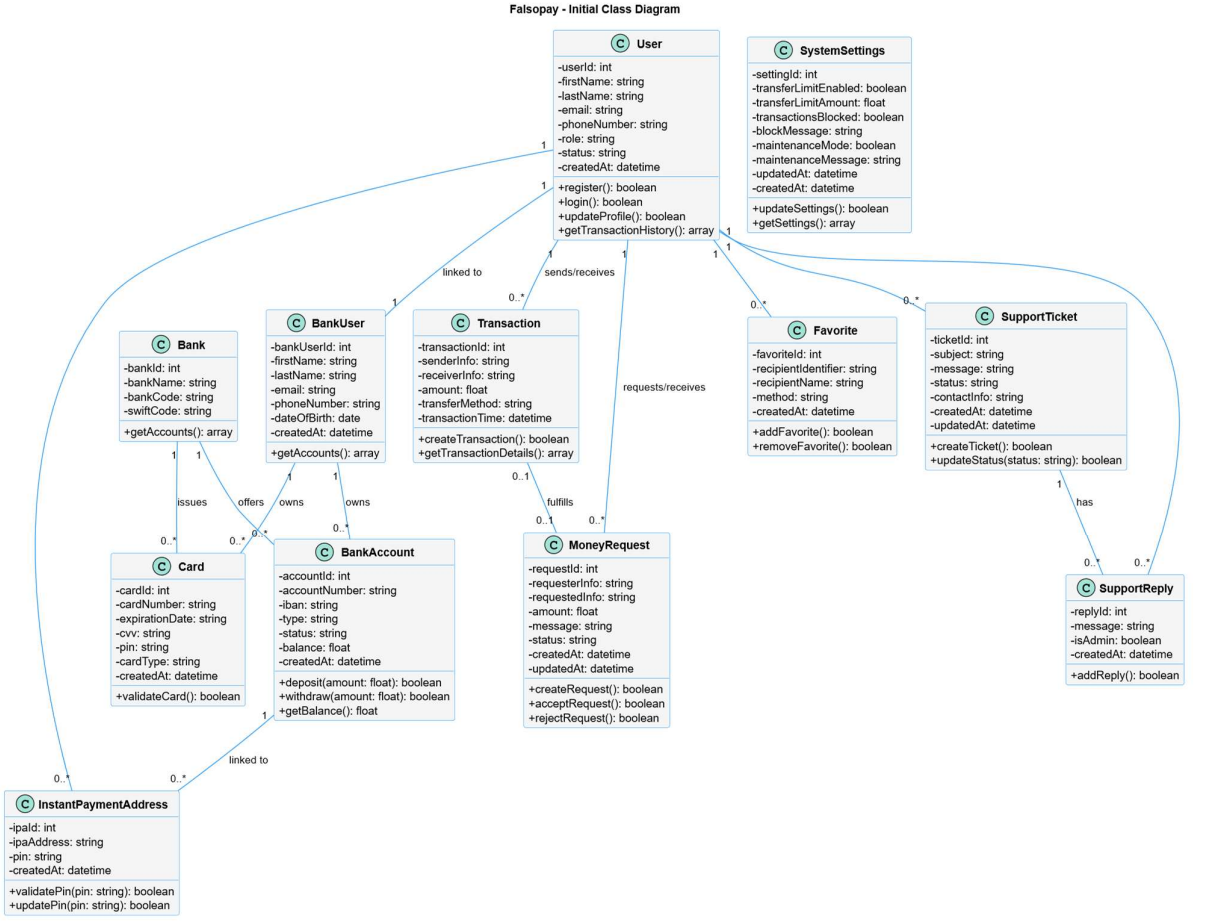
Test Type	Coverage	Results
Unit Tests	83%	30 tests passed, 6 failed
Integration Tests	93%	28 tests passed, 2 failed
System Tests	90%	19 tests passed, 1 failed
User Acceptance Tests	100%	10 tests passed, 0 failed

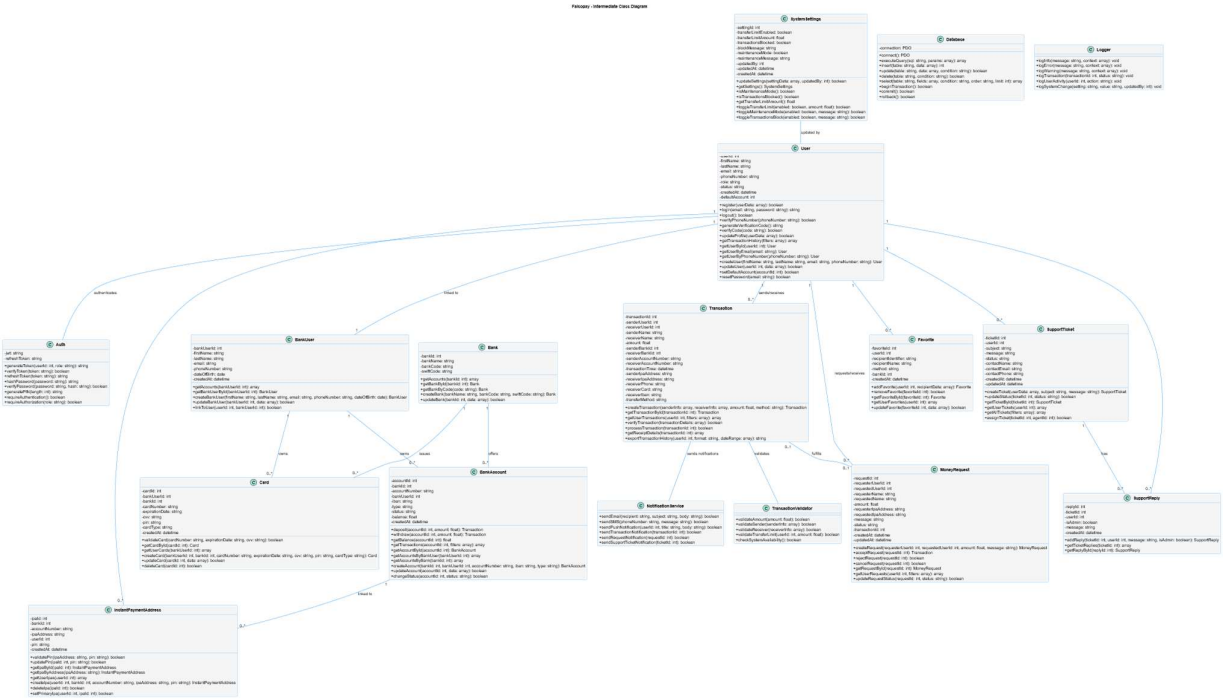
Appendix

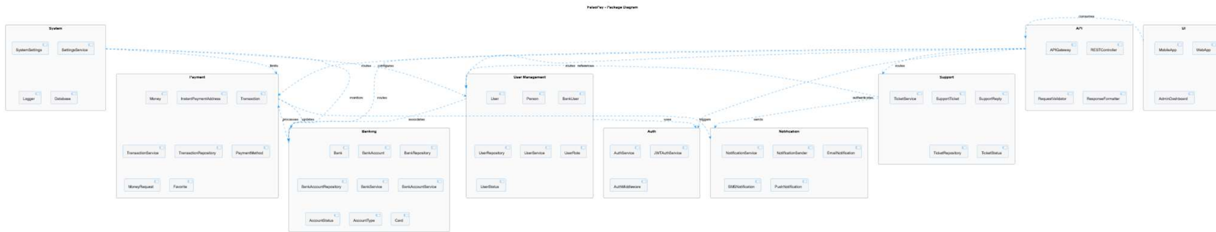
12. Structural & Behavioural Diagrams

a) Class Diagrams

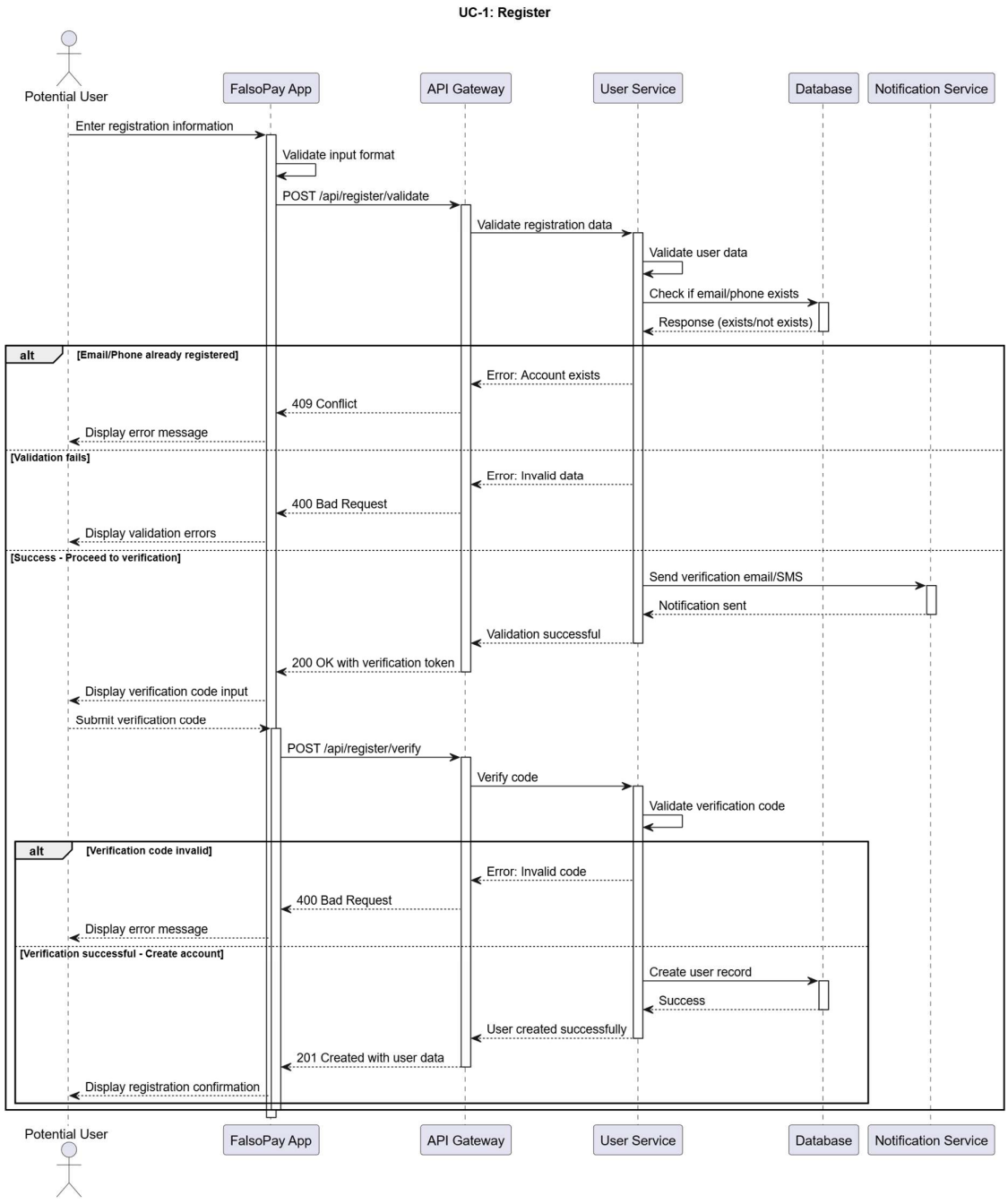


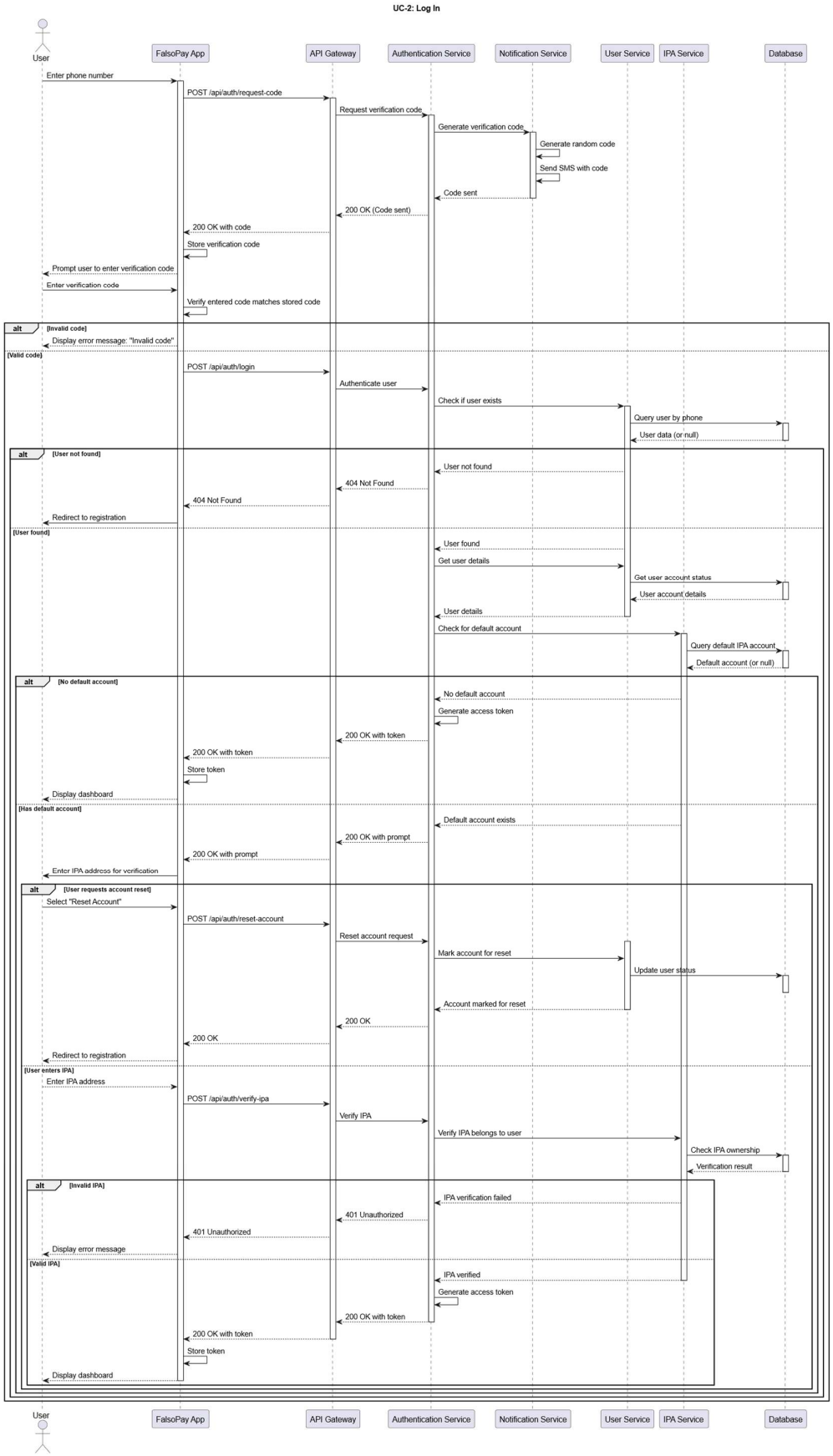

Falsopay - Initial Class Diagram
User
-userId: int
-firstName: string
-lastName: string
-email: string
-phoneNumber: string
-role: string
-status: string
-createdAt: datetime
+register(): boolean
+login(): boolean
+updateProfile(): boolean
+getTransactionHistory(): array
SystemSettings
-settingId: int
-transferLimitEnabled: boolean
-transferLimitAmount: float
-transactionsBlocked: boolean
-blockMessage: string
-maintenanceMode: boolean
-maintenanceMessage: string
-updatedAt: datetime
-createdAt: datetime
+updateSettings(): boolean
+getSettings(): array
Bank
-bankId: int
-bankName: string
-bankCode: string
-swiftCode: string
+getAccounts(): array
BankUser
-bankUserId: int
-firstName: string
-lastName: string
-email: string
-phoneNumber: string
-dateOfBirth: date
-createdAt: datetime
+getAccounts(): array
Transaction
-transactionId: int
-senderInfo: string
-receiverInfo: string
-amount: float
-transferMethod: string
-transactionTime: datetime
+createTransaction(): boolean
+getTransactionDetails(): array
Favorite
-favoriteId: int
-recipientIdentifier: string
-recipientName: string
-method: string
-createdAt: datetime
+addFavorite(): boolean
+removeFavorite(): boolean
SupportTicket
-ticketId: int
-subject: string
-message: string
-status: string
-contactInfo: string
-createdAt: datetime
-updatedAt: datetime
+createTicket(): boolean
+updateStatus(status: string): boolean
Card
-cardId: int
-cardNumber: string
-expirationDate: string
-cvv: string
-pin: string
-cardType: string
-createdAt: datetime
+validateCard(): boolean
BankAccount
-accountId: int
-accountNumber: string
-iban: string
-type: string
-status: string
-balance: float
-createdAt: datetime
+deposit(amount: float): boolean
+withdraw(amount: float): boolean
+getBalance(): float
MoneyRequest
-requestId: int
-requesterInfo: string
-requestedInfo: string
-amount: float
-message: string
-status: string
-createdAt: datetime
-updatedAt: datetime
+createRequest(): boolean
+acceptRequest(): boolean
+rejectRequest(): boolean
SupportReply
-replyId: int
-message: string
-isAdmin: boolean
-createdAt: datetime
+addReply(): boolean
InstantPaymentAddress
-ipaId: int
-ipaAddress: string
-pin: string
-createdAt: datetime
+validatePin(pin: string): boolean
+updatePin(pin: string): boolean
linked to
sends/receives
requests/receives
issues
offers
owns
owns
fulfills
has
linked to



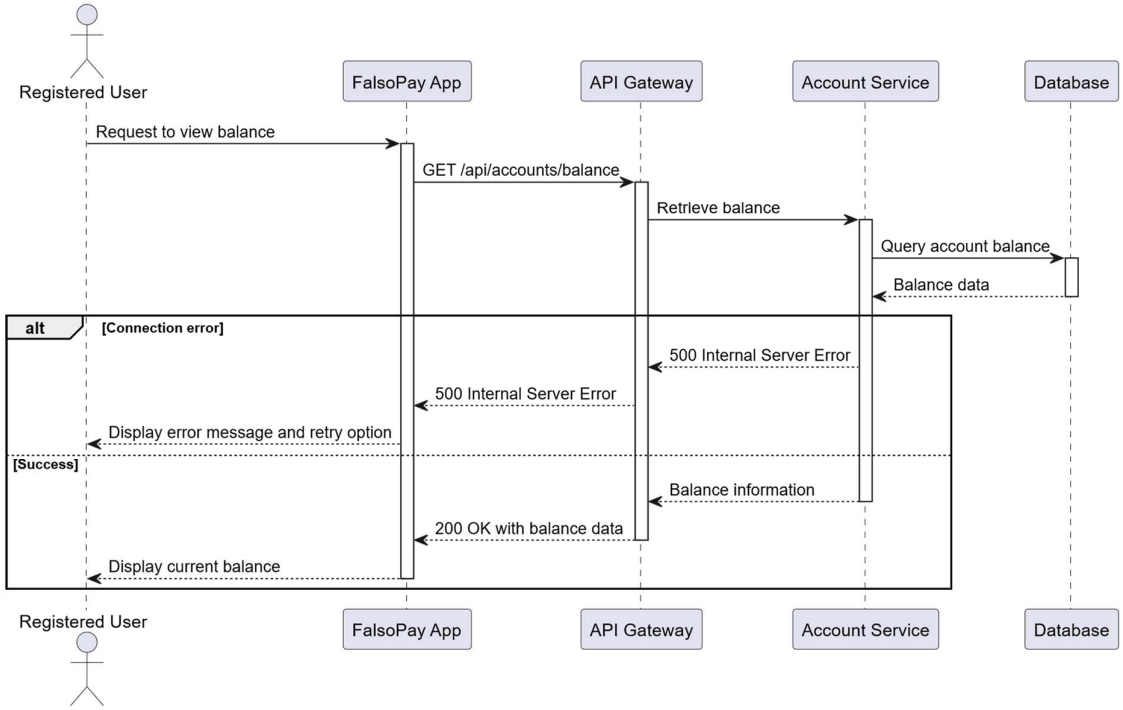

System
SystemSettings
SettingsService
Logger
Database
Payment
Money
InstantPaymentAddress
Transaction
TransactionService
TransactionRepository
PaymentMethod
MoneyRequest
Favorite
Banking
Bank
BankAccount
BankRepository
BankAccountRepository
BankService
BankAccountService
AccountStatus
AccountType
Card
User Management
User
Person
BankUser
UserRepository
UserService
UserRole
UserStatus
Auth
AuthService
JWTAuthService
AuthMiddleware
Notification
NotificationService
NotificationSender
EmailNotification
SMSNotification
PushNotification
Support
TicketService
SupportTicket
SupportReply
TicketRepository
TicketStatus
API
APIGateway
RESTController
RequestValidator
ResponseFormatter
UI
MobileApp
WebApp
AdminDashboard
limits
routes
configures
monitors
processes
updates
associates
references
uses
triggers
sends
consumes

b) Sequence Diagrams

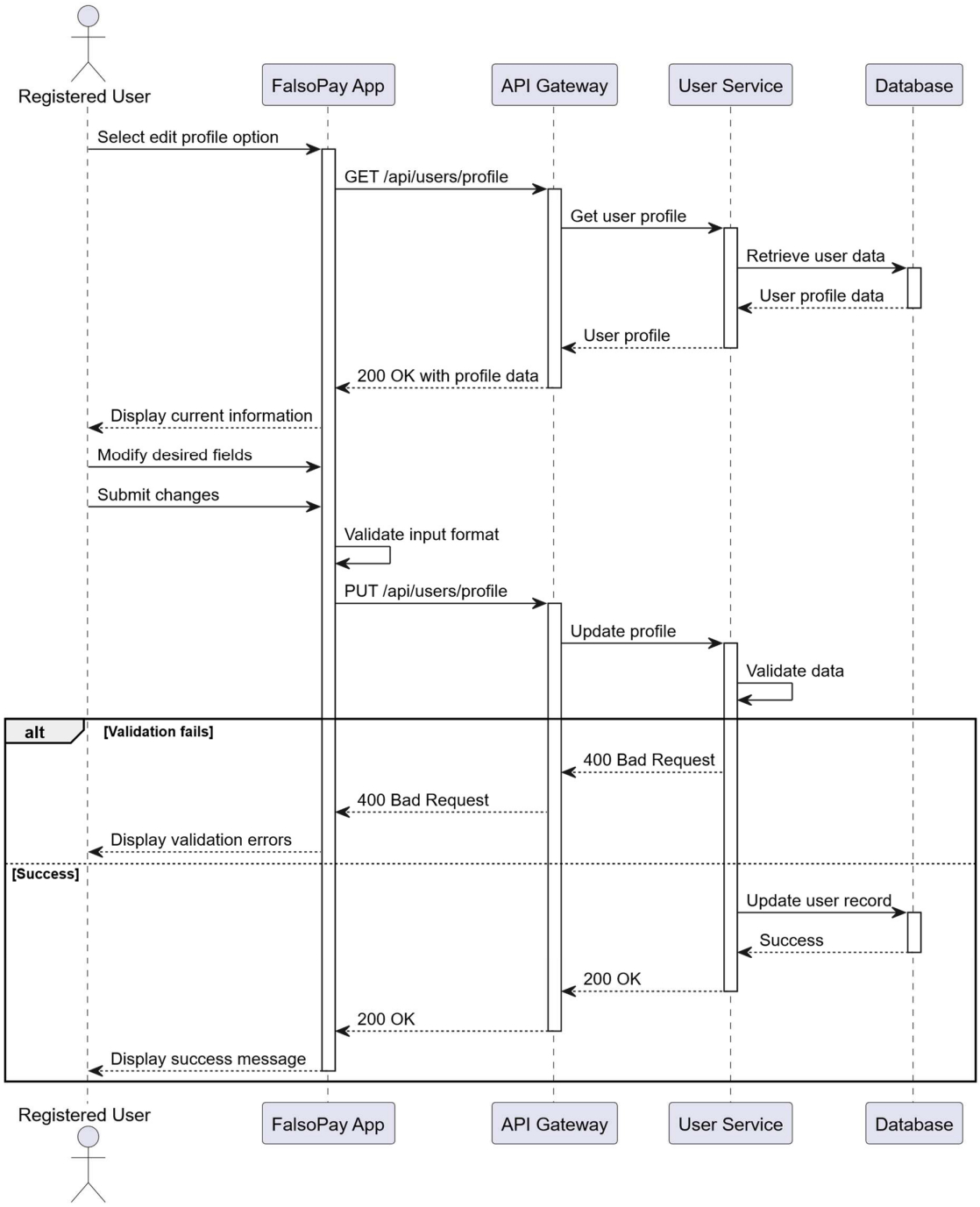


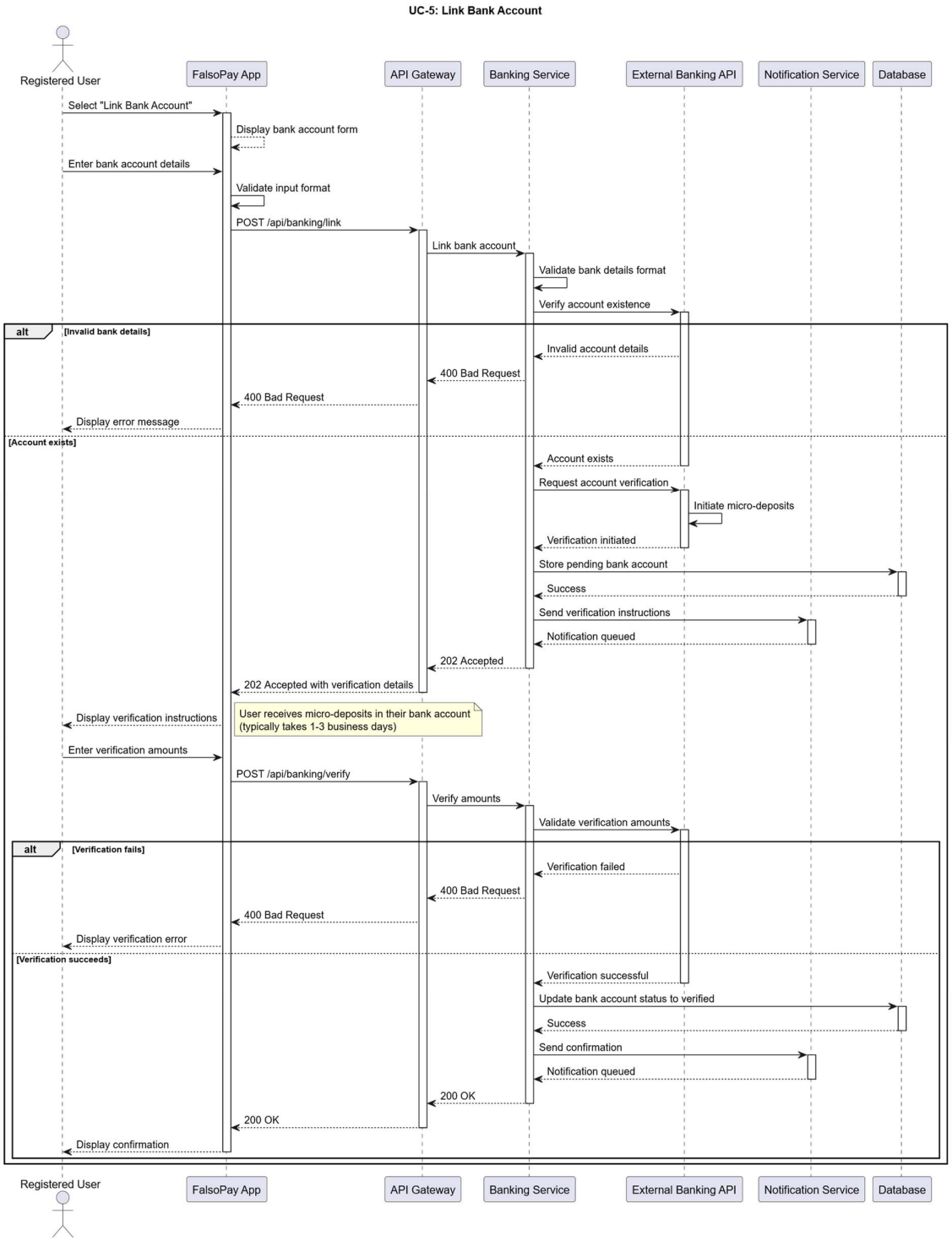

UC-2: Log In
User
FalsoPay App
API Gateway
Authentication Service
Notification Service
User Service
IPA Service
Database
Enter phone number
POST /api/auth/request-code
Request verification code
Generate verification code
Generate random code
Send SMS with code
Code sent
200 OK (Code sent)
200 OK with code
Store verification code
Prompt user to enter verification code
Enter verification code
Verify entered code matches stored code
alt
[Invalid code]
Display error message: "Invalid code"
[Valid code]
POST /api/auth/login
Authenticate user
Check if user exists
Query user by phone
User data (or null)
alt
[User not found]
User not found
404 Not Found
404 Not Found
Redirect to registration
[User found]
User found
Get user details
Get user account status
User account details
User details
Check for default account
Query default IPA account
Default account (or null)
alt
[No default account]
No default account
Generate access token
200 OK with token
200 OK with token
Store token
Display dashboard
[Has default account]
Default account exists
200 OK with prompt
200 OK with prompt
Enter IPA address for verification
alt
[User requests account reset]
Select "Reset Account"
POST /api/auth/reset-account
Reset account request
Mark account for reset
Update user status
Account marked for reset
200 OK
200 OK
Redirect to registration
[User enters IPA]
Enter IPA address
POST /api/auth/verify-ipa
Verify IPA
Verify IPA belongs to user
Check IPA ownership
Verification result
alt
[Invalid IPA]
IPA verification failed
401 Unauthorized
401 Unauthorized
Display error message
[Valid IPA]
IPA verified
Generate access token
200 OK with token
200 OK with token
Store token
Display dashboard

UC-3: Check Balance

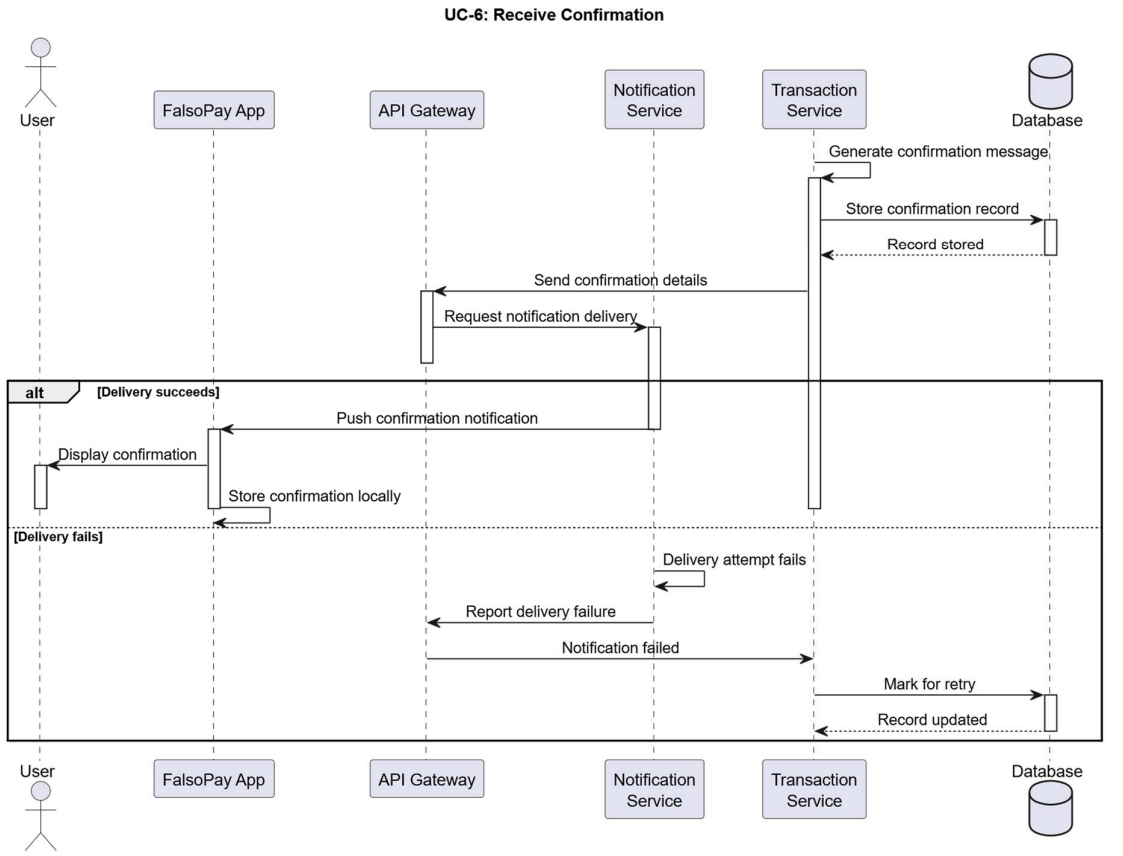


UC-4: Edit Personal Info



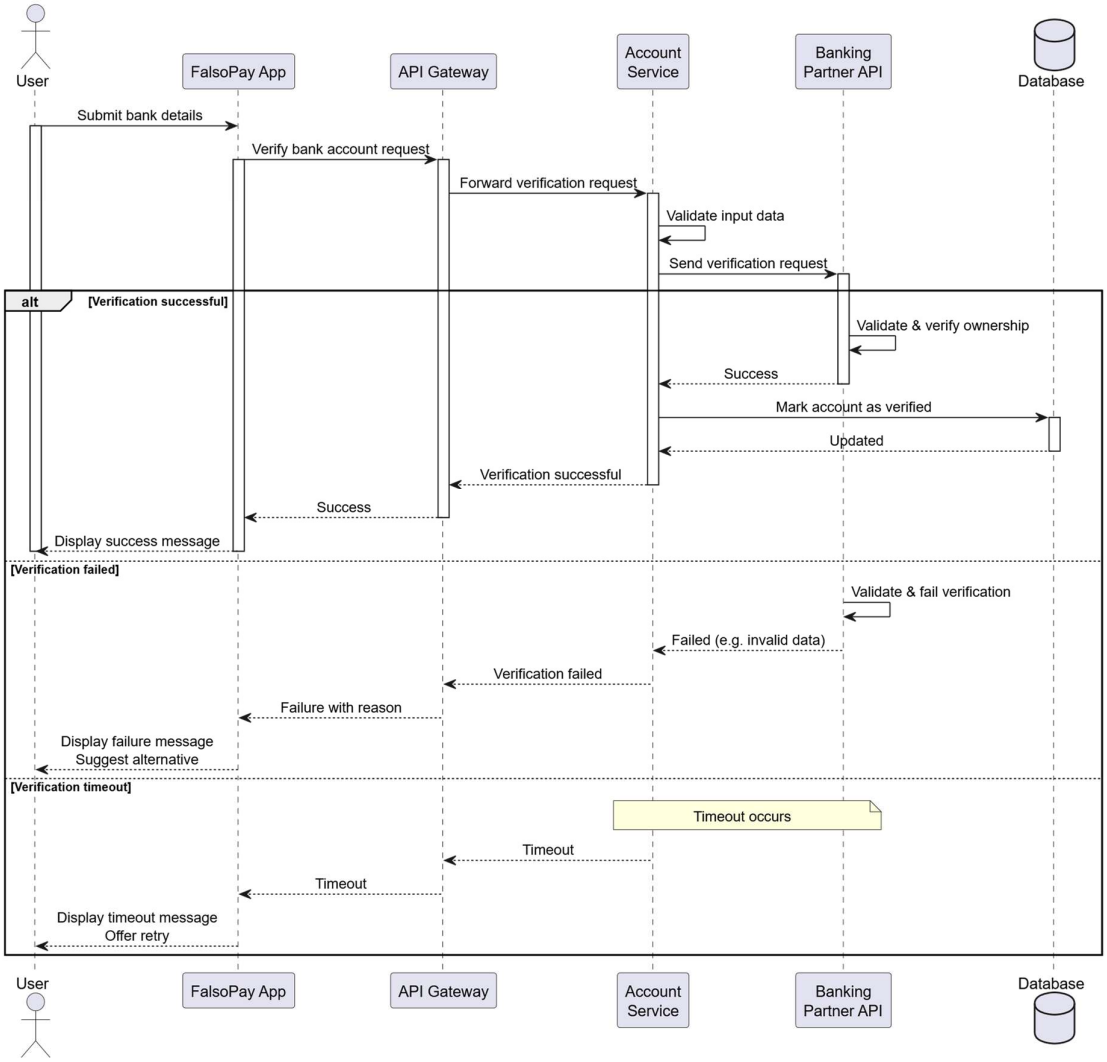

UC-5: Link Bank Account
Registered User
FalsoPay App
API Gateway
Banking Service
External Banking API
Notification Service
Database
Select "Link Bank Account"
Display bank account form
Enter bank account details
Validate input format
POST /api/banking/link
Link bank account
Validate bank details format
Verify account existence
alt
[Invalid bank details]
Invalid account details
400 Bad Request
400 Bad Request
Display error message
[Account exists]
Account exists
Request account verification
Initiate micro-deposits
Verification initiated
Store pending bank account
Success
Send verification instructions
Notification queued
202 Accepted
202 Accepted with verification details
Display verification instructions
User receives micro-deposits in their bank account
(typically takes 1-3 business days)
Enter verification amounts
POST /api/banking/verify
Verify amounts
Validate verification amounts
alt
[Verification fails]
Verification failed
400 Bad Request
400 Bad Request
Display verification error
[Verification succeeds]
Verification successful
Update bank account status to verified
Success
Send confirmation
Notification queued
200 OK
200 OK
Display confirmation
Registered User
FalsoPay App
API Gateway
Banking Service
External Banking API
Notification Service
Database

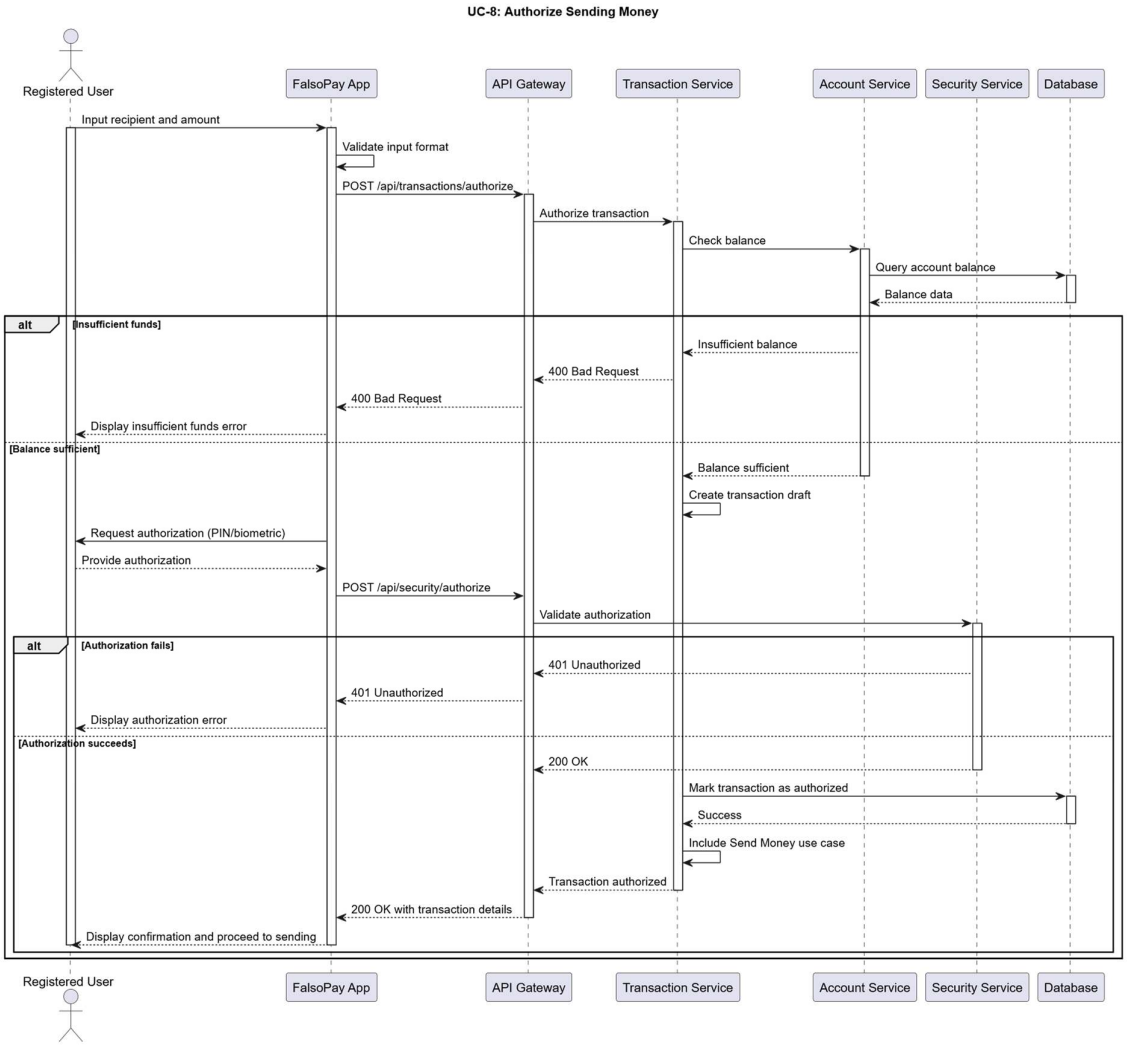
Please use '!option handwritten true' to enable handwritten

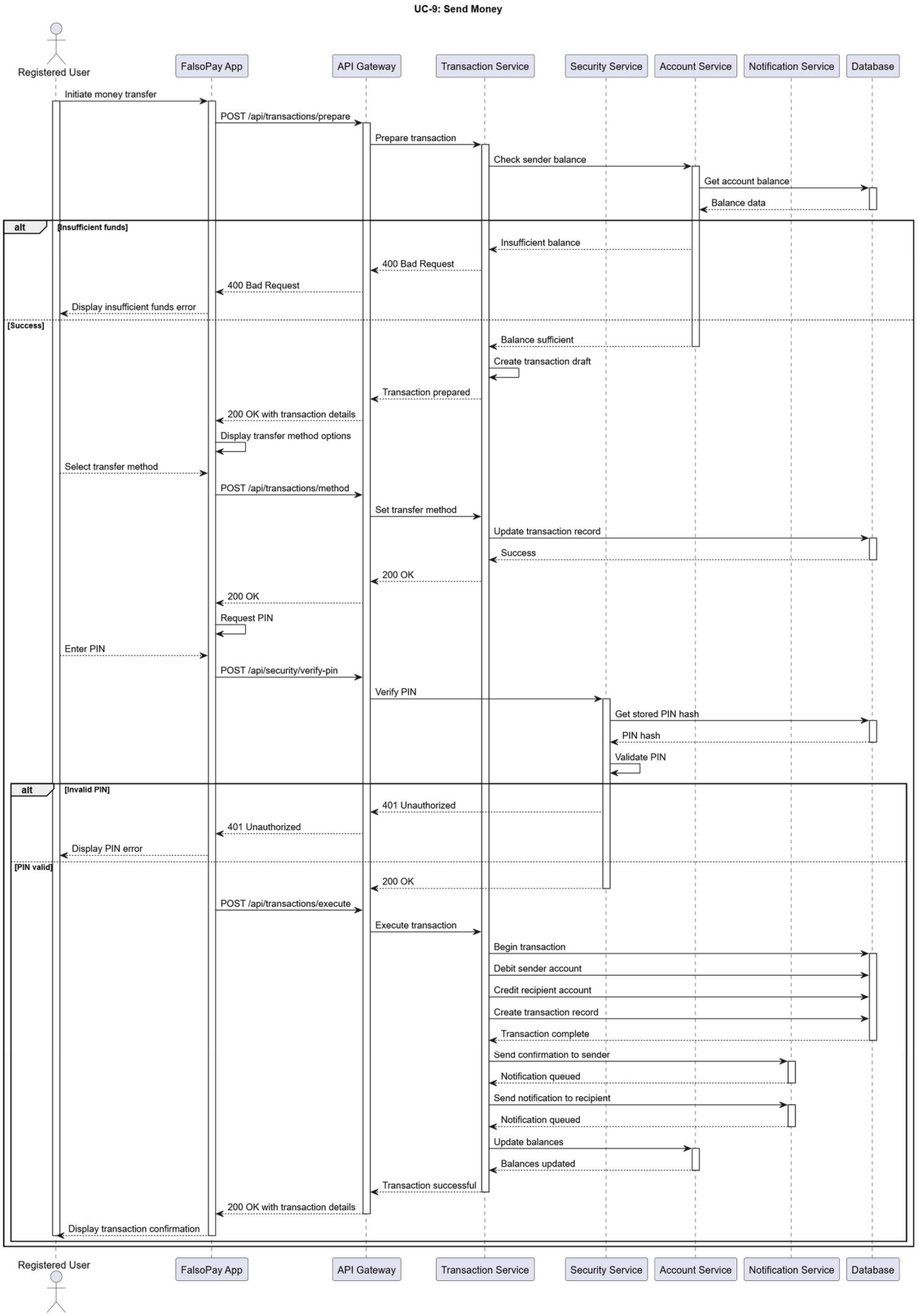


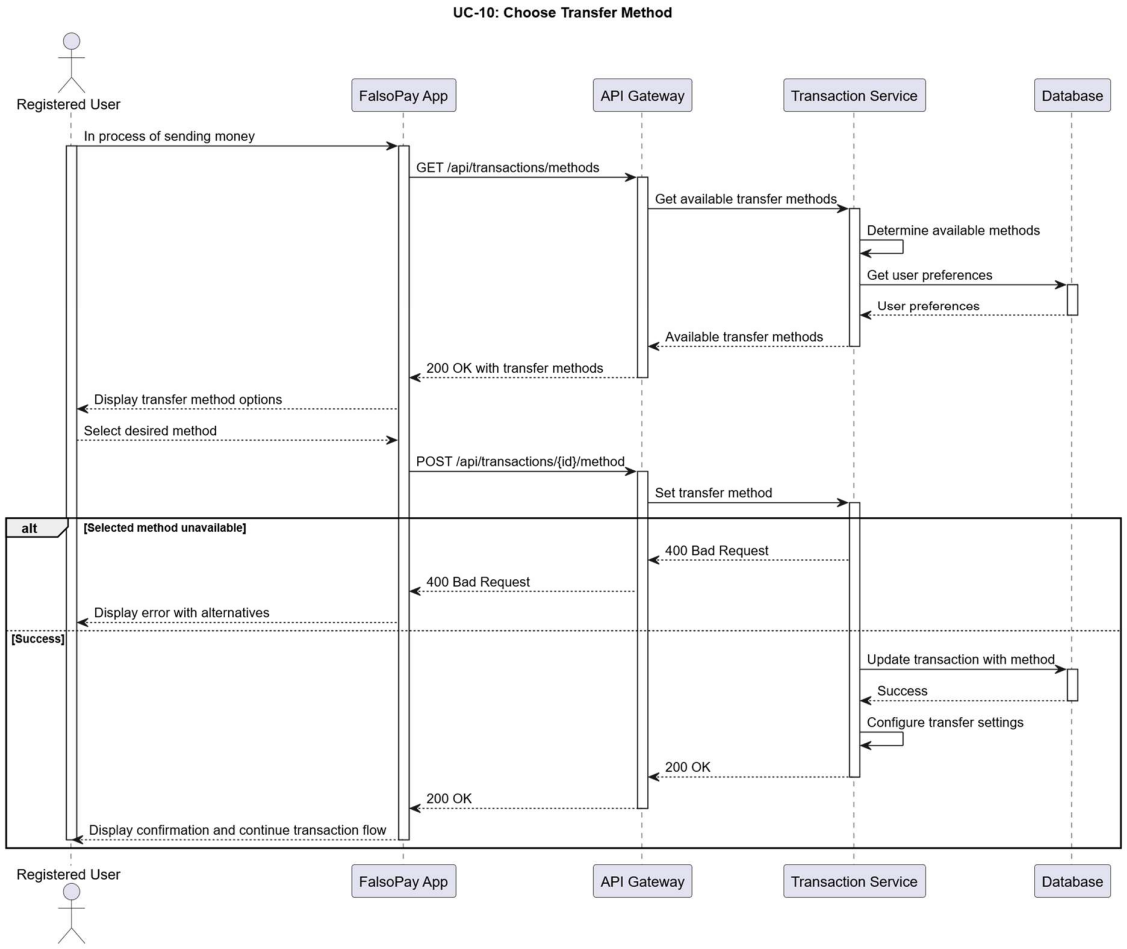
Please use '!option handwritten true' to enable handwritten

UC-7: Verify Bank Account Details

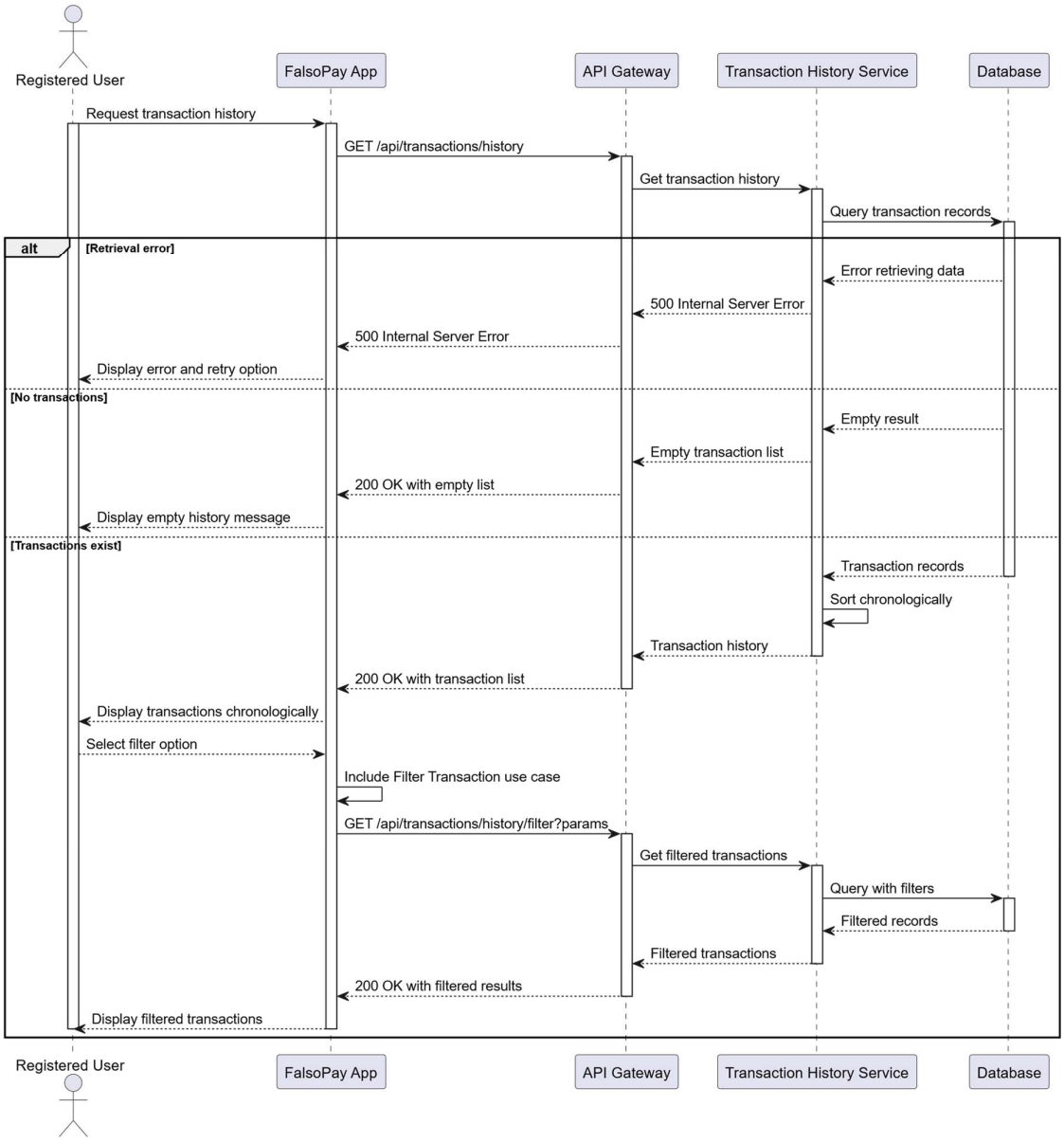


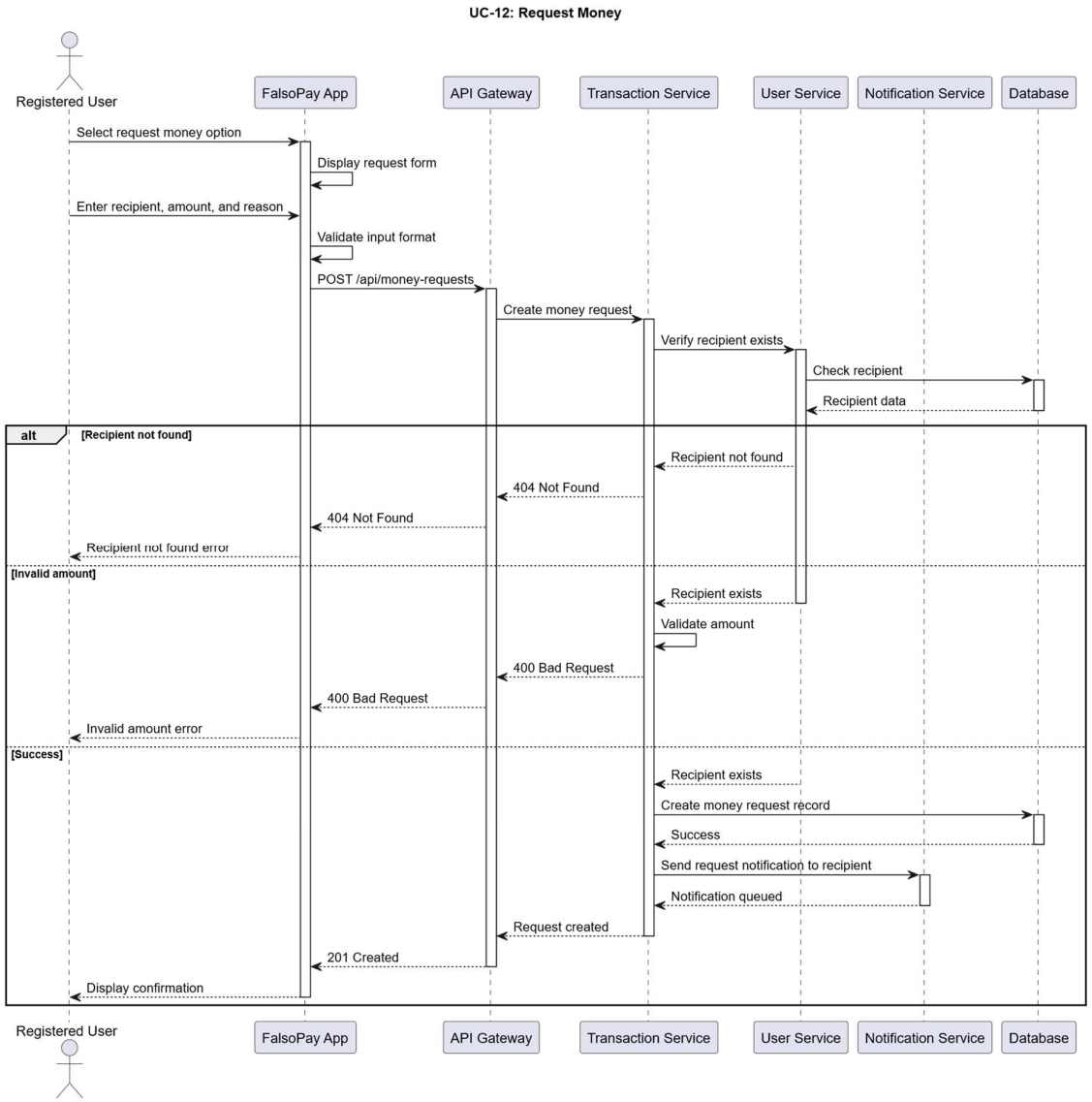

UC-8: Authorize Sending Money
Registered User
FalsoPay App
API Gateway
Transaction Service
Account Service
Security Service
Database
Input recipient and amount
Validate input format
POST /api/transactions/authorize
Authorize transaction
Check balance
Query account balance
Balance data
alt
[Insufficient funds]
Insufficient balance
400 Bad Request
400 Bad Request
Display insufficient funds error
[Balance sufficient]
Balance sufficient
Create transaction draft
Request authorization (PIN/biometric)
Provide authorization
POST /api/security/authorize
Validate authorization
alt
[Authorization fails]
401 Unauthorized
401 Unauthorized
Display authorization error
[Authorization succeeds]
200 OK
Mark transaction as authorized
Success
Include Send Money use case
Transaction authorized
200 OK with transaction details
Display confirmation and proceed to sending
Registered User
FalsoPay App
API Gateway
Transaction Service
Account Service
Security Service
Database


UC-9: Send Money
Registered User
FalsoPay App
API Gateway
Transaction Service
Security Service
Account Service
Notification Service
Database
Initiate money transfer
POST /api/transactions/prepare
Prepare transaction
Check sender balance
Get account balance
Balance data
alt
[Insufficient funds]
Insufficient balance
400 Bad Request
400 Bad Request
Display insufficient funds error
[Success]
Balance sufficient
Create transaction draft
Transaction prepared
200 OK with transaction details
Display transfer method options
Select transfer method
POST /api/transactions/method
Set transfer method
Update transaction record
Success
200 OK
200 OK
Request PIN
Enter PIN
POST /api/security/verify-pin
Verify PIN
Get stored PIN hash
PIN hash
Validate PIN
alt
[Invalid PIN]
401 Unauthorized
401 Unauthorized
Display PIN error
[PIN valid]
200 OK
POST /api/transactions/execute
Execute transaction
Begin transaction
Debit sender account
Credit recipient account
Create transaction record
Transaction complete
Send confirmation to sender
Notification queued
Send notification to recipient
Notification queued
Update balances
Balances updated
Transaction successful
200 OK with transaction details
Display transaction confirmation

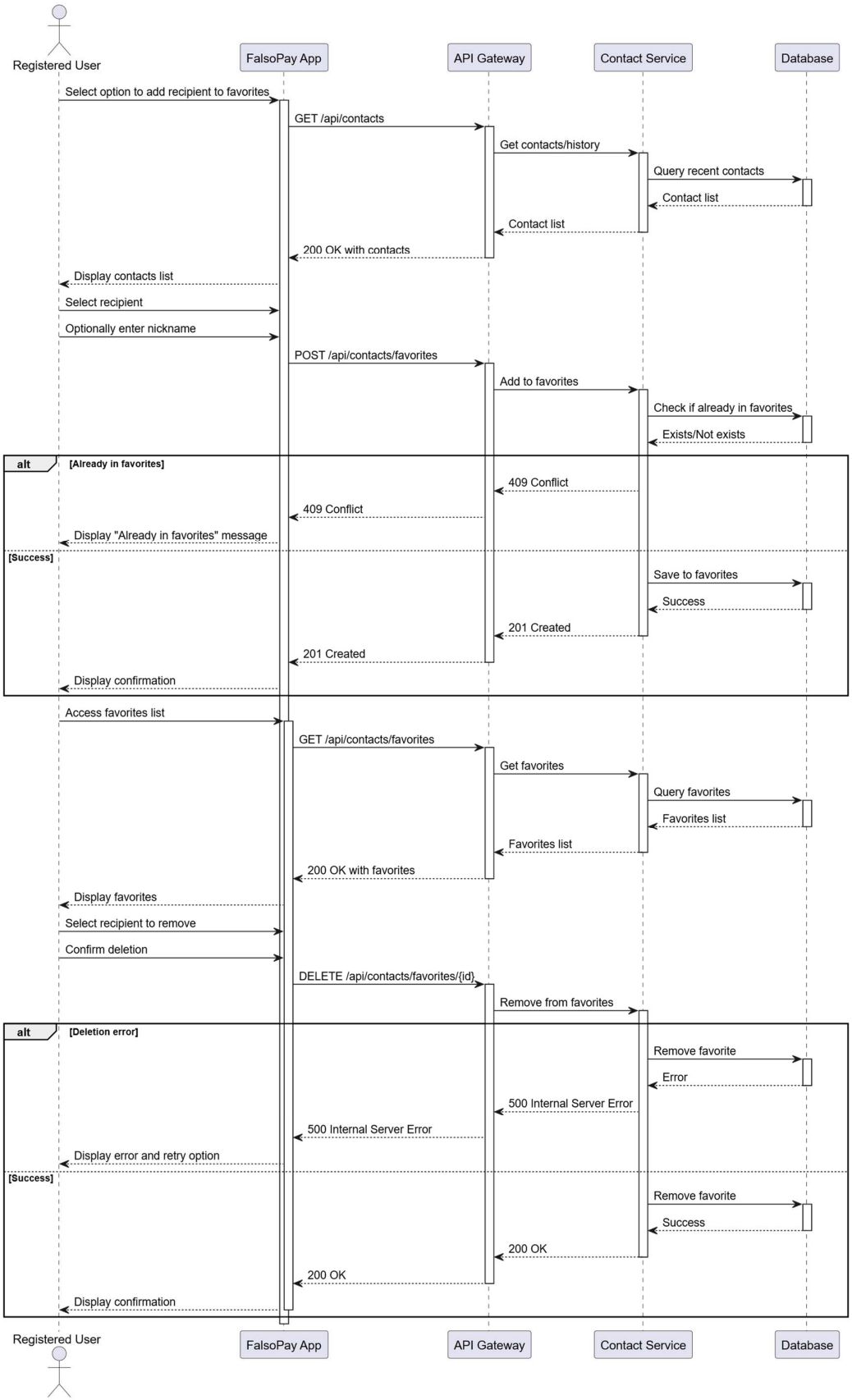

UC-10: Choose Transfer Method
Registered User
FalsoPay App
API Gateway
Transaction Service
Database
In process of sending money
GET /api/transactions/methods
Get available transfer methods
Determine available methods
Get user preferences
User preferences
Available transfer methods
200 OK with transfer methods
Display transfer method options
Select desired method
POST /api/transactions/{id}/method
Set transfer method
alt
[Selected method unavailable]
400 Bad Request
400 Bad Request
Display error with alternatives
[Success]
Update transaction with method
Success
Configure transfer settings
200 OK
200 OK
Display confirmation and continue transaction flow
Registered User
FalsoPay App
API Gateway
Transaction Service
Database

UC-11: View Transaction History

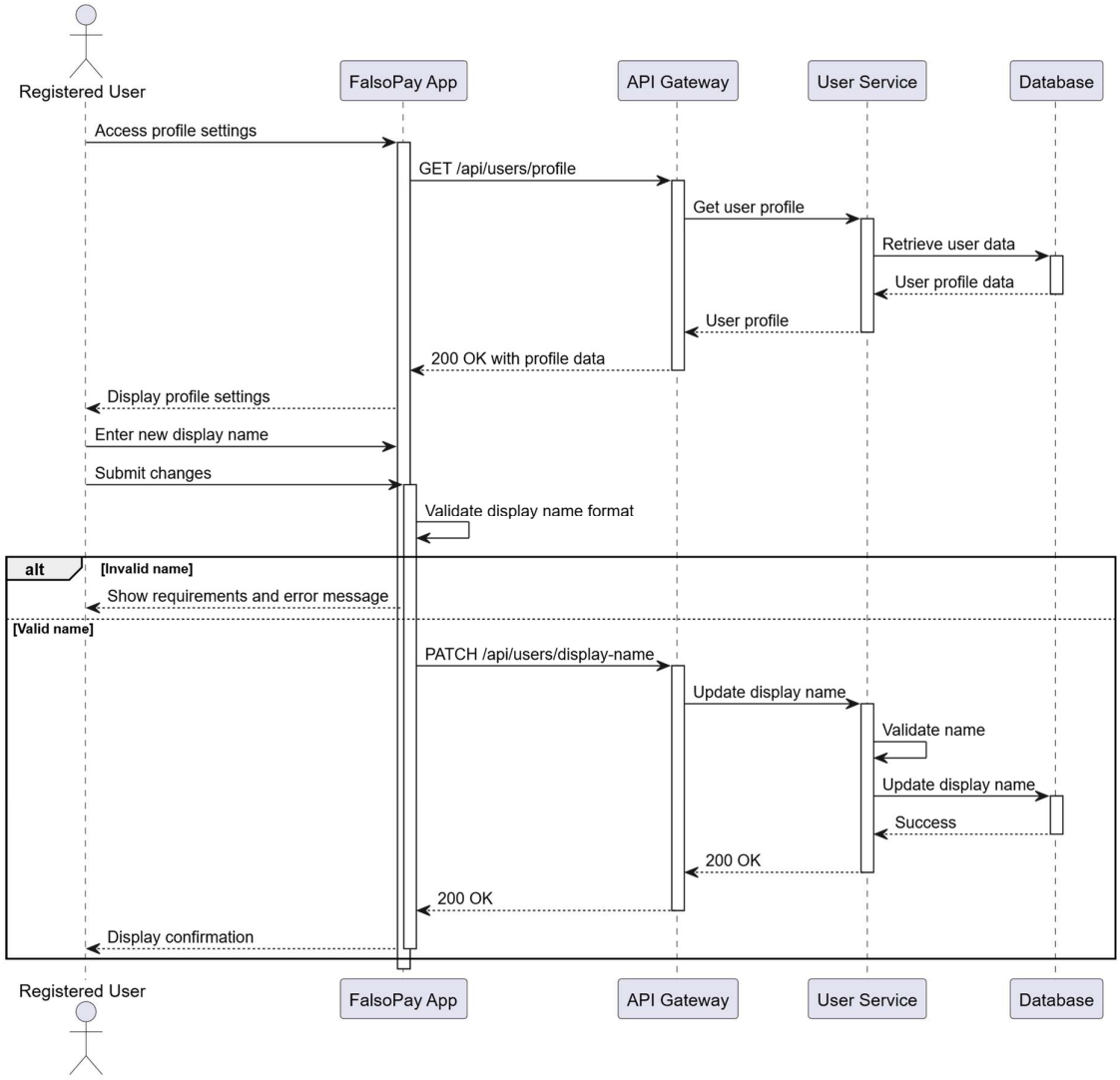



UC-12: Request Money
Registered User
FalsoPay App
API Gateway
Transaction Service
User Service
Notification Service
Database
Select request money option
Display request form
Enter recipient, amount, and reason
Validate input format
POST /api/money-requests
Create money request
Verify recipient exists
Check recipient
Recipient data
alt
[Recipient not found]
Recipient not found
404 Not Found
404 Not Found
Recipient not found error
[Invalid amount]
Recipient exists
Validate amount
400 Bad Request
400 Bad Request
Invalid amount error
[Success]
Recipient exists
Create money request record
Success
Send request notification to recipient
Notification queued
Request created
201 Created
Display confirmation
Registered User
FalsoPay App
API Gateway
Transaction Service
User Service
Notification Service
Database

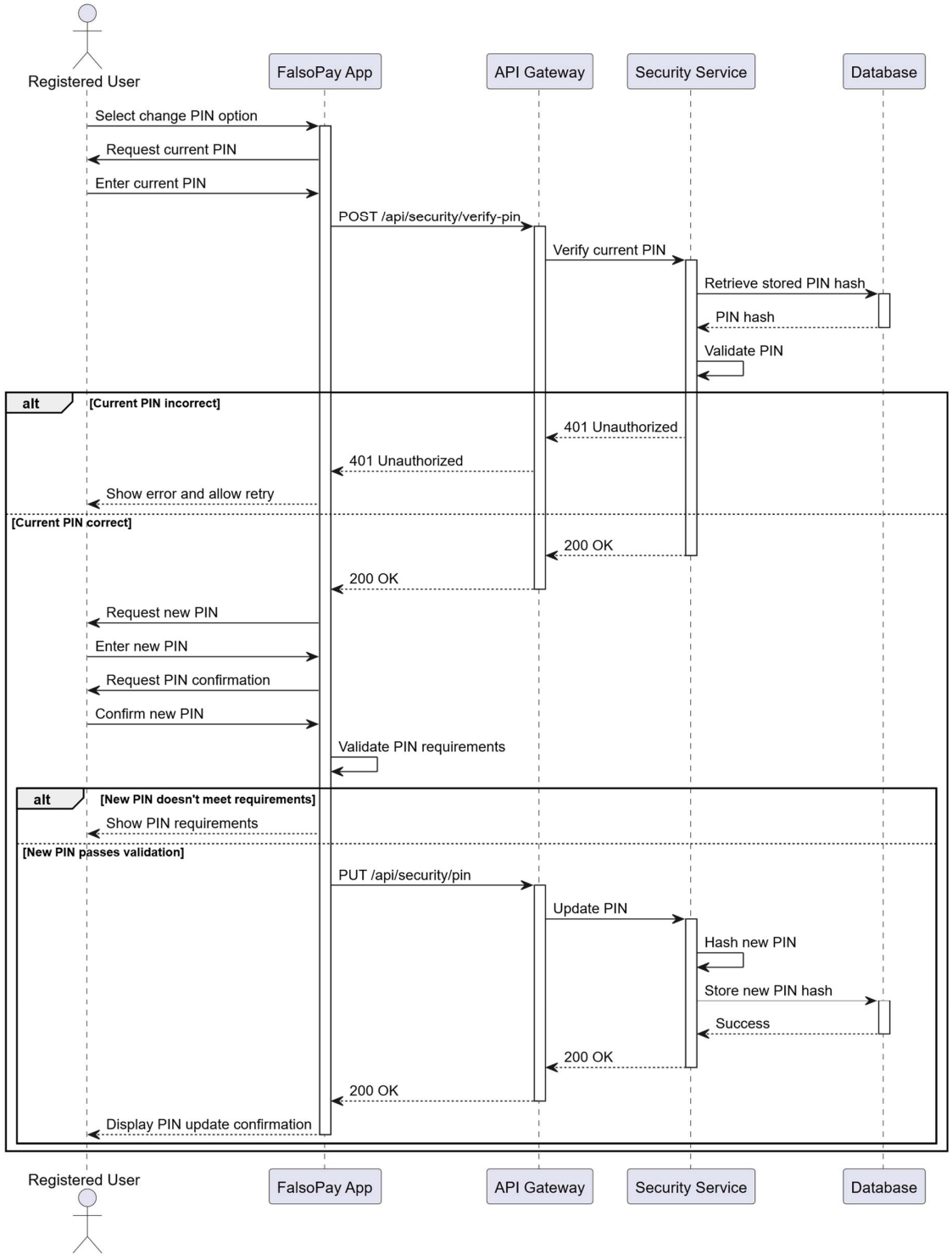
UC-13/14: Add Recipients to Favorites and Delete Favorite



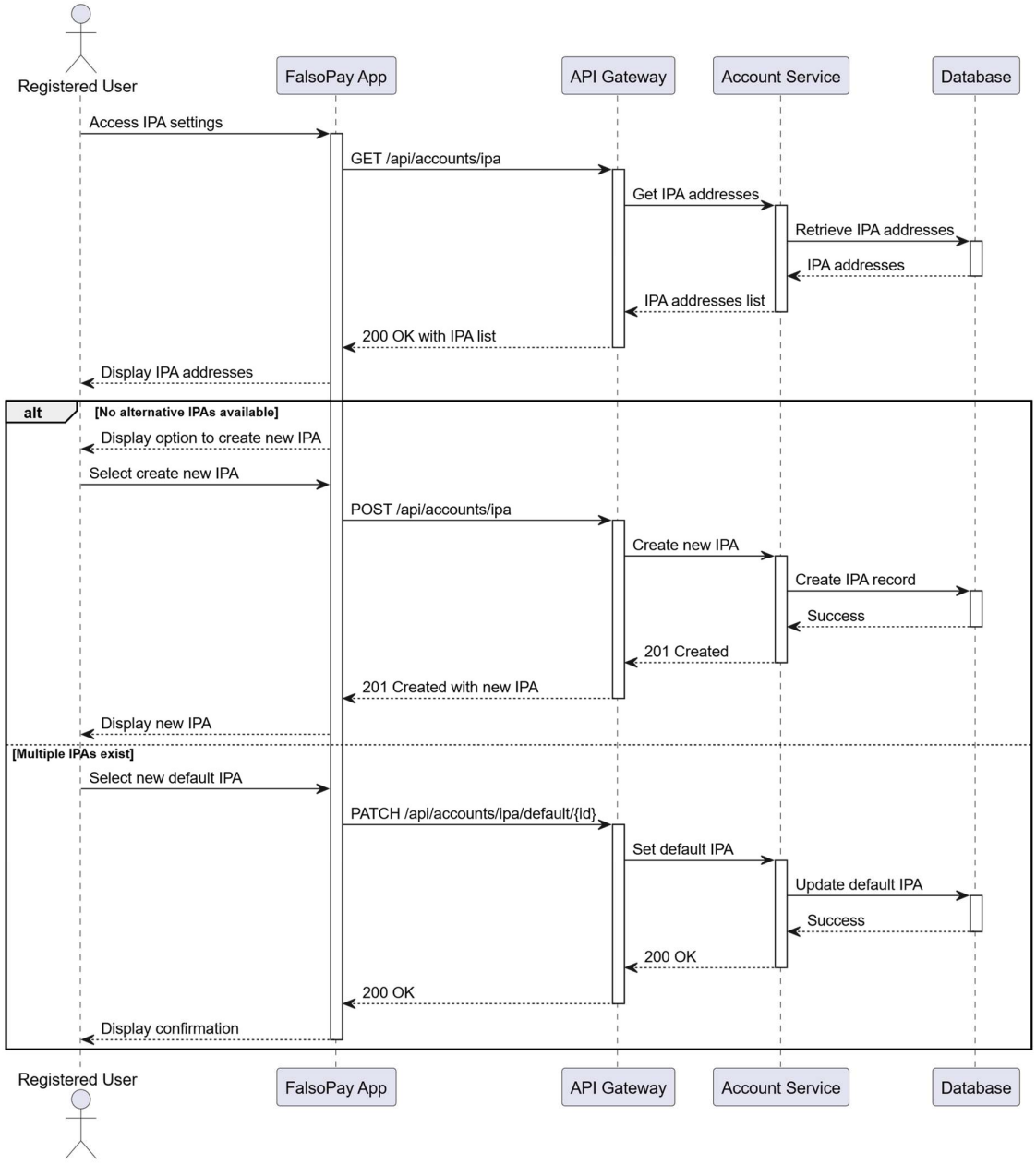
UC-15: Change Display Name

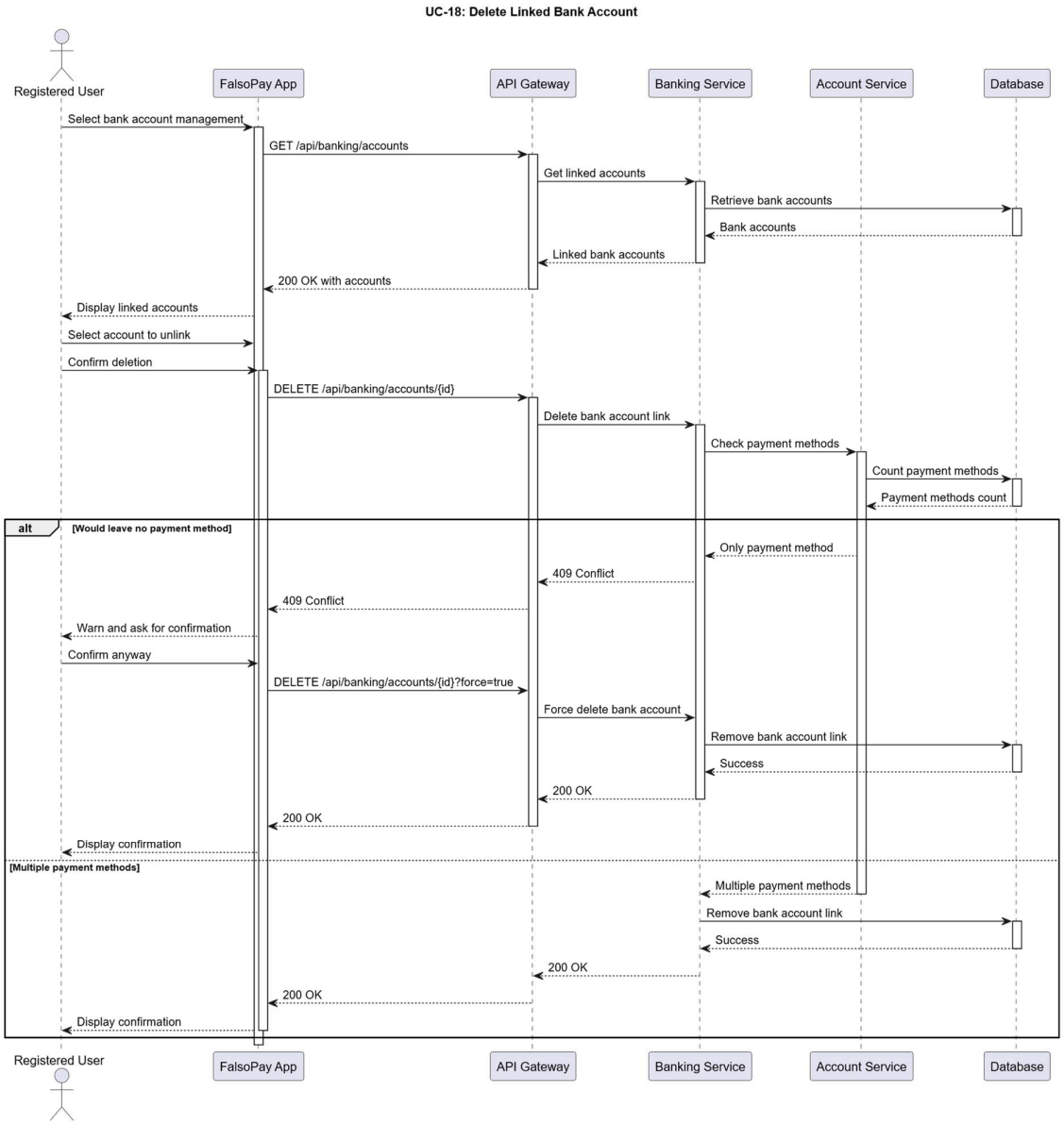


UC-16: Change PIN

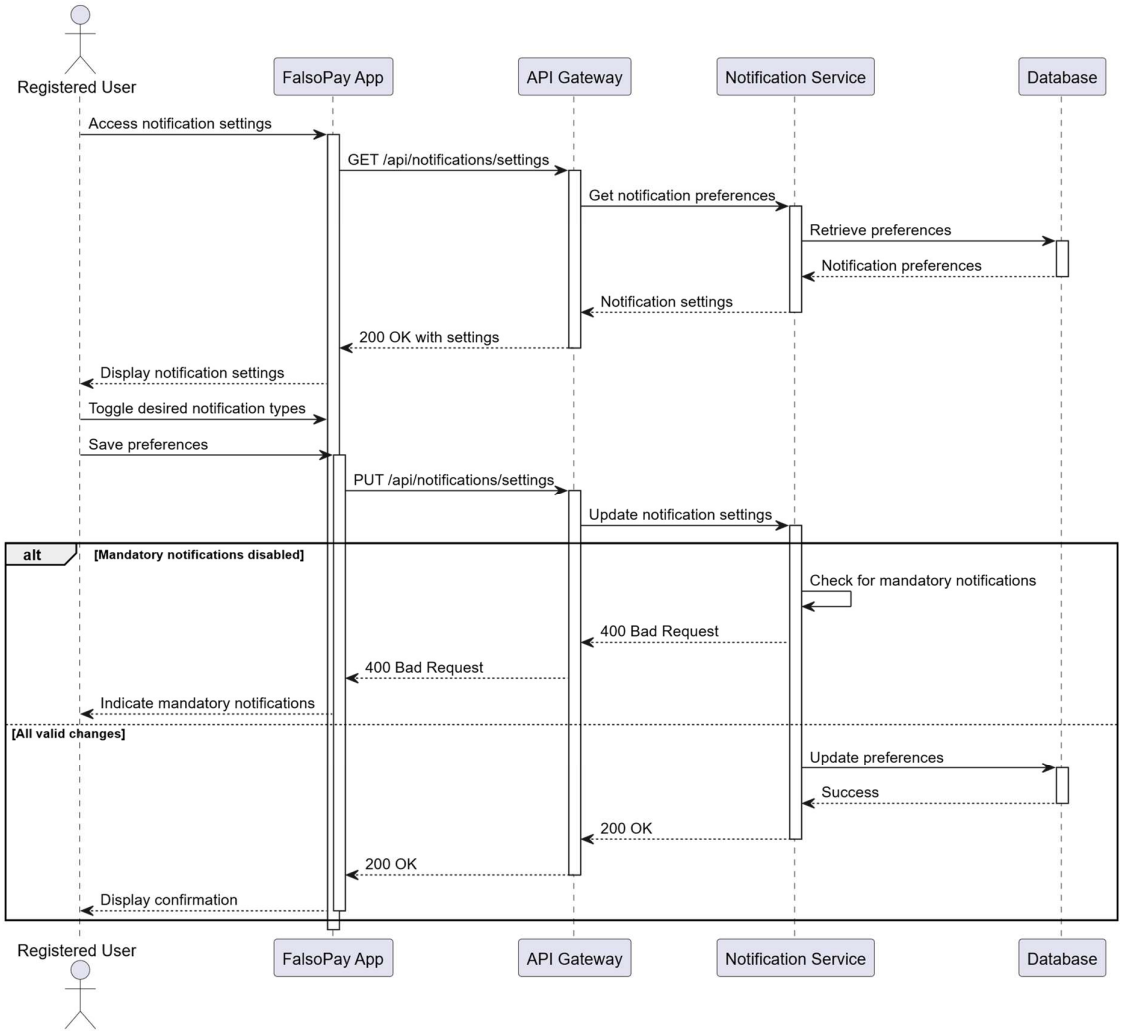


UC-17: Change Default IPA Address

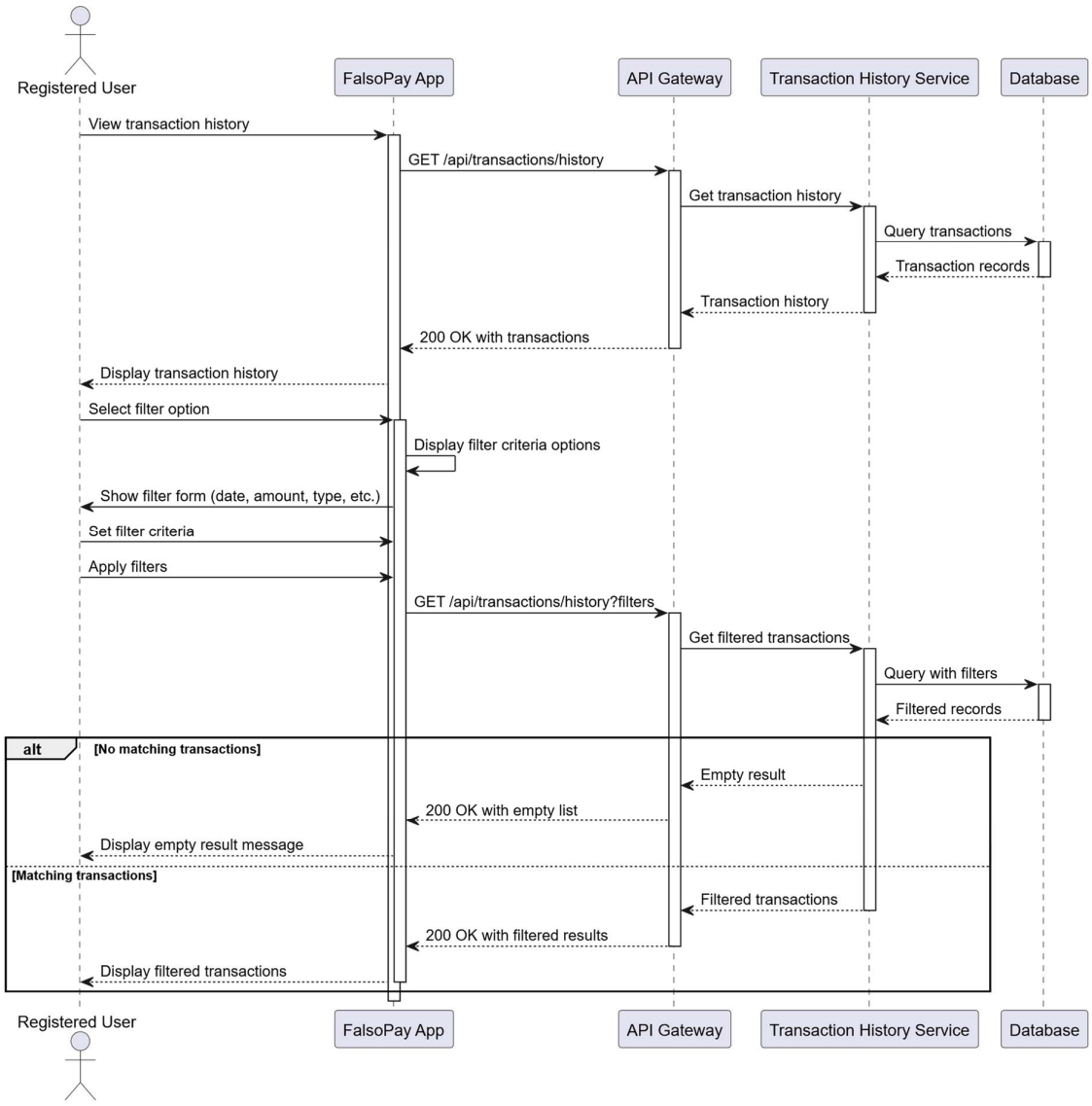



UC-18: Delete Linked Bank Account
Registered User
FalsoPay App
API Gateway
Banking Service
Account Service
Database
Select bank account management
GET /api/banking/accounts
Get linked accounts
Retrieve bank accounts
Bank accounts
Linked bank accounts
200 OK with accounts
Display linked accounts
Select account to unlink
Confirm deletion
DELETE /api/banking/accounts/{id}
Delete bank account link
Check payment methods
Count payment methods
Payment methods count
alt
[Would leave no payment method]
Only payment method
409 Conflict
409 Conflict
Warn and ask for confirmation
Confirm anyway
DELETE /api/banking/accounts/{id}?force=true
Force delete bank account
Remove bank account link
Success
200 OK
200 OK
Display confirmation
[Multiple payment methods]
Multiple payment methods
Remove bank account link
Success
200 OK
200 OK
Display confirmation
Registered User
FalsoPay App
API Gateway
Banking Service
Account Service
Database

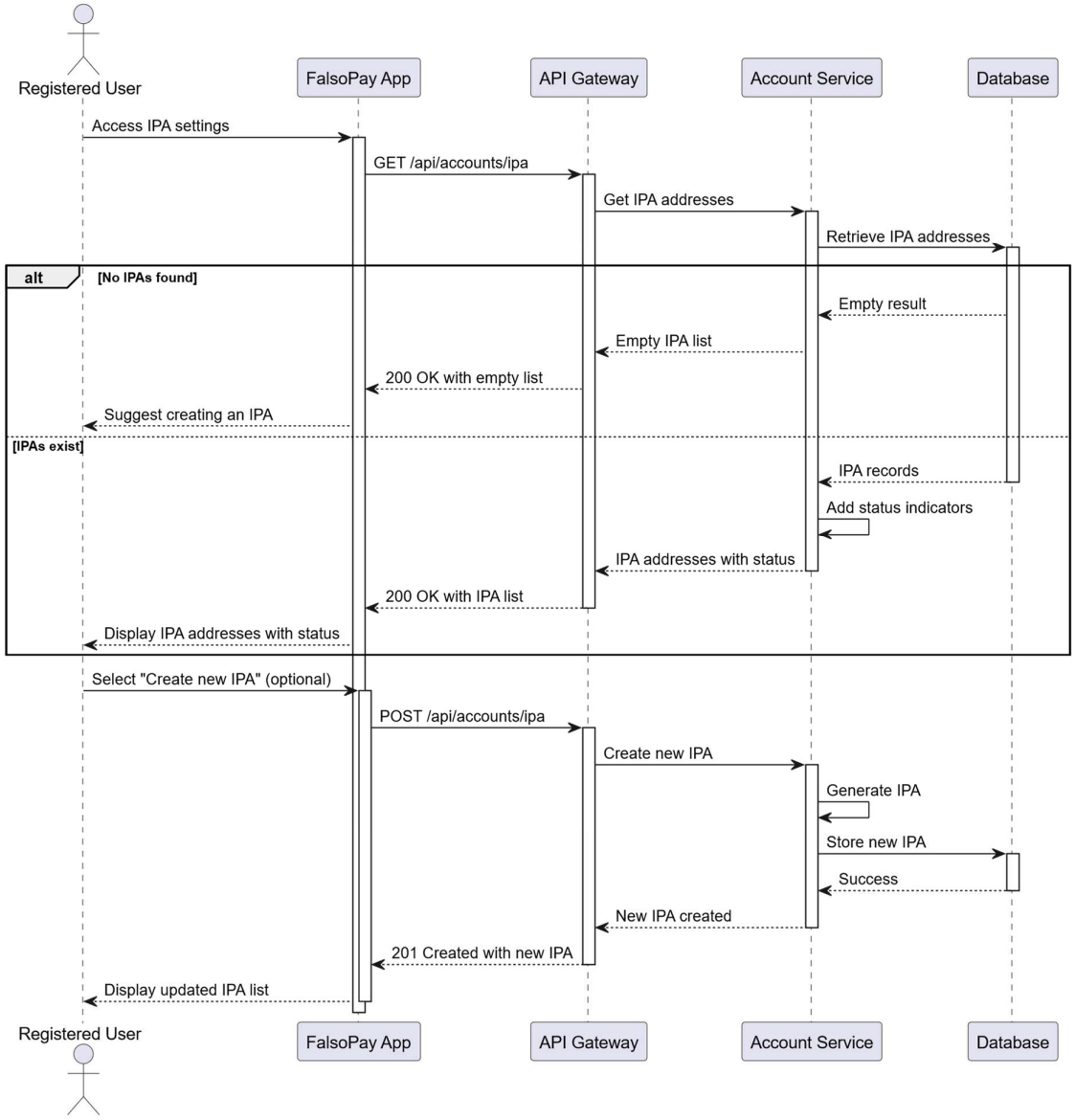
UC-19: Enable/Disable Notifications

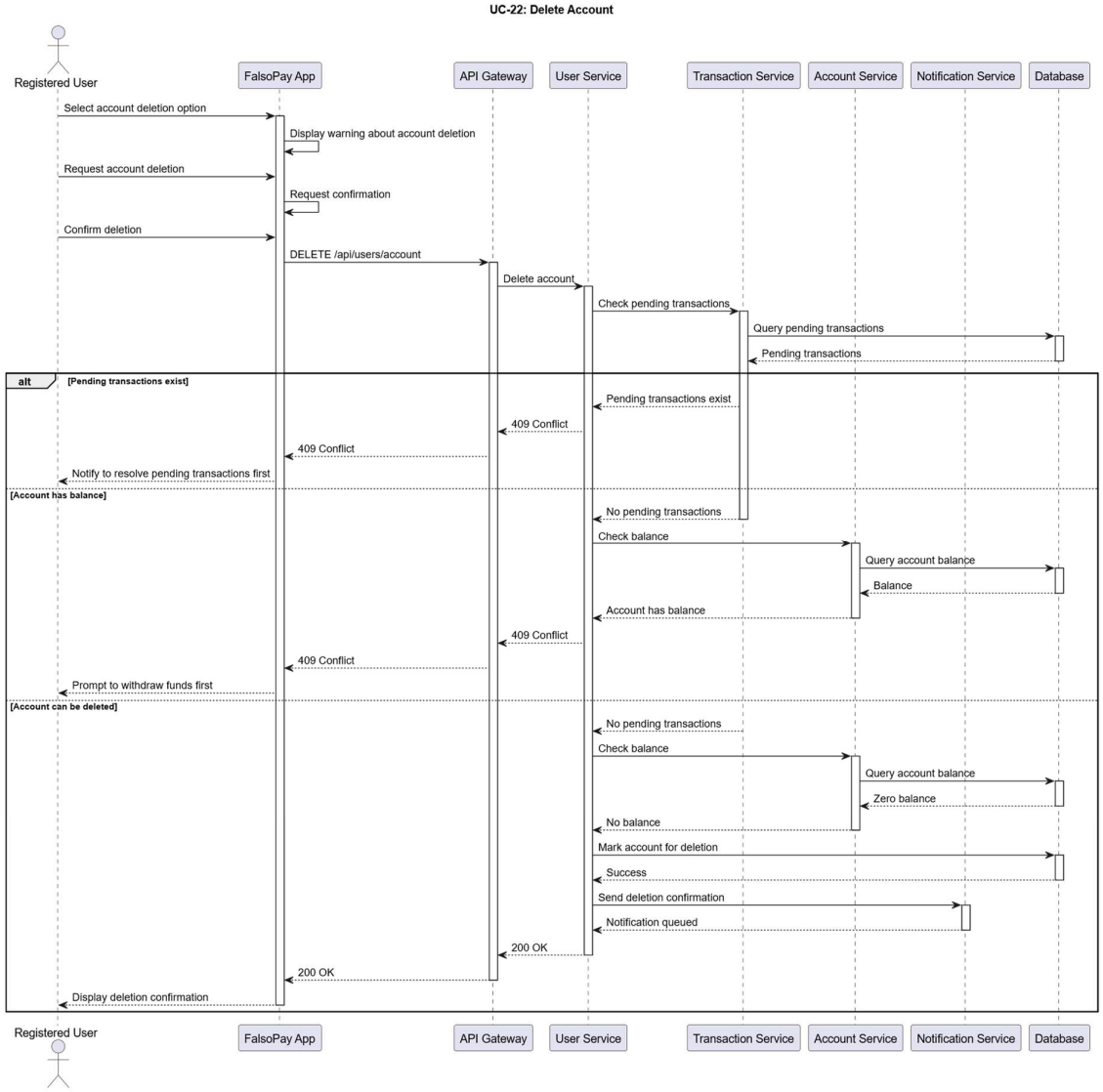


UC-20: Filter Transaction

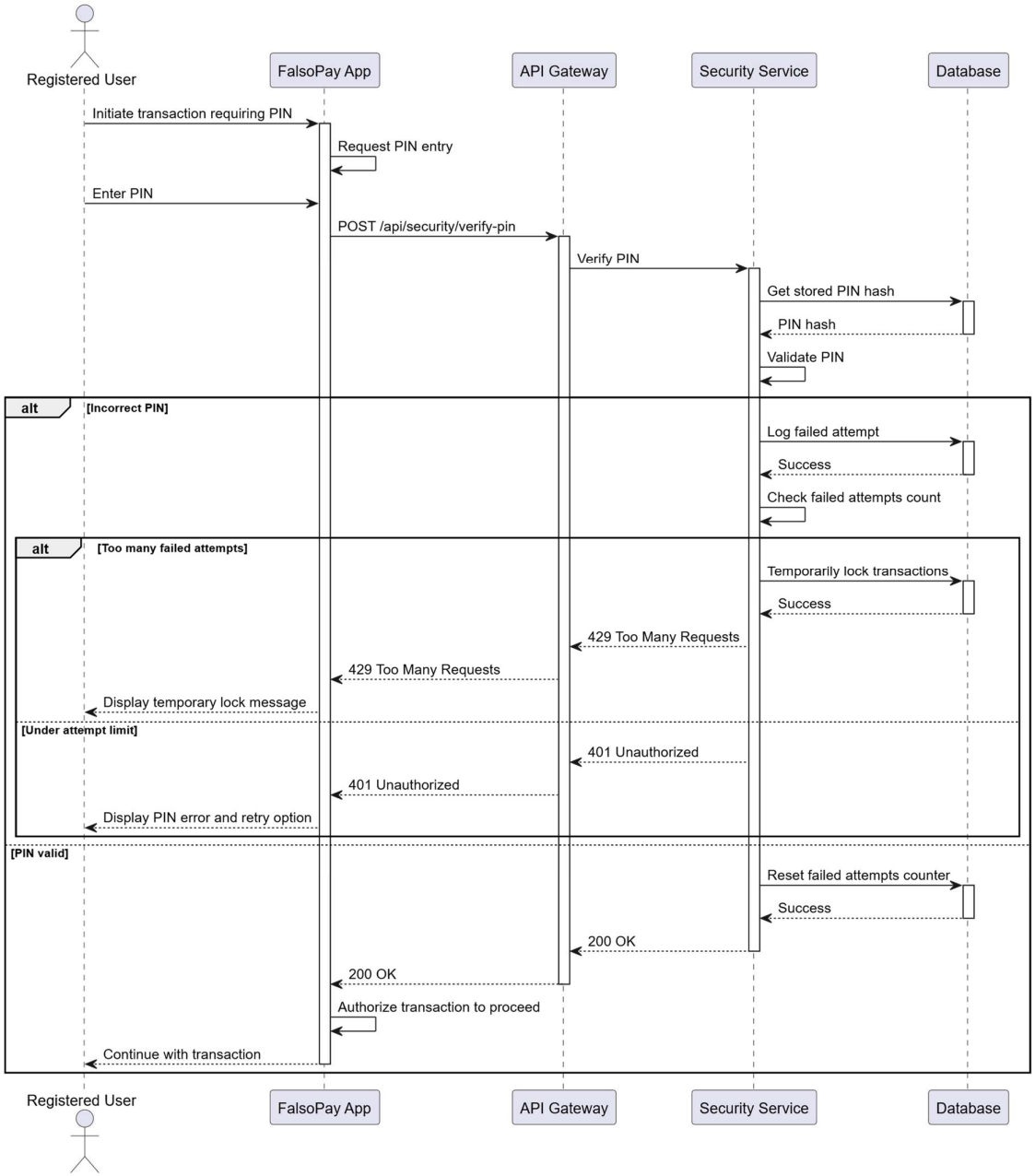


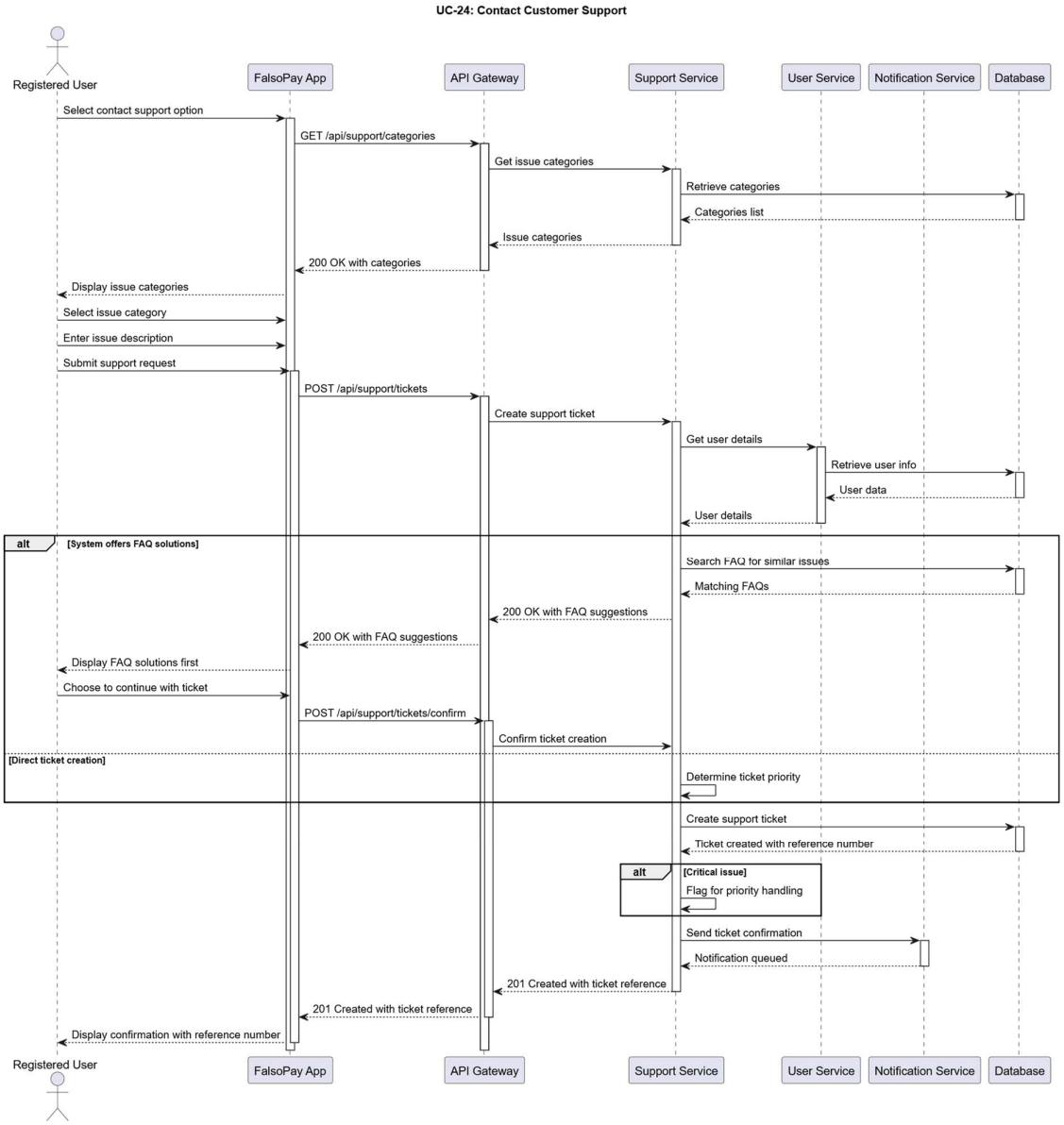
UC-21: View Your IPA Addresses

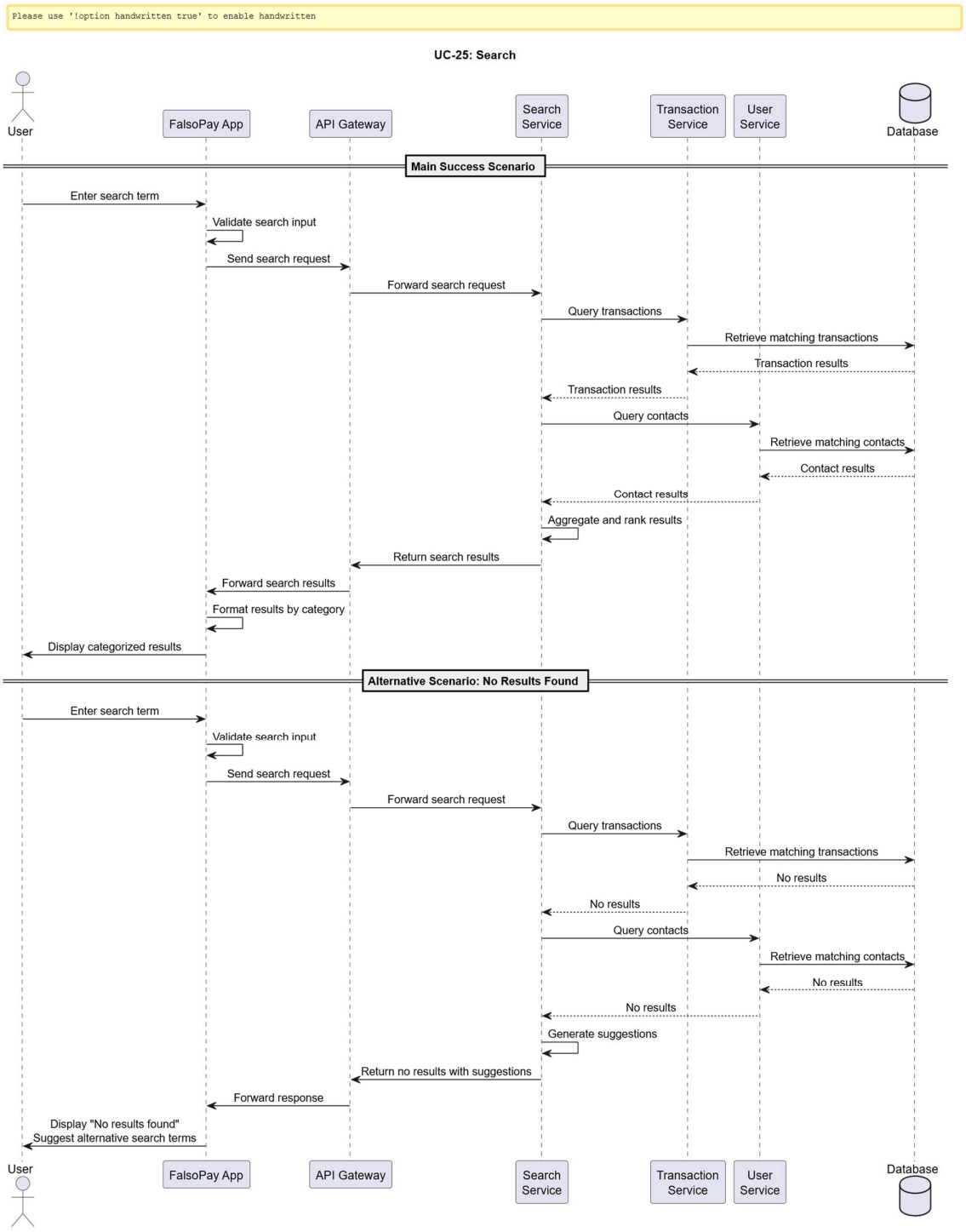


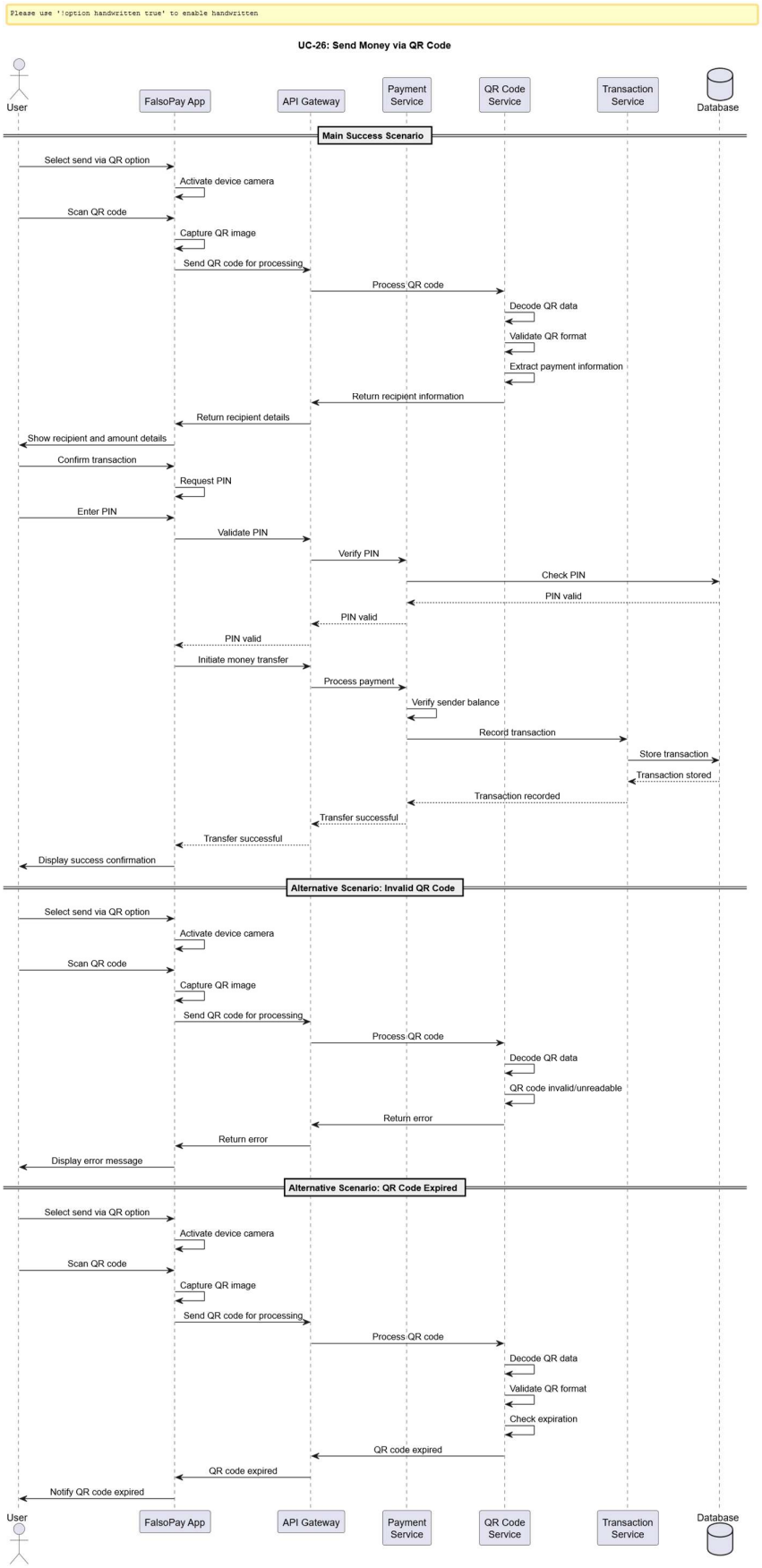

UC-22: Delete Account
Registered User
FalsoPay App
API Gateway
User Service
Transaction Service
Account Service
Notification Service
Database
Select account deletion option
Display warning about account deletion
Request account deletion
Request confirmation
Confirm deletion
DELETE /api/users/account
Delete account
Check pending transactions
Query pending transactions
Pending transactions
alt
[Pending transactions exist]
Pending transactions exist
409 Conflict
409 Conflict
Notify to resolve pending transactions first
[Account has balance]
No pending transactions
Check balance
Query account balance
Balance
Account has balance
409 Conflict
409 Conflict
Prompt to withdraw funds first
[Account can be deleted]
No pending transactions
Check balance
Query account balance
Zero balance
No balance
Mark account for deletion
Success
Send deletion confirmation
Notification queued
200 OK
200 OK
Display deletion confirmation
Registered User
FalsoPay App
API Gateway
User Service
Transaction Service
Account Service
Notification Service
Database

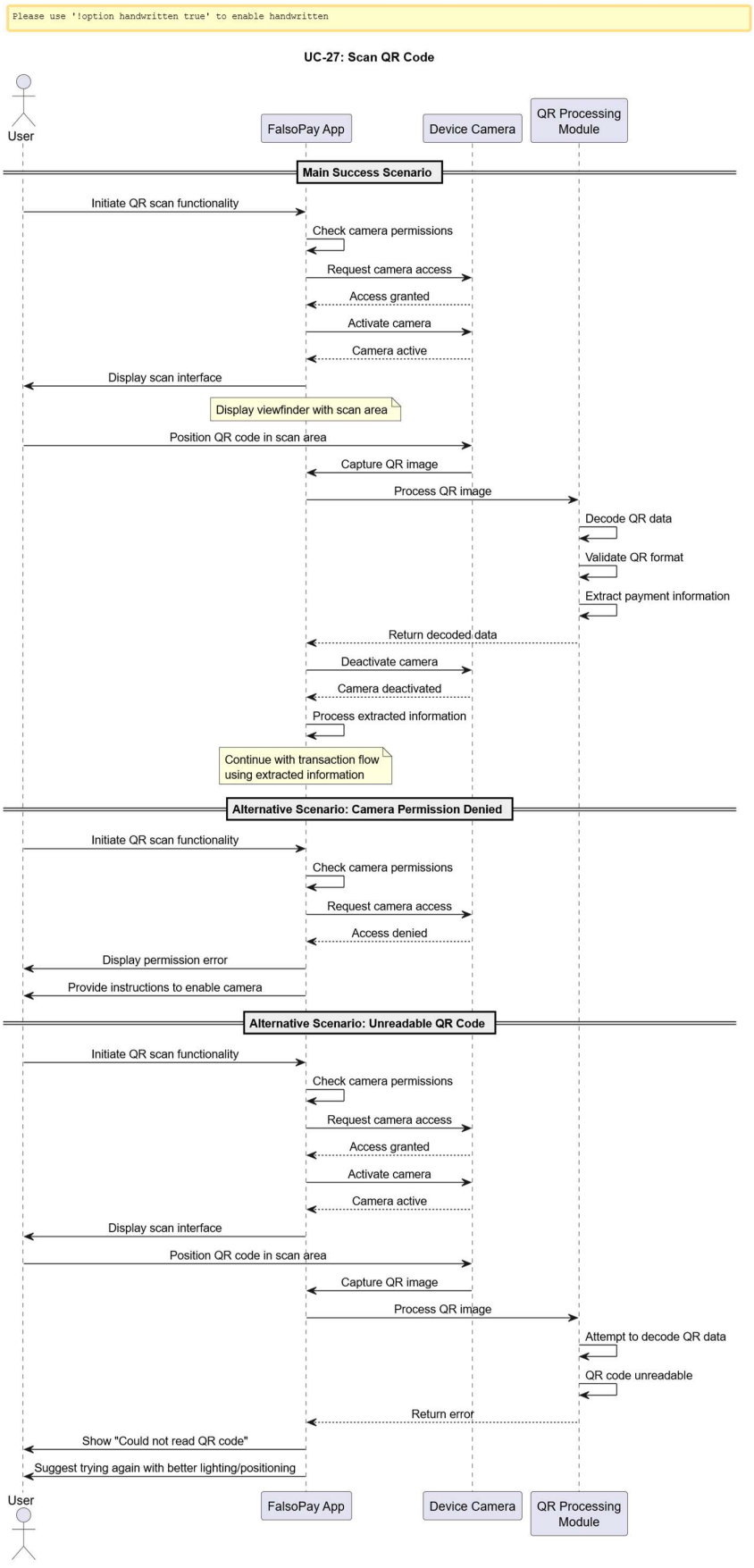
UC-23: PIN For Transaction

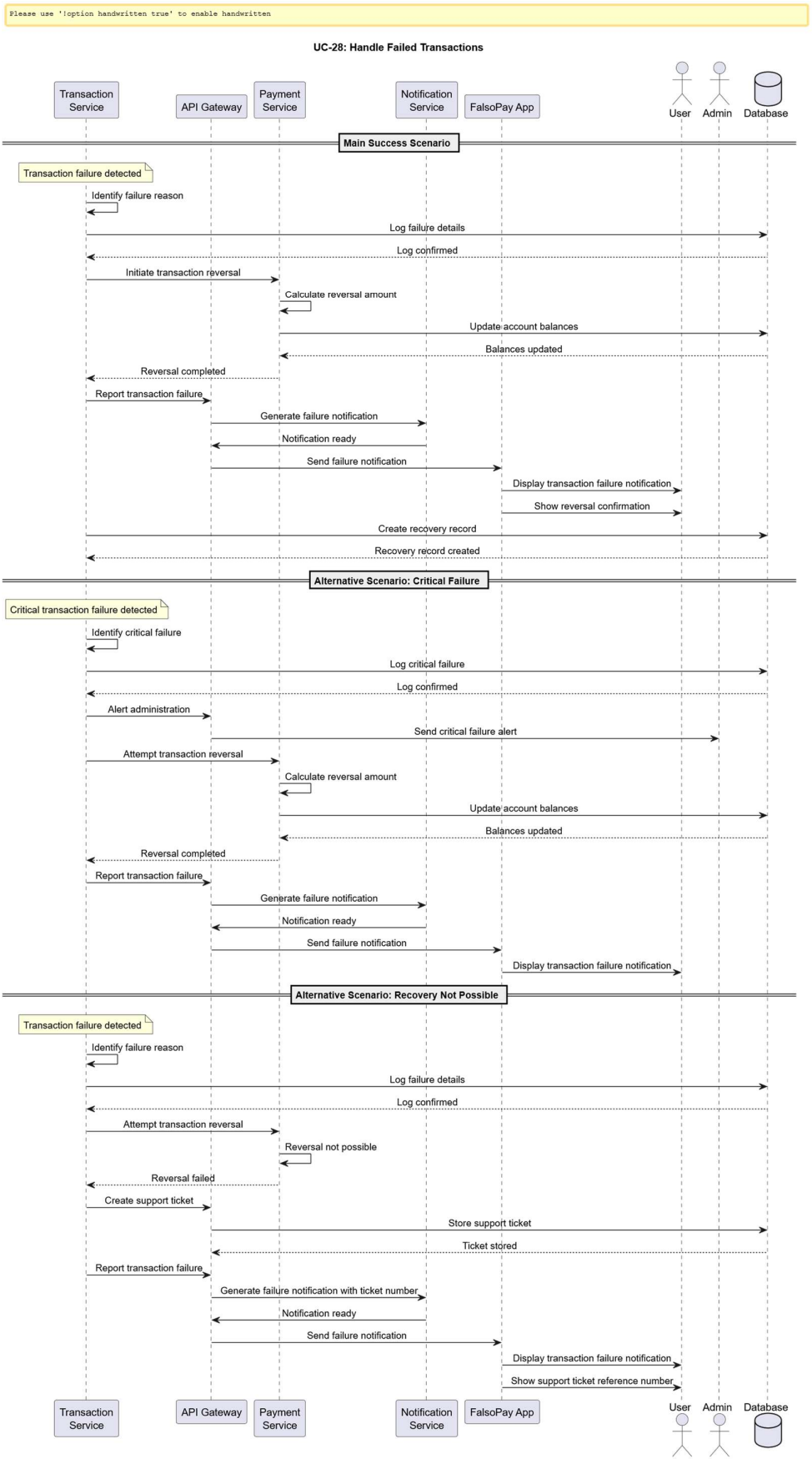


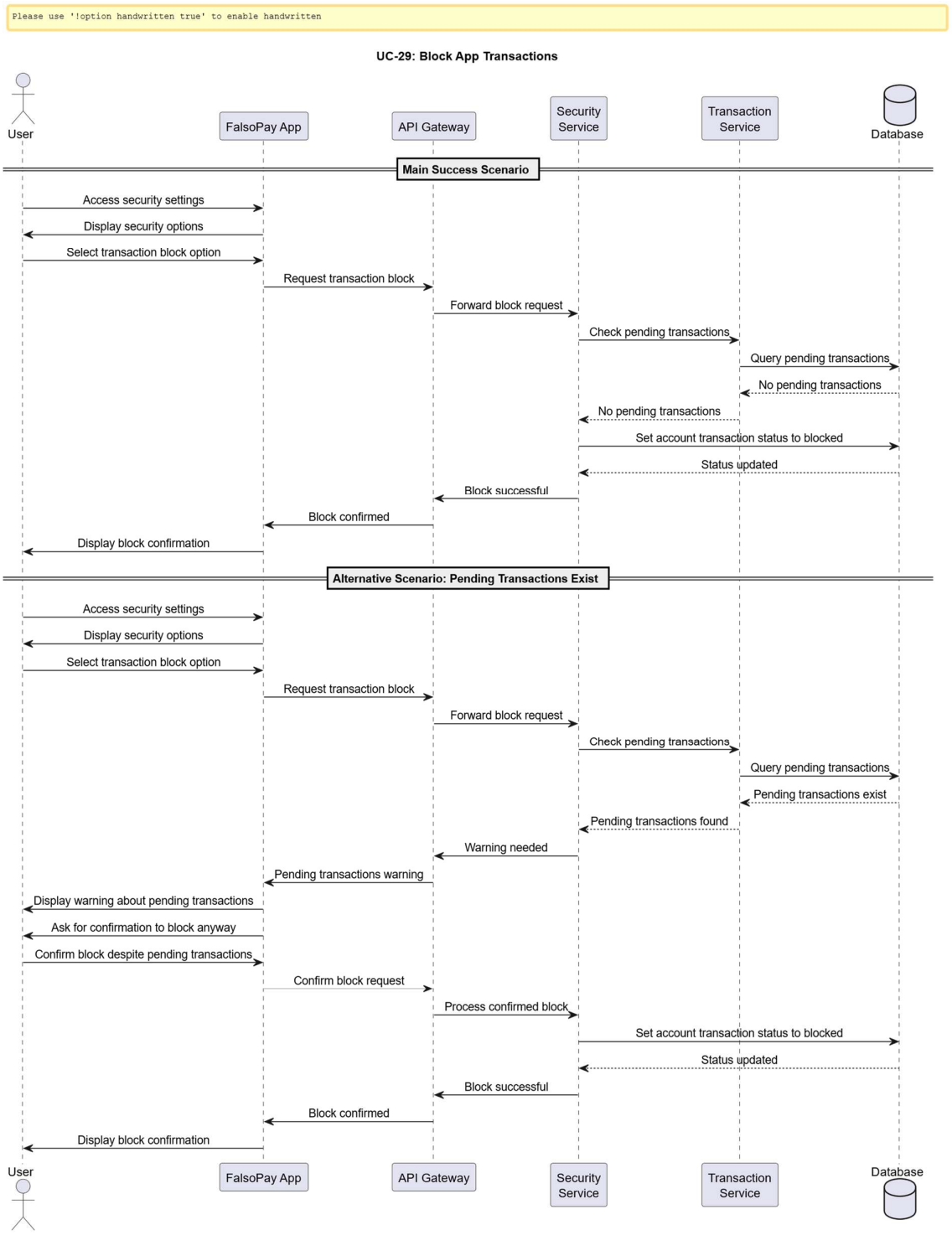

UC-24: Contact Customer Support
Registered User
FalsoPay App
API Gateway
Support Service
User Service
Notification Service
Database
Select contact support option
GET /api/support/categories
Get issue categories
Retrieve categories
Categories list
Issue categories
200 OK with categories
Display issue categories
Select issue category
Enter issue description
Submit support request
POST /api/support/tickets
Create support ticket
Get user details
Retrieve user info
User data
User details
alt
[System offers FAQ solutions]
Search FAQ for similar issues
Matching FAQs
200 OK with FAQ suggestions
200 OK with FAQ suggestions
Display FAQ solutions first
Choose to continue with ticket
POST /api/support/tickets/confirm
Confirm ticket creation
[Direct ticket creation]
Determine ticket priority
Create support ticket
Ticket created with reference number
alt
[Critical issue]
Flag for priority handling
Send ticket confirmation
Notification queued
201 Created with ticket reference
201 Created with ticket reference
Display confirmation with reference number
Registered User
FalsoPay App
API Gateway
Support Service
User Service
Notification Service
Database

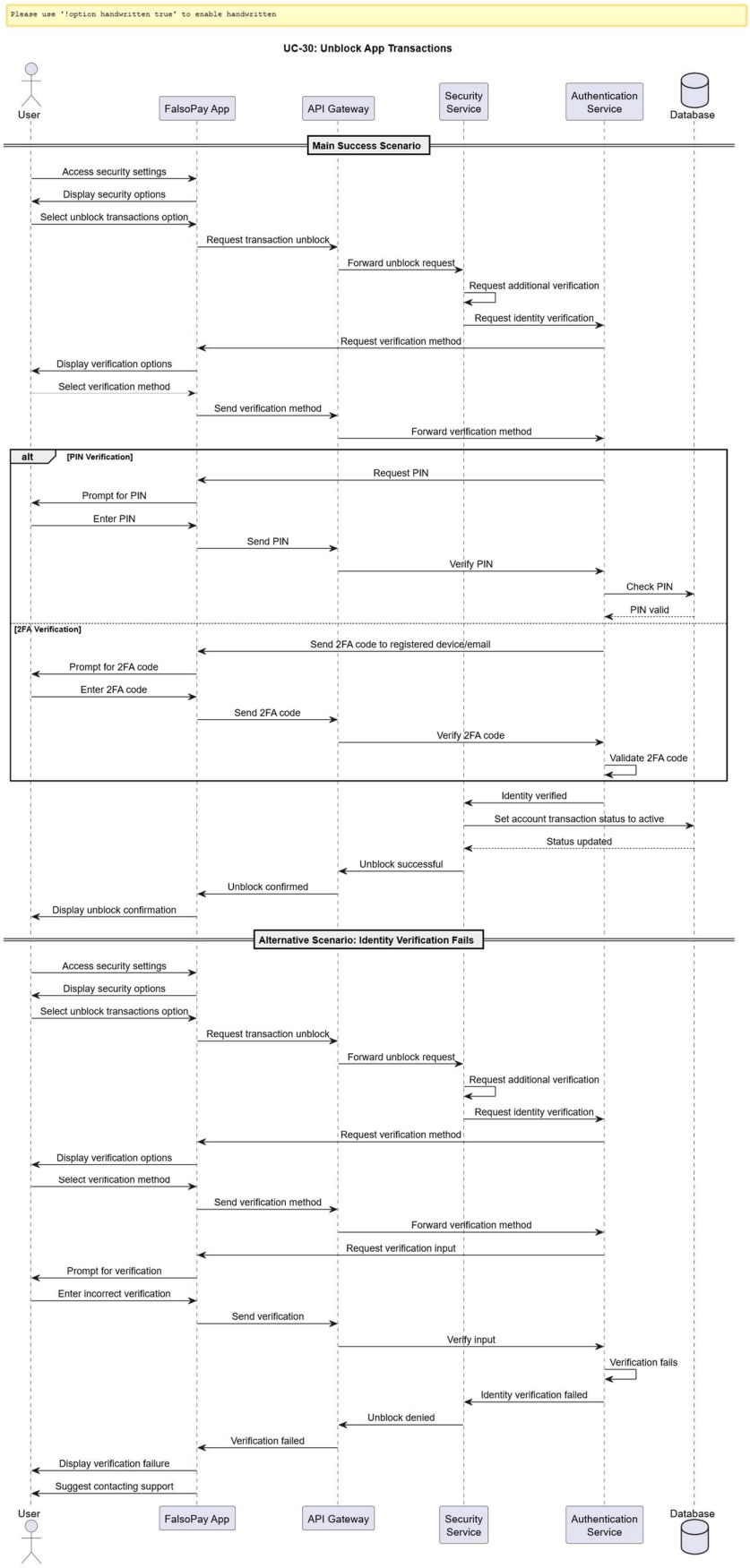

Please use '!option handwritten true' to enable handwritten
UC-25: Search
User
FalsoPay App
API Gateway
Search Service
Transaction Service
User Service
Database
Main Success Scenario
Enter search term
Validate search input
Send search request
Forward search request
Query transactions
Retrieve matching transactions
Transaction results
Transaction results
Query contacts
Retrieve matching contacts
Contact results
Contact results
Aggregate and rank results
Return search results
Forward search results
Format results by category
Display categorized results
Alternative Scenario: No Results Found
Enter search term
Validate search input
Send search request
Forward search request
Query transactions
Retrieve matching transactions
No results
No results
Query contacts
Retrieve matching contacts
No results
No results
Generate suggestions
Return no results with suggestions
Forward response
Display "No results found"
Suggest alternative search terms
User
FalsoPay App
API Gateway
Search Service
Transaction Service
User Service
Database

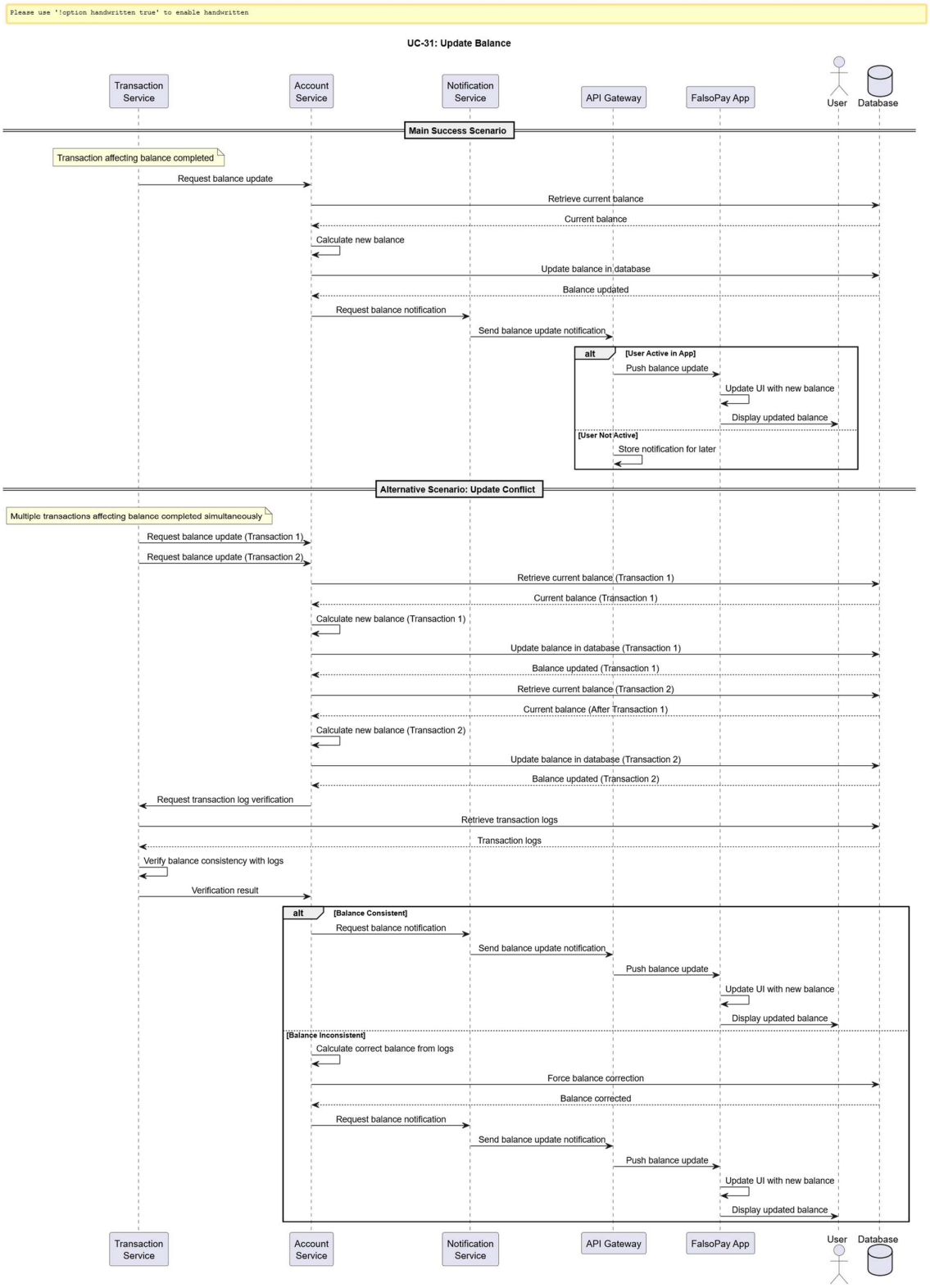

Please use '!option handwritten true' to enable handwritten
UC-26: Send Money via QR Code
User
FalsoPay App
API Gateway
Payment Service
QR Code Service
Transaction Service
Database
Main Success Scenario
Select send via QR option
Activate device camera
Scan QR code
Capture QR image
Send QR code for processing
Process QR code
Decode QR data
Validate QR format
Extract payment information
Return recipient information
Return recipient details
Show recipient and amount details
Confirm transaction
Request PIN
Enter PIN
Validate PIN
Verify PIN
Check PIN
PIN valid
PIN valid
PIN valid
Initiate money transfer
Process payment
Verify sender balance
Record transaction
Store transaction
Transaction stored
Transaction recorded
Transfer successful
Transfer successful
Display success confirmation
Alternative Scenario: Invalid QR Code
Select send via QR option
Activate device camera
Scan QR code
Capture QR image
Send QR code for processing
Process QR code
Decode QR data
QR code invalid/unreadable
Return error
Return error
Display error message
Alternative Scenario: QR Code Expired
Select send via QR option
Activate device camera
Scan QR code
Capture QR image
Send QR code for processing
Process QR code
Decode QR data
Validate QR format
Check expiration
QR code expired
QR code expired
Notify QR code expired
User
FalsoPay App
API Gateway
Payment Service
QR Code Service
Transaction Service
Database

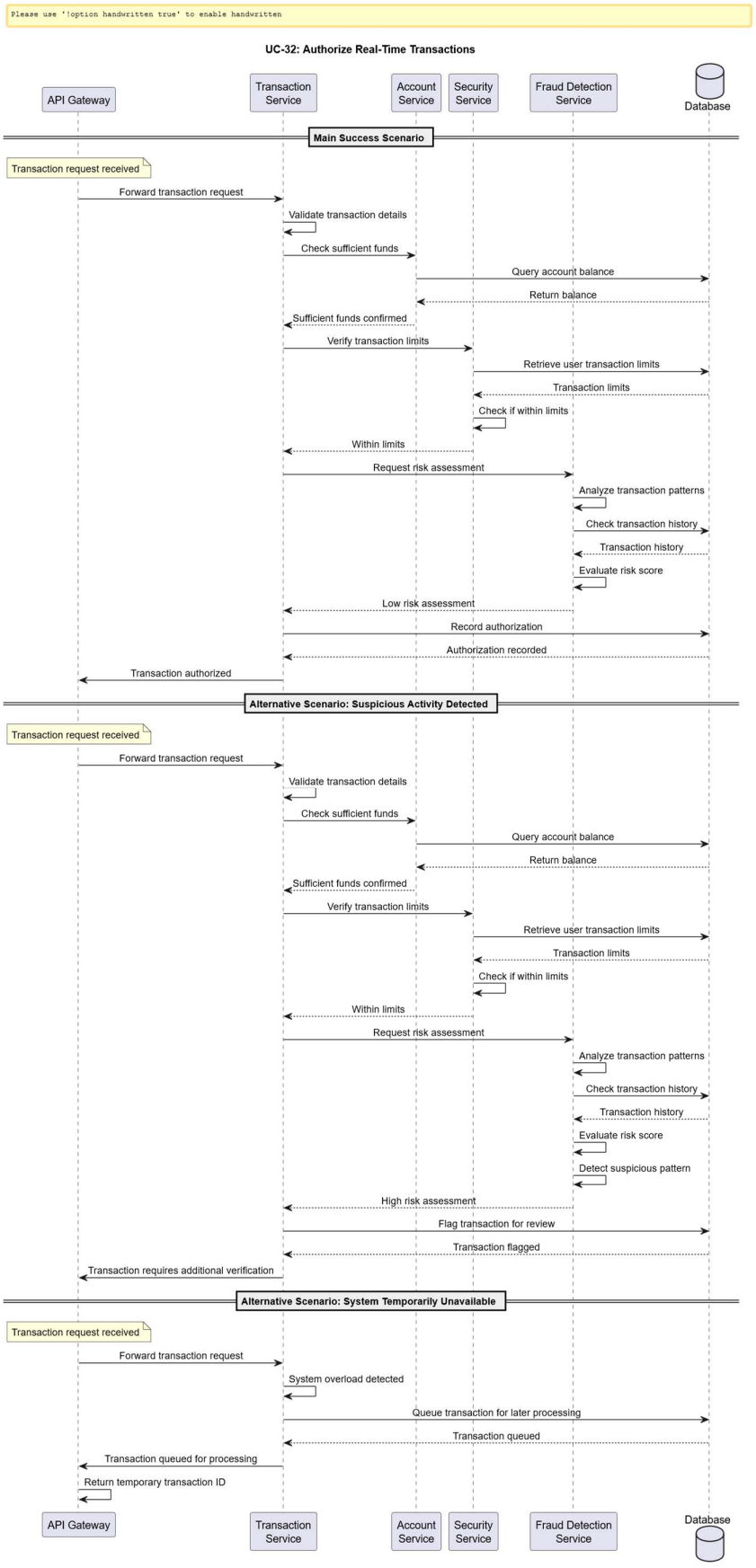

Please use '!option handwritten true' to enable handwritten
UC-27: Scan QR Code
User
FalsoPay App
Device Camera
QR Processing Module
Main Success Scenario
Initiate QR scan functionality
Check camera permissions
Request camera access
Access granted
Activate camera
Camera active
Display scan interface
Display viewfinder with scan area
Position QR code in scan area
Capture QR image
Process QR image
Decode QR data
Validate QR format
Extract payment information
Return decoded data
Deactivate camera
Camera deactivated
Process extracted information
Continue with transaction flow using extracted information
Alternative Scenario: Camera Permission Denied
Initiate QR scan functionality
Check camera permissions
Request camera access
Access denied
Display permission error
Provide instructions to enable camera
Alternative Scenario: Unreadable QR Code
Initiate QR scan functionality
Check camera permissions
Request camera access
Access granted
Activate camera
Camera active
Display scan interface
Position QR code in scan area
Capture QR image
Process QR image
Attempt to decode QR data
QR code unreadable
Return error
Show "Could not read QR code"
Suggest trying again with better lighting/positioning
User
FalsoPay App
Device Camera
QR Processing Module

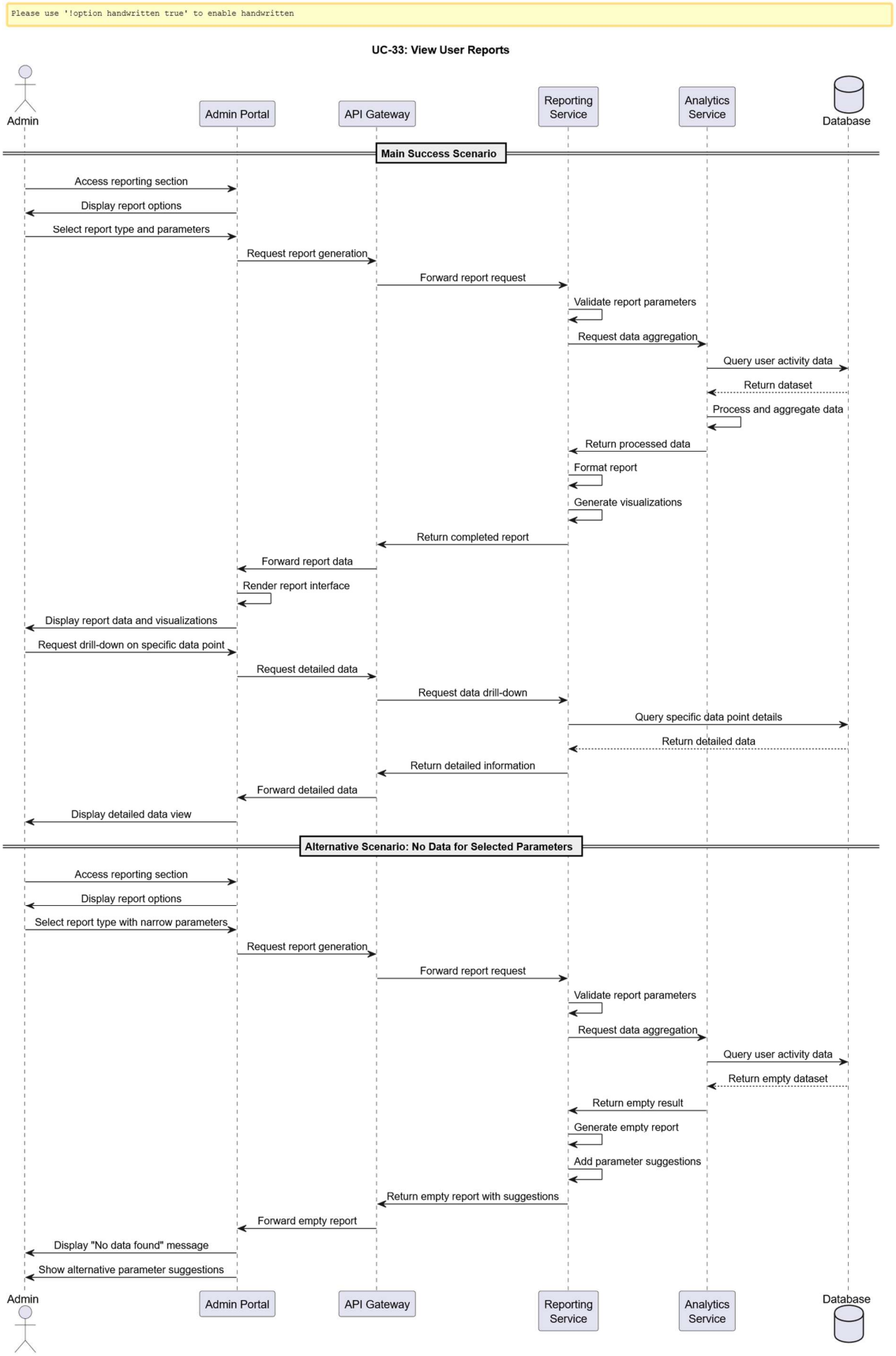

Please use '!option handwritten true' to enable handwritten
UC-28: Handle Failed Transactions
Transaction Service
API Gateway
Payment Service
Notification Service
FalsoPay App
User
Admin
Database
Main Success Scenario
Transaction failure detected
Identify failure reason
Log failure details
Log confirmed
Initiate transaction reversal
Calculate reversal amount
Update account balances
Balances updated
Reversal completed
Report transaction failure
Generate failure notification
Notification ready
Send failure notification
Display transaction failure notification
Show reversal confirmation
Create recovery record
Recovery record created
Alternative Scenario: Critical Failure
Critical transaction failure detected
Identify critical failure
Log critical failure
Log confirmed
Alert administration
Send critical failure alert
Attempt transaction reversal
Calculate reversal amount
Update account balances
Balances updated
Reversal completed
Report transaction failure
Generate failure notification
Notification ready
Send failure notification
Display transaction failure notification
Alternative Scenario: Recovery Not Possible
Transaction failure detected
Identify failure reason
Log failure details
Log confirmed
Attempt transaction reversal
Reversal not possible
Reversal failed
Create support ticket
Store support ticket
Ticket stored
Report transaction failure
Generate failure notification with ticket number
Notification ready
Send failure notification
Display transaction failure notification
Show support ticket reference number
Transaction Service
API Gateway
Payment Service
Notification Service
FalsoPay App
User
Admin
Database

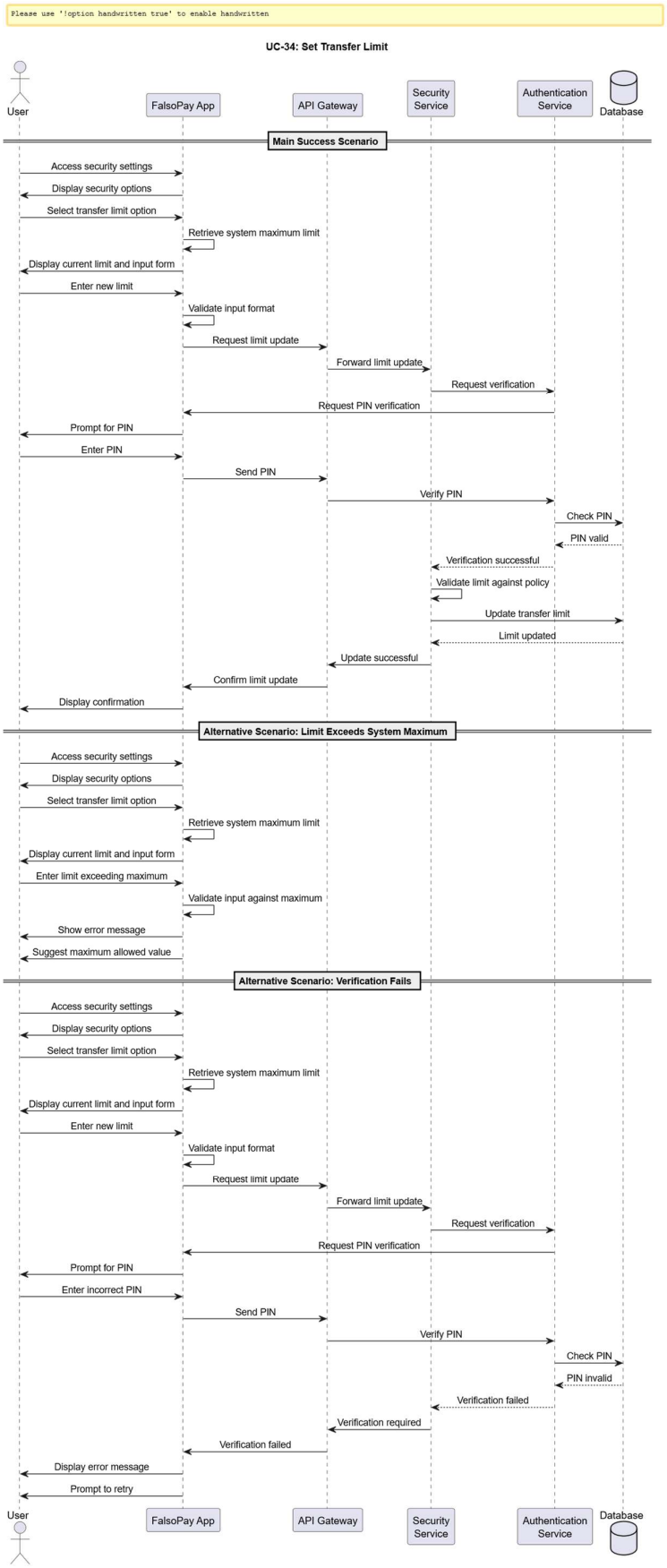

Please use '!option handwritten true' to enable handwritten
UC-29: Block App Transactions
User
FalsoPay App
API Gateway
Security Service
Transaction Service
Database
Main Success Scenario
Access security settings
Display security options
Select transaction block option
Request transaction block
Forward block request
Check pending transactions
Query pending transactions
No pending transactions
No pending transactions
Set account transaction status to blocked
Status updated
Block successful
Block confirmed
Display block confirmation
Alternative Scenario: Pending Transactions Exist
Access security settings
Display security options
Select transaction block option
Request transaction block
Forward block request
Check pending transactions
Query pending transactions
Pending transactions exist
Pending transactions found
Warning needed
Pending transactions warning
Display warning about pending transactions
Ask for confirmation to block anyway
Confirm block despite pending transactions
Confirm block request
Process confirmed block
Set account transaction status to blocked
Status updated
Block successful
Block confirmed
Display block confirmation
User
FalsoPay App
API Gateway
Security Service
Transaction Service
Database

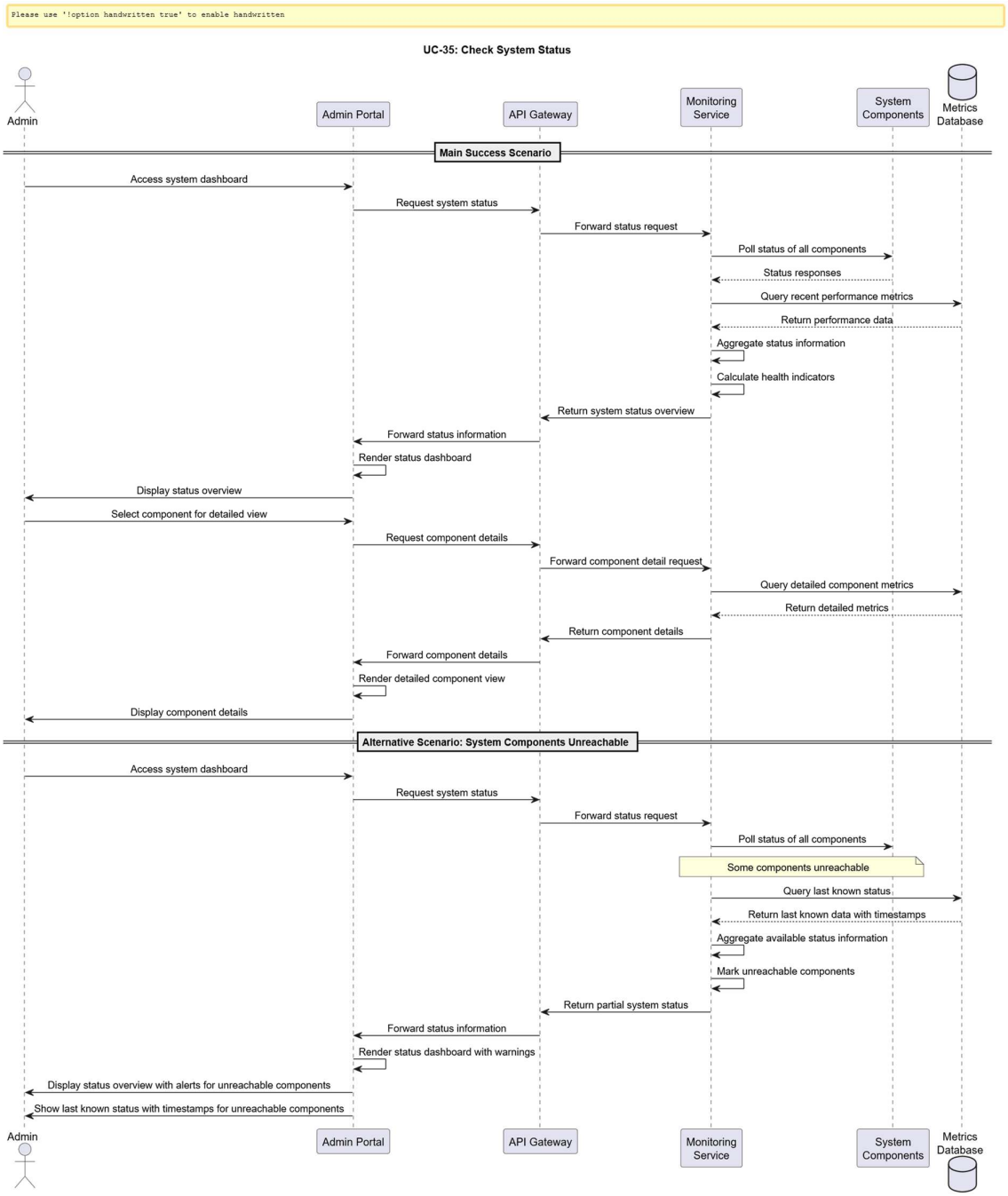

Please use '!option handwritten true' to enable handwritten
UC-30: Unblock App Transactions
User
FalsoPay App
API Gateway
Security Service
Authentication Service
Database
Main Success Scenario
Access security settings
Display security options
Select unblock transactions option
Request transaction unblock
Forward unblock request
Request additional verification
Request identity verification
Request verification method
Display verification options
Select verification method
Send verification method
Forward verification method
alt
[PIN Verification]
Request PIN
Prompt for PIN
Enter PIN
Send PIN
Verify PIN
Check PIN
PIN valid
[2FA Verification]
Send 2FA code to registered device/email
Prompt for 2FA code
Enter 2FA code
Send 2FA code
Verify 2FA code
Validate 2FA code
Identity verified
Set account transaction status to active
Status updated
Unblock successful
Unblock confirmed
Display unblock confirmation
Alternative Scenario: Identity Verification Fails
Access security settings
Display security options
Select unblock transactions option
Request transaction unblock
Forward unblock request
Request additional verification
Request identity verification
Request verification method
Display verification options
Select verification method
Send verification method
Forward verification method
Request verification input
Prompt for verification
Enter incorrect verification
Send verification
Verify input
Verification fails
Identity verification failed
Unblock denied
Verification failed
Display verification failure
Suggest contacting support
User
FalsoPay App
API Gateway
Security Service
Authentication Service
Database

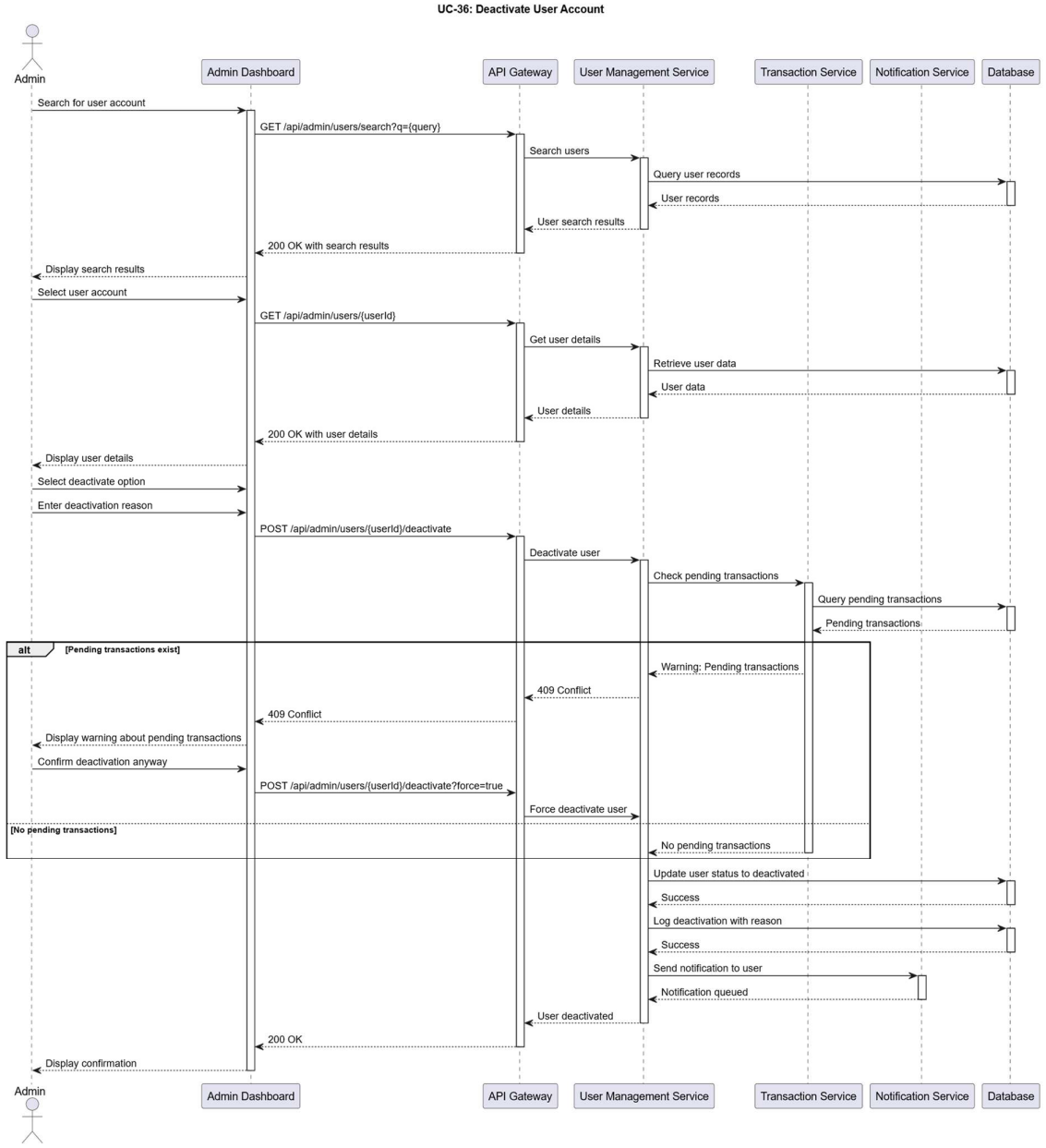

Please use '!option handwritten true' to enable handwritten
UC-31: Update Balance
Transaction Service
Account Service
Notification Service
API Gateway
FalsoPay App
User
Database
Main Success Scenario
Transaction affecting balance completed
Request balance update
Retrieve current balance
Current balance
Calculate new balance
Update balance in database
Balance updated
Request balance notification
Send balance update notification
alt
[User Active in App]
Push balance update
Update UI with new balance
Display updated balance
[User Not Active]
Store notification for later
Alternative Scenario: Update Conflict
Multiple transactions affecting balance completed simultaneously
Request balance update (Transaction 1)
Request balance update (Transaction 2)
Retrieve current balance (Transaction 1)
Current balance (Transaction 1)
Calculate new balance (Transaction 1)
Update balance in database (Transaction 1)
Balance updated (Transaction 1)
Retrieve current balance (Transaction 2)
Current balance (After Transaction 1)
Calculate new balance (Transaction 2)
Update balance in database (Transaction 2)
Balance updated (Transaction 2)
Request transaction log verification
Retrieve transaction logs
Transaction logs
Verify balance consistency with logs
Verification result
alt
[Balance Consistent]
Request balance notification
Send balance update notification
Push balance update
Update UI with new balance
Display updated balance
[Balance Inconsistent]
Calculate correct balance from logs
Force balance correction
Balance corrected
Request balance notification
Send balance update notification
Push balance update
Update UI with new balance
Display updated balance
Transaction Service
Account Service
Notification Service
API Gateway
FalsoPay App
User
Database

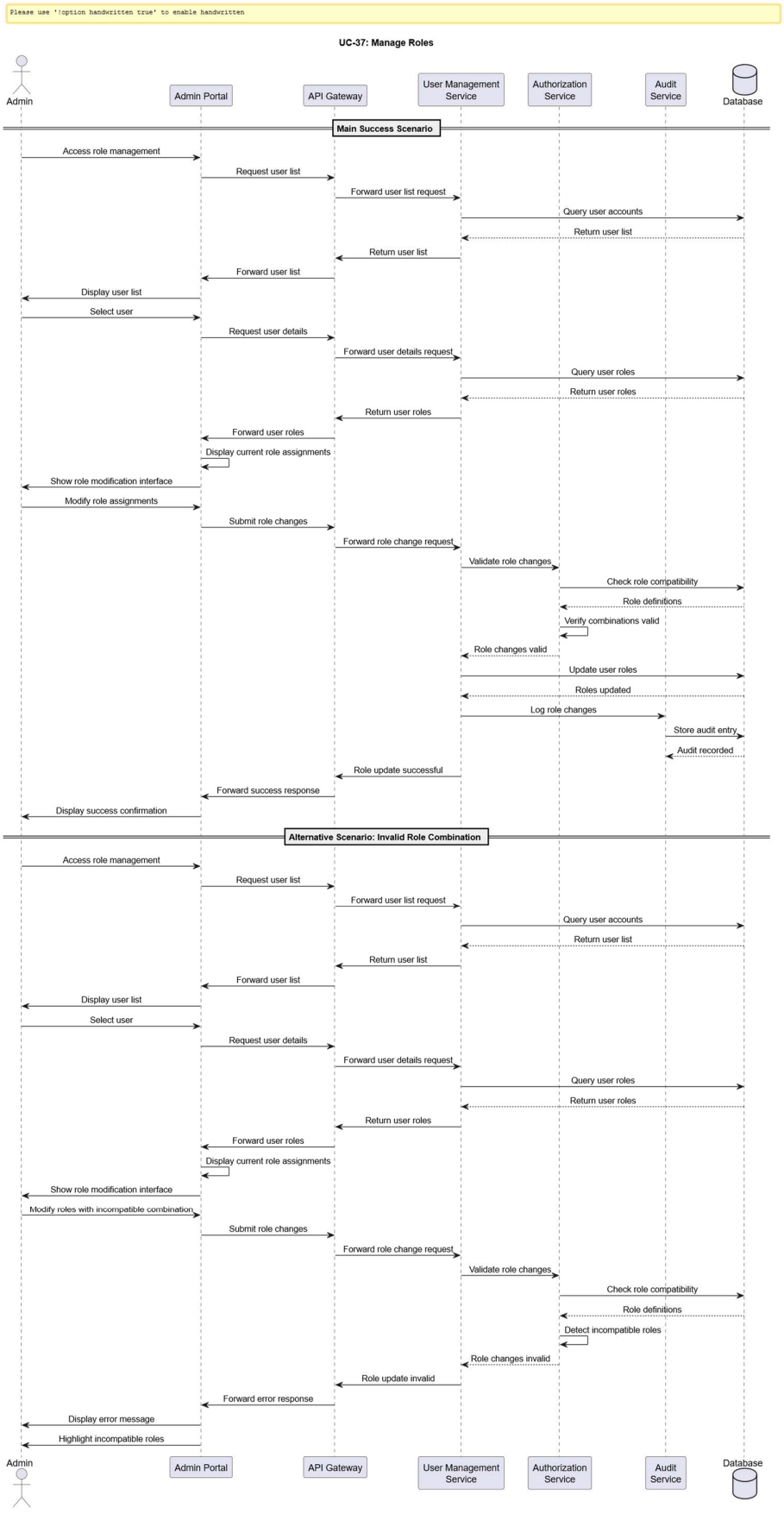

Please use '!option handwritten true' to enable handwritten
UC-32: Authorize Real-Time Transactions
API Gateway
Transaction Service
Account Service
Security Service
Fraud Detection Service
Database
Main Success Scenario
Transaction request received
Forward transaction request
Validate transaction details
Check sufficient funds
Query account balance
Return balance
Sufficient funds confirmed
Verify transaction limits
Retrieve user transaction limits
Transaction limits
Check if within limits
Within limits
Request risk assessment
Analyze transaction patterns
Check transaction history
Transaction history
Evaluate risk score
Low risk assessment
Record authorization
Authorization recorded
Transaction authorized
Alternative Scenario: Suspicious Activity Detected
Transaction request received
Forward transaction request
Validate transaction details
Check sufficient funds
Query account balance
Return balance
Sufficient funds confirmed
Verify transaction limits
Retrieve user transaction limits
Transaction limits
Check if within limits
Within limits
Request risk assessment
Analyze transaction patterns
Check transaction history
Transaction history
Evaluate risk score
Detect suspicious pattern
High risk assessment
Flag transaction for review
Transaction flagged
Transaction requires additional verification
Alternative Scenario: System Temporarily Unavailable
Transaction request received
Forward transaction request
System overload detected
Queue transaction for later processing
Transaction queued
Transaction queued for processing
Return temporary transaction ID
API Gateway
Transaction Service
Account Service
Security Service
Fraud Detection Service
Database

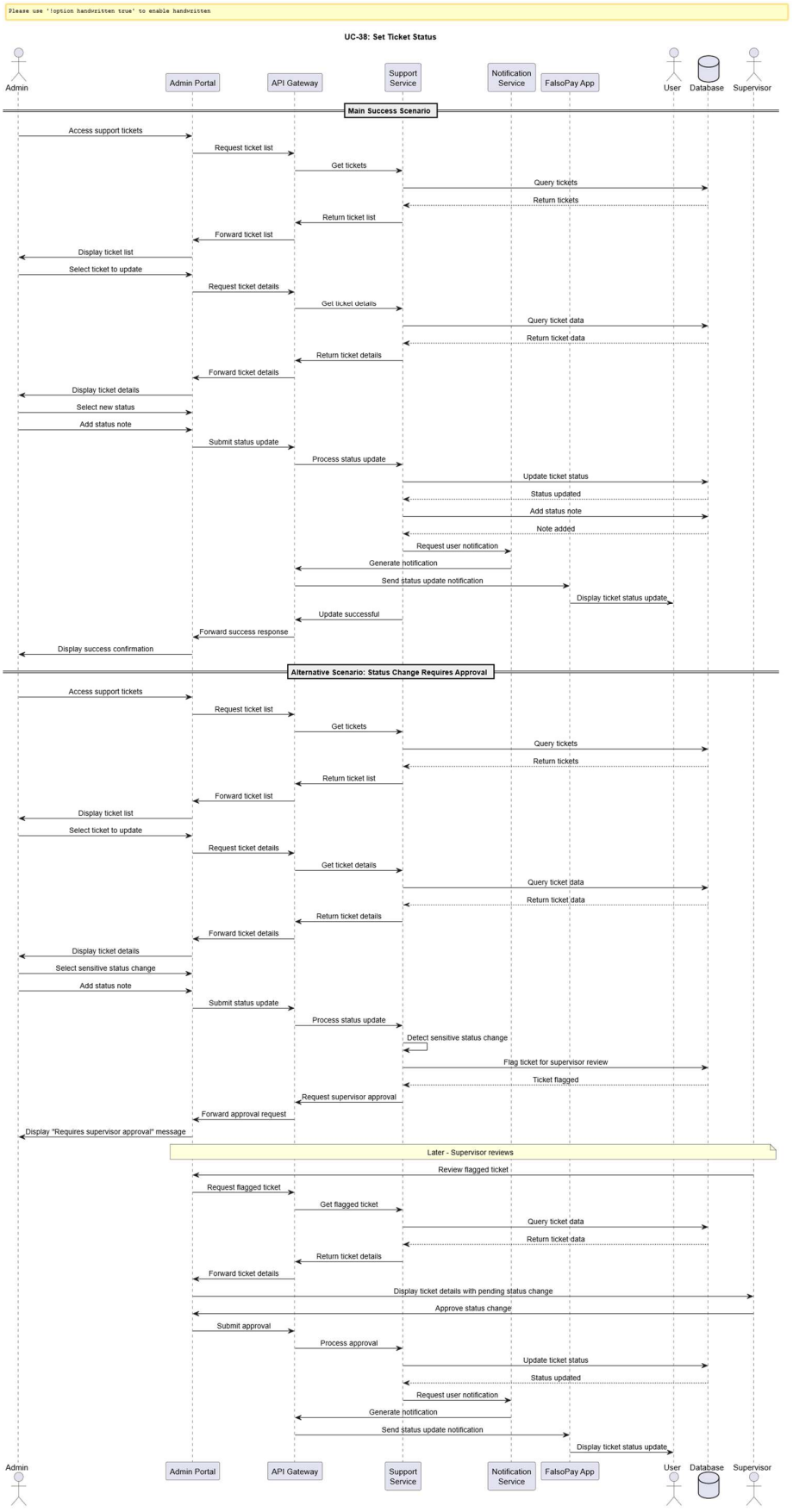

Please use '!option handwritten true' to enable handwritten
UC-33: View User Reports
Admin
Admin Portal
API Gateway
Reporting Service
Analytics Service
Database
Main Success Scenario
Access reporting section
Display report options
Select report type and parameters
Request report generation
Forward report request
Validate report parameters
Request data aggregation
Query user activity data
Return dataset
Process and aggregate data
Return processed data
Format report
Generate visualizations
Return completed report
Forward report data
Render report interface
Display report data and visualizations
Request drill-down on specific data point
Request detailed data
Request data drill-down
Query specific data point details
Return detailed data
Return detailed information
Forward detailed data
Display detailed data view
Alternative Scenario: No Data for Selected Parameters
Access reporting section
Display report options
Select report type with narrow parameters
Request report generation
Forward report request
Validate report parameters
Request data aggregation
Query user activity data
Return empty dataset
Return empty result
Generate empty report
Add parameter suggestions
Return empty report with suggestions
Forward empty report
Display "No data found" message
Show alternative parameter suggestions
Admin
Admin Portal
API Gateway
Reporting Service
Analytics Service
Database

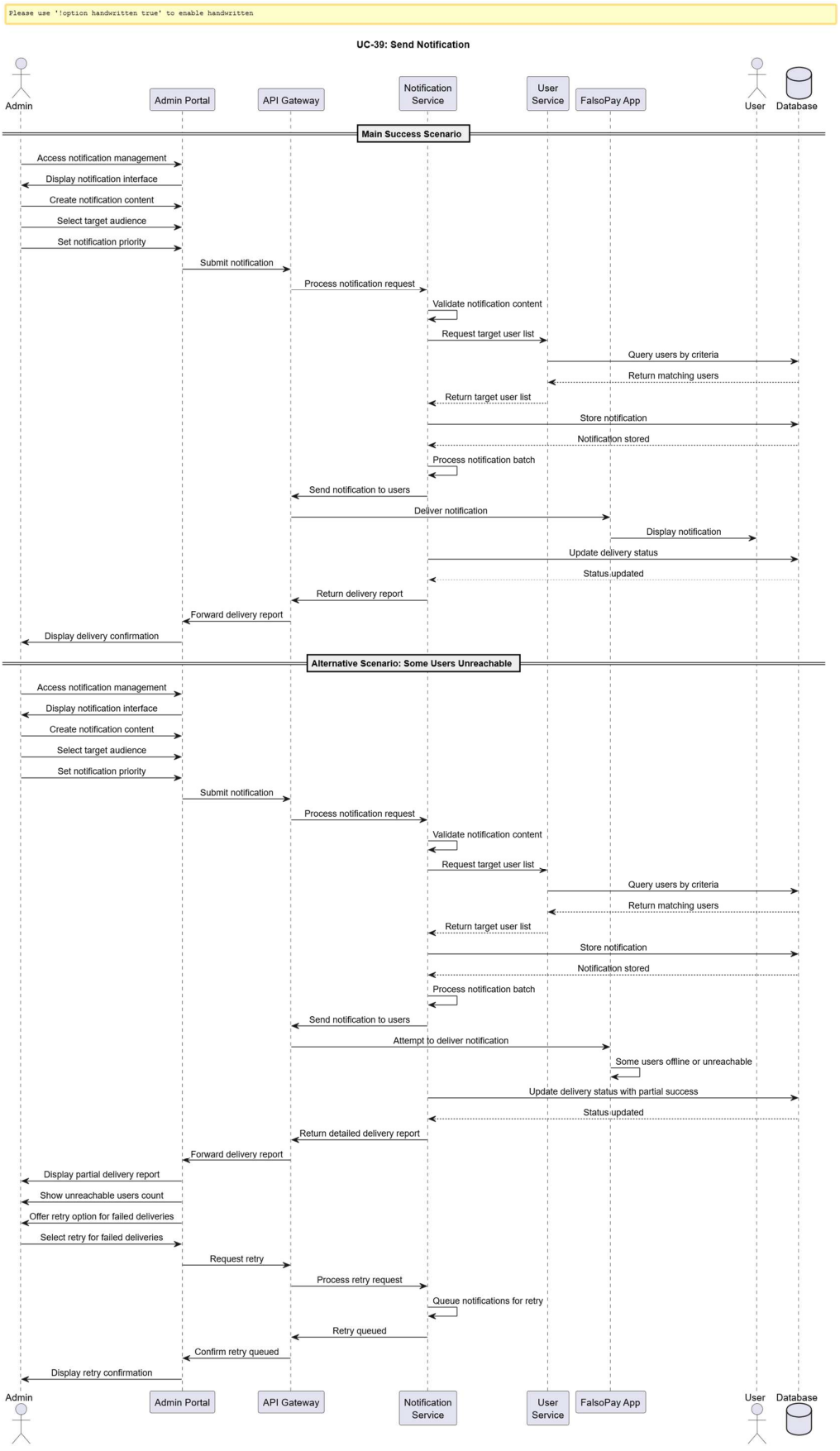

Please use '!option handwritten true' to enable handwritten
UC-34: Set Transfer Limit
User
FalsoPay App
API Gateway
Security Service
Authentication Service
Database
Main Success Scenario
Access security settings
Display security options
Select transfer limit option
Retrieve system maximum limit
Display current limit and input form
Enter new limit
Validate input format
Request limit update
Forward limit update
Request verification
Request PIN verification
Prompt for PIN
Enter PIN
Send PIN
Verify PIN
Check PIN
PIN valid
Verification successful
Validate limit against policy
Update transfer limit
Limit updated
Update successful
Confirm limit update
Display confirmation
Alternative Scenario: Limit Exceeds System Maximum
Access security settings
Display security options
Select transfer limit option
Retrieve system maximum limit
Display current limit and input form
Enter limit exceeding maximum
Validate input against maximum
Show error message
Suggest maximum allowed value
Alternative Scenario: Verification Fails
Access security settings
Display security options
Select transfer limit option
Retrieve system maximum limit
Display current limit and input form
Enter new limit
Validate input format
Request limit update
Forward limit update
Request verification
Request PIN verification
Prompt for PIN
Enter incorrect PIN
Send PIN
Verify PIN
Check PIN
PIN invalid
Verification failed
Verification required
Verification failed
Display error message
Prompt to retry
User
FalsoPay App
API Gateway
Security Service
Authentication Service
Database

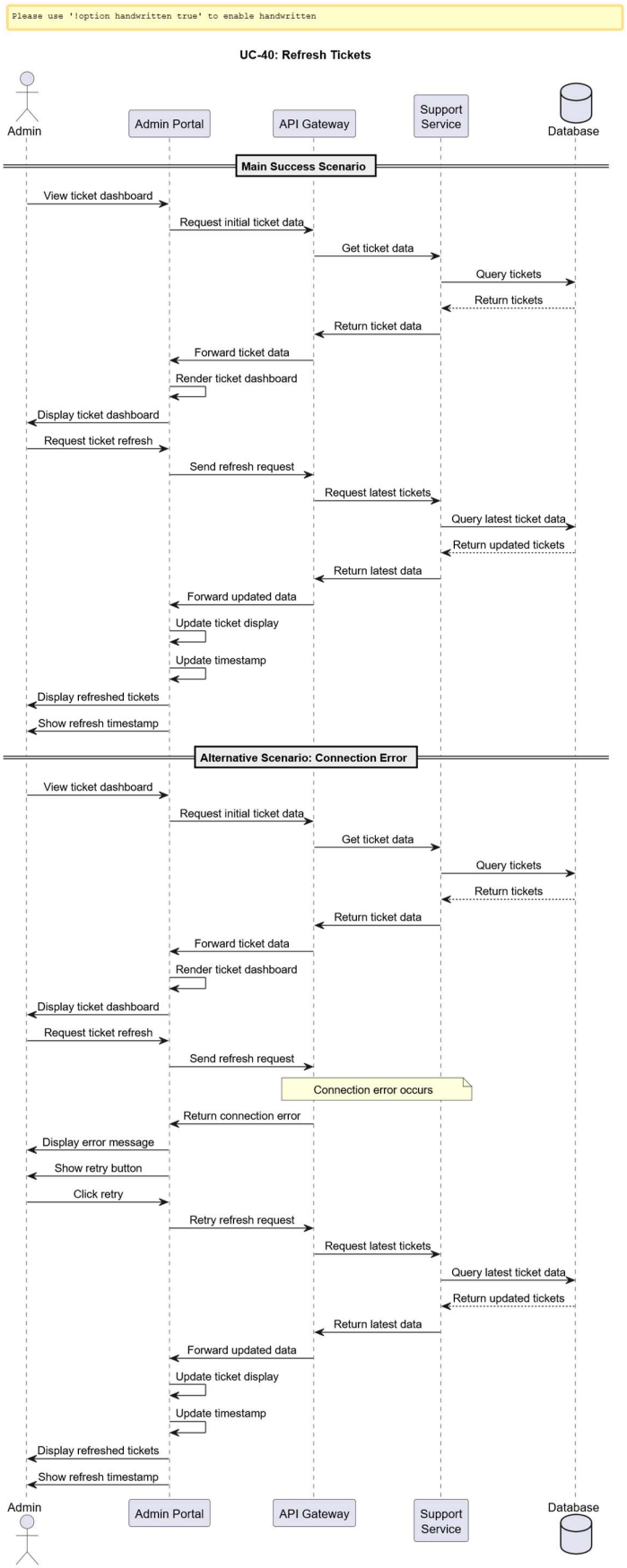

Please use '!option handwritten true' to enable handwritten
UC-35: Check System Status
Admin
Admin Portal
API Gateway
Monitoring Service
System Components
Metrics Database
Main Success Scenario
Access system dashboard
Request system status
Forward status request
Poll status of all components
Status responses
Query recent performance metrics
Return performance data
Aggregate status information
Calculate health indicators
Return system status overview
Forward status information
Render status dashboard
Display status overview
Select component for detailed view
Request component details
Forward component detail request
Query detailed component metrics
Return detailed metrics
Return component details
Forward component details
Render detailed component view
Display component details
Alternative Scenario: System Components Unreachable
Access system dashboard
Request system status
Forward status request
Poll status of all components
Some components unreachable
Query last known status
Return last known data with timestamps
Aggregate available status information
Mark unreachable components
Return partial system status
Forward status information
Render status dashboard with warnings
Display status overview with alerts for unreachable components
Show last known status with timestamps for unreachable components
Admin
Admin Portal
API Gateway
Monitoring Service
System Components
Metrics Database

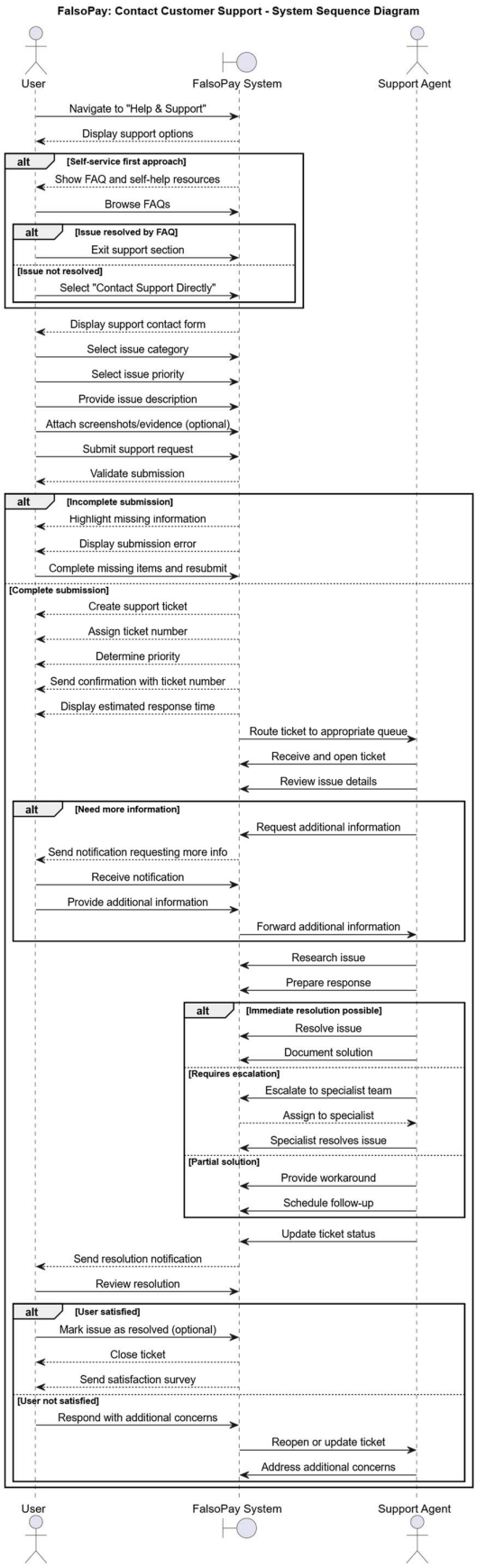

UC-36: Deactivate User Account
Admin
Admin Dashboard
API Gateway
User Management Service
Transaction Service
Notification Service
Database
Search for user account
GET /api/admin/users/search?q={query}
Search users
Query user records
User records
User search results
200 OK with search results
Display search results
Select user account
GET /api/admin/users/{userId}
Get user details
Retrieve user data
User data
User details
200 OK with user details
Display user details
Select deactivate option
Enter deactivation reason
POST /api/admin/users/{userId}/deactivate
Deactivate user
Check pending transactions
Query pending transactions
Pending transactions
alt
[Pending transactions exist]
Warning: Pending transactions
409 Conflict
409 Conflict
Display warning about pending transactions
Confirm deactivation anyway
POST /api/admin/users/{userId}/deactivate?force=true
Force deactivate user
[No pending transactions]
No pending transactions
Update user status to deactivated
Success
Log deactivation with reason
Success
Send notification to user
Notification queued
User deactivated
200 OK
Display confirmation
Admin
Admin Dashboard
API Gateway
User Management Service
Transaction Service
Notification Service
Database

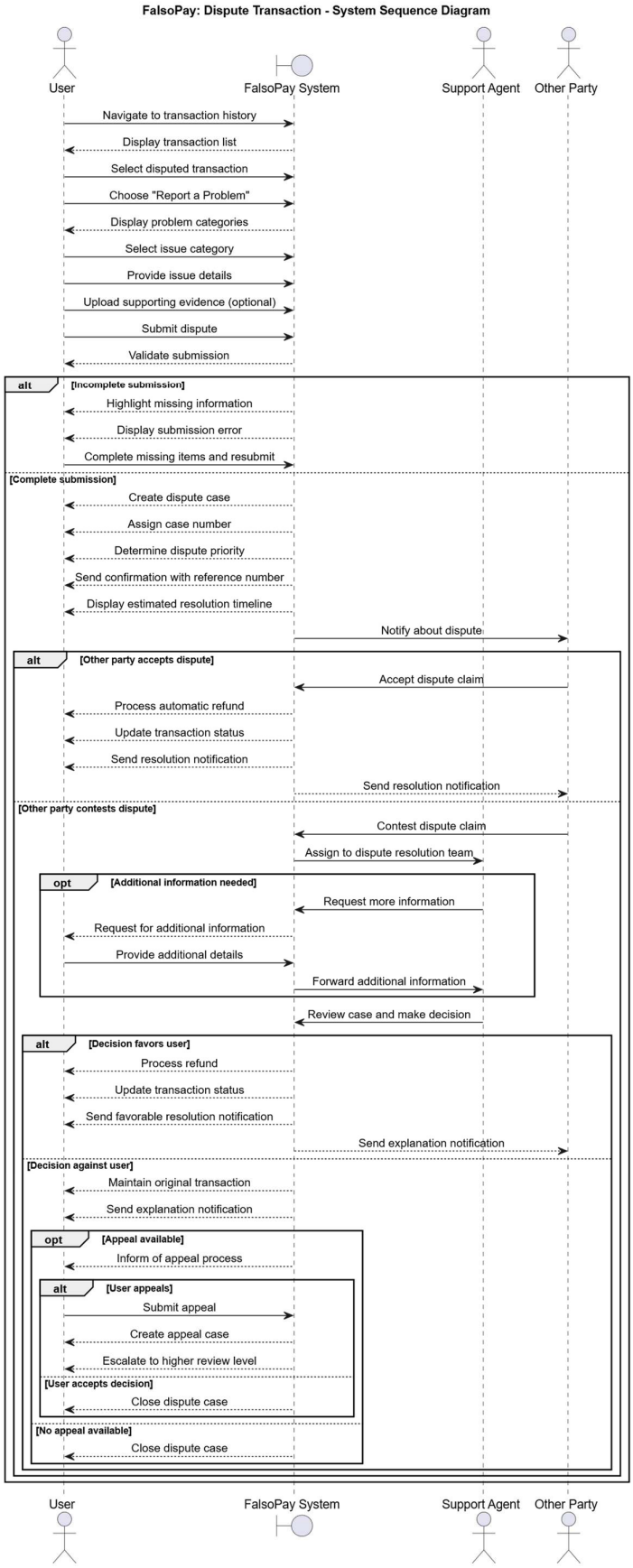

Please use '!option handwritten true' to enable handwritten
UC-37: Manage Roles
Admin
Admin Portal
API Gateway
User Management Service
Authorization Service
Audit Service
Database
Main Success Scenario
Access role management
Request user list
Forward user list request
Query user accounts
Return user list
Return user list
Forward user list
Display user list
Select user
Request user details
Forward user details request
Query user roles
Return user roles
Return user roles
Forward user roles
Display current role assignments
Show role modification interface
Modify role assignments
Submit role changes
Forward role change request
Validate role changes
Check role compatibility
Role definitions
Verify combinations valid
Role changes valid
Update user roles
Roles updated
Log role changes
Store audit entry
Audit recorded
Role update successful
Forward success response
Display success confirmation
Alternative Scenario: Invalid Role Combination
Access role management
Request user list
Forward user list request
Query user accounts
Return user list
Return user list
Forward user list
Display user list
Select user
Request user details
Forward user details request
Query user roles
Return user roles
Return user roles
Forward user roles
Display current role assignments
Show role modification interface
Modify roles with incompatible combination
Submit role changes
Forward role change request
Validate role changes
Check role compatibility
Role definitions
Detect incompatible roles
Role changes invalid
Role update invalid
Forward error response
Display error message
Highlight incompatible roles
Admin
Admin Portal
API Gateway
User Management Service
Authorization Service
Audit Service
Database

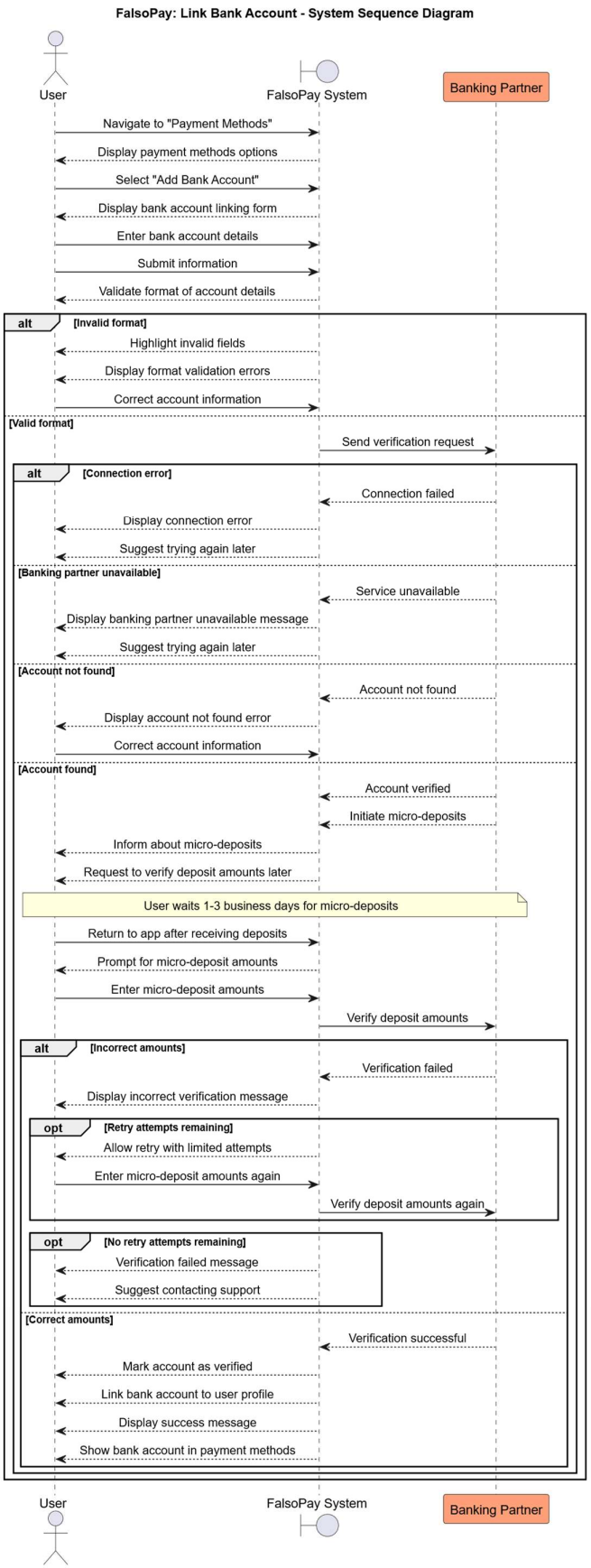

Please use '!option handwritten true' to enable handwritten
UC-38: Set Ticket Status
Admin
Admin Portal
API Gateway
Support Service
Notification Service
FalsoPay App
User
Database
Supervisor
Main Success Scenario
Access support tickets
Request ticket list
Get tickets
Query tickets
Return tickets
Return ticket list
Forward ticket list
Display ticket list
Select ticket to update
Request ticket details
Get ticket details
Query ticket data
Return ticket data
Return ticket details
Forward ticket details
Display ticket details
Select new status
Add status note
Submit status update
Process status update
Update ticket status
Status updated
Add status note
Note added
Request user notification
Generate notification
Send status update notification
Display ticket status update
Update successful
Forward success response
Display success confirmation
Alternative Scenario: Status Change Requires Approval
Access support tickets
Request ticket list
Get tickets
Query tickets
Return tickets
Return ticket list
Forward ticket list
Display ticket list
Select ticket to update
Request ticket details
Get ticket details
Query ticket data
Return ticket data
Return ticket details
Forward ticket details
Display ticket details
Select sensitive status change
Add status note
Submit status update
Process status update
Detect sensitive status change
Flag ticket for supervisor review
Ticket flagged
Request supervisor approval
Forward approval request
Display "Requires supervisor approval" message
Later - Supervisor reviews
Review flagged ticket
Request flagged ticket
Get flagged ticket
Query ticket data
Return ticket data
Return ticket details
Forward ticket details
Display ticket details with pending status change
Approve status change
Submit approval
Process approval
Update ticket status
Status updated
Request user notification
Generate notification
Send status update notification
Display ticket status update

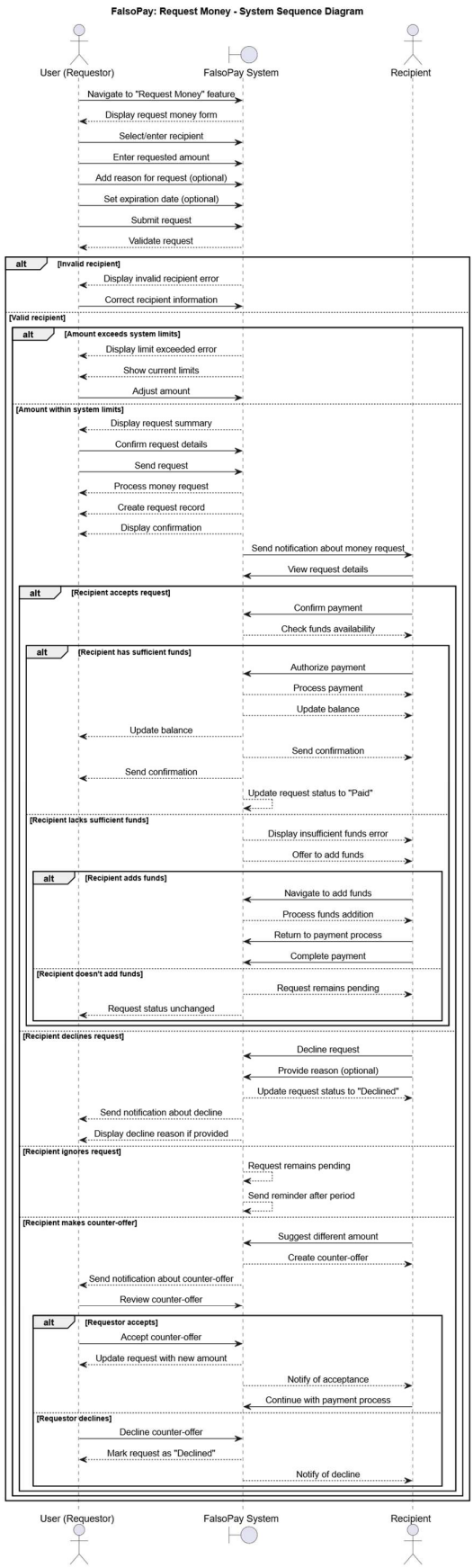

Please use '!option handwritten true' to enable handwritten
UC-39: Send Notification
Admin
Admin Portal
API Gateway
Notification Service
User Service
FalsoPay App
User
Database
Main Success Scenario
Access notification management
Display notification interface
Create notification content
Select target audience
Set notification priority
Submit notification
Process notification request
Validate notification content
Request target user list
Query users by criteria
Return matching users
Return target user list
Store notification
Notification stored
Process notification batch
Send notification to users
Deliver notification
Display notification
Update delivery status
Status updated
Return delivery report
Forward delivery report
Display delivery confirmation
Alternative Scenario: Some Users Unreachable
Access notification management
Display notification interface
Create notification content
Select target audience
Set notification priority
Submit notification
Process notification request
Validate notification content
Request target user list
Query users by criteria
Return matching users
Return target user list
Store notification
Notification stored
Process notification batch
Send notification to users
Attempt to deliver notification
Some users offline or unreachable
Update delivery status with partial success
Status updated
Return detailed delivery report
Forward delivery report
Display partial delivery report
Show unreachable users count
Offer retry option for failed deliveries
Select retry for failed deliveries
Request retry
Process retry request
Queue notifications for retry
Retry queued
Confirm retry queued
Display retry confirmation
Admin
Admin Portal
API Gateway
Notification Service
User Service
FalsoPay App
User
Database


Please use '!option handwritten true' to enable handwritten
UC-40: Refresh Tickets
Admin
Admin Portal
API Gateway
Support Service
Database
Main Success Scenario
View ticket dashboard
Request initial ticket data
Get ticket data
Query tickets
Return tickets
Return ticket data
Forward ticket data
Render ticket dashboard
Display ticket dashboard
Request ticket refresh
Send refresh request
Request latest tickets
Query latest ticket data
Return updated tickets
Return latest data
Forward updated data
Update ticket display
Update timestamp
Display refreshed tickets
Show refresh timestamp
Alternative Scenario: Connection Error
View ticket dashboard
Request initial ticket data
Get ticket data
Query tickets
Return tickets
Return ticket data
Forward ticket data
Render ticket dashboard
Display ticket dashboard
Request ticket refresh
Send refresh request
Connection error occurs
Return connection error
Display error message
Show retry button
Click retry
Retry refresh request
Request latest tickets
Query latest ticket data
Return updated tickets
Return latest data
Forward updated data
Update ticket display
Update timestamp
Display refreshed tickets
Show refresh timestamp
Admin
Admin Portal
API Gateway
Support Service
Database

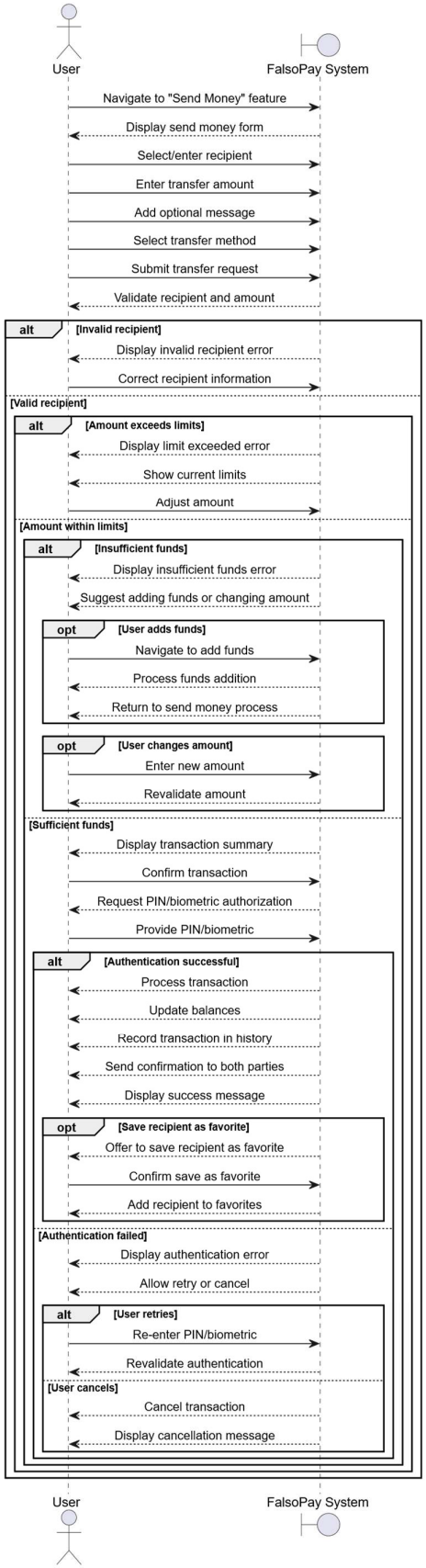

FalsoPay: Contact Customer Support - System Sequence Diagram
User
FalsoPay System
Support Agent
Navigate to "Help & Support"
Display support options
alt [Self-service first approach]
Show FAQ and self-help resources
Browse FAQs
alt [Issue resolved by FAQ]
Exit support section
[Issue not resolved]
Select "Contact Support Directly"
Display support contact form
Select issue category
Select issue priority
Provide issue description
Attach screenshots/evidence (optional)
Submit support request
Validate submission
alt [Incomplete submission]
Highlight missing information
Display submission error
Complete missing items and resubmit
[Complete submission]
Create support ticket
Assign ticket number
Determine priority
Send confirmation with ticket number
Display estimated response time
Route ticket to appropriate queue
Receive and open ticket
Review issue details
alt [Need more information]
Request additional information
Send notification requesting more info
Receive notification
Provide additional information
Forward additional information
Research issue
Prepare response
alt [Immediate resolution possible]
Resolve issue
Document solution
[Requires escalation]
Escalate to specialist team
Assign to specialist
Specialist resolves issue
[Partial solution]
Provide workaround
Schedule follow-up
Update ticket status
Send resolution notification
Review resolution
alt [User satisfied]
Mark issue as resolved (optional)
Close ticket
Send satisfaction survey
[User not satisfied]
Respond with additional concerns
Reopen or update ticket
Address additional concerns
User
FalsoPay System
Support Agent


FalsoPay: Dispute Transaction - System Sequence Diagram
User
FalsoPay System
Support Agent
Other Party
Navigate to transaction history
Display transaction list
Select disputed transaction
Choose "Report a Problem"
Display problem categories
Select issue category
Provide issue details
Upload supporting evidence (optional)
Submit dispute
Validate submission
alt
[Incomplete submission]
Highlight missing information
Display submission error
Complete missing items and resubmit
[Complete submission]
Create dispute case
Assign case number
Determine dispute priority
Send confirmation with reference number
Display estimated resolution timeline
Notify about dispute
alt
[Other party accepts dispute]
Accept dispute claim
Process automatic refund
Update transaction status
Send resolution notification
Send resolution notification
[Other party contests dispute]
Contest dispute claim
Assign to dispute resolution team
opt
[Additional information needed]
Request more information
Request for additional information
Provide additional details
Forward additional information
Review case and make decision
alt
[Decision favors user]
Process refund
Update transaction status
Send favorable resolution notification
Send explanation notification
[Decision against user]
Maintain original transaction
Send explanation notification
opt
[Appeal available]
Inform of appeal process
alt
[User appeals]
Submit appeal
Create appeal case
Escalate to higher review level
[User accepts decision]
Close dispute case
[No appeal available]
Close dispute case
User
FalsoPay System
Support Agent
Other Party

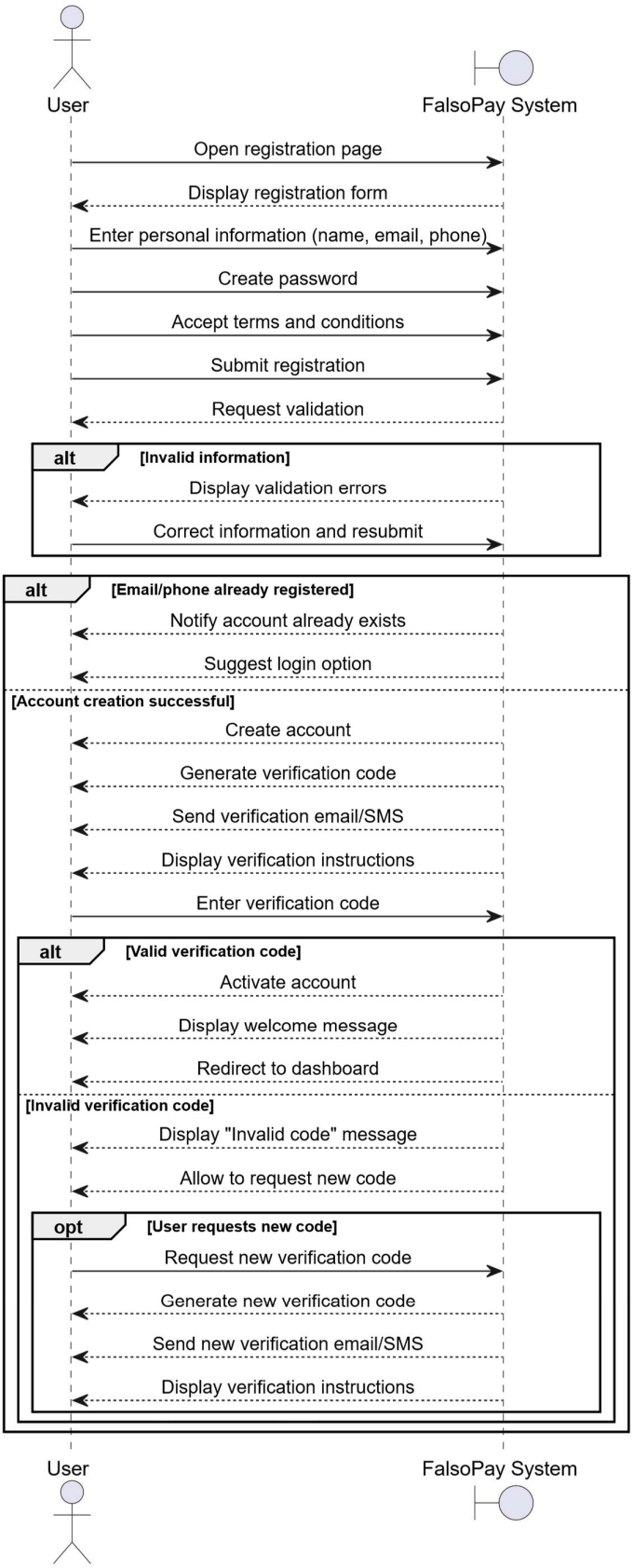

FalsoPay: Link Bank Account - System Sequence Diagram
User
FalsoPay System
Banking Partner
Navigate to "Payment Methods"
Display payment methods options
Select "Add Bank Account"
Display bank account linking form
Enter bank account details
Submit information
Validate format of account details
alt [Invalid format]
Highlight invalid fields
Display format validation errors
Correct account information
[Valid format]
Send verification request
alt [Connection error]
Connection failed
Display connection error
Suggest trying again later
[Banking partner unavailable]
Service unavailable
Display banking partner unavailable message
Suggest trying again later
[Account not found]
Account not found
Display account not found error
Correct account information
[Account found]
Account verified
Initiate micro-deposits
Inform about micro-deposits
Request to verify deposit amounts later
User waits 1-3 business days for micro-deposits
Return to app after receiving deposits
Prompt for micro-deposit amounts
Enter micro-deposit amounts
Verify deposit amounts
alt [Incorrect amounts]
Verification failed
Display incorrect verification message
opt [Retry attempts remaining]
Allow retry with limited attempts
Enter micro-deposit amounts again
Verify deposit amounts again
opt [No retry attempts remaining]
Verification failed message
Suggest contacting support
[Correct amounts]
Verification successful
Mark account as verified
Link bank account to user profile
Display success message
Show bank account in payment methods
User
FalsoPay System
Banking Partner


FalsoPay: Request Money - System Sequence Diagram
User (Requestor)
FalsoPay System
Recipient
Navigate to "Request Money" feature
Display request money form
Select/enter recipient
Enter requested amount
Add reason for request (optional)
Set expiration date (optional)
Submit request
Validate request
alt [Invalid recipient]
Display invalid recipient error
Correct recipient information
[Valid recipient]
alt [Amount exceeds system limits]
Display limit exceeded error
Show current limits
Adjust amount
[Amount within system limits]
Display request summary
Confirm request details
Send request
Process money request
Create request record
Display confirmation
Send notification about money request
View request details
alt [Recipient accepts request]
Confirm payment
Check funds availability
alt [Recipient has sufficient funds]
Authorize payment
Process payment
Update balance
Update balance
Send confirmation
Send confirmation
Update request status to "Paid"
[Recipient lacks sufficient funds]
Display insufficient funds error
Offer to add funds
alt [Recipient adds funds]
Navigate to add funds
Process funds addition
Return to payment process
Complete payment
[Recipient doesn't add funds]
Request remains pending
Request status unchanged
[Recipient declines request]
Decline request
Provide reason (optional)
Update request status to "Declined"
Send notification about decline
Display decline reason if provided
[Recipient ignores request]
Request remains pending
Send reminder after period
[Recipient makes counter-offer]
Suggest different amount
Create counter-offer
Send notification about counter-offer
Review counter-offer
alt [Requestor accepts]
Accept counter-offer
Update request with new amount
Notify of acceptance
Continue with payment process
[Requestor declines]
Decline counter-offer
Mark request as "Declined"
Notify of decline
User (Requestor)
FalsoPay System
Recipient

FalsoPay: Send Money - System Sequence Diagram

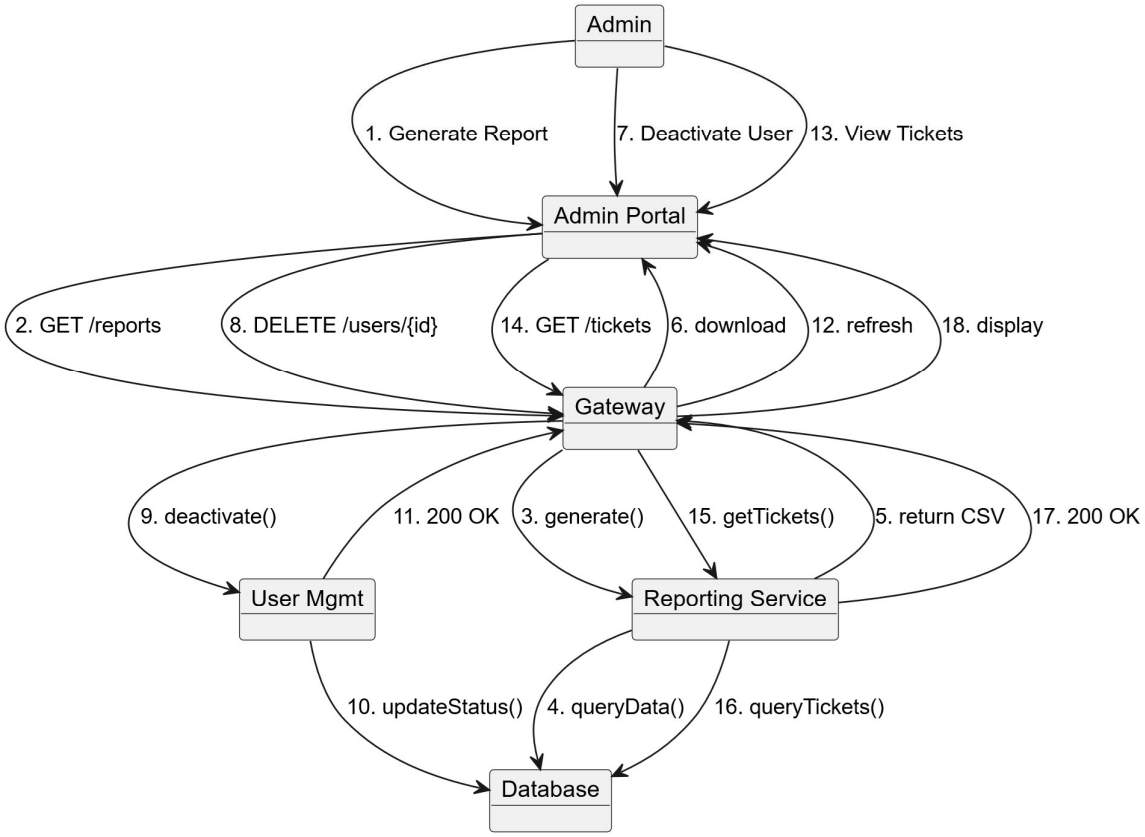


FalsoPay: User Registration - System Sequence Diagram

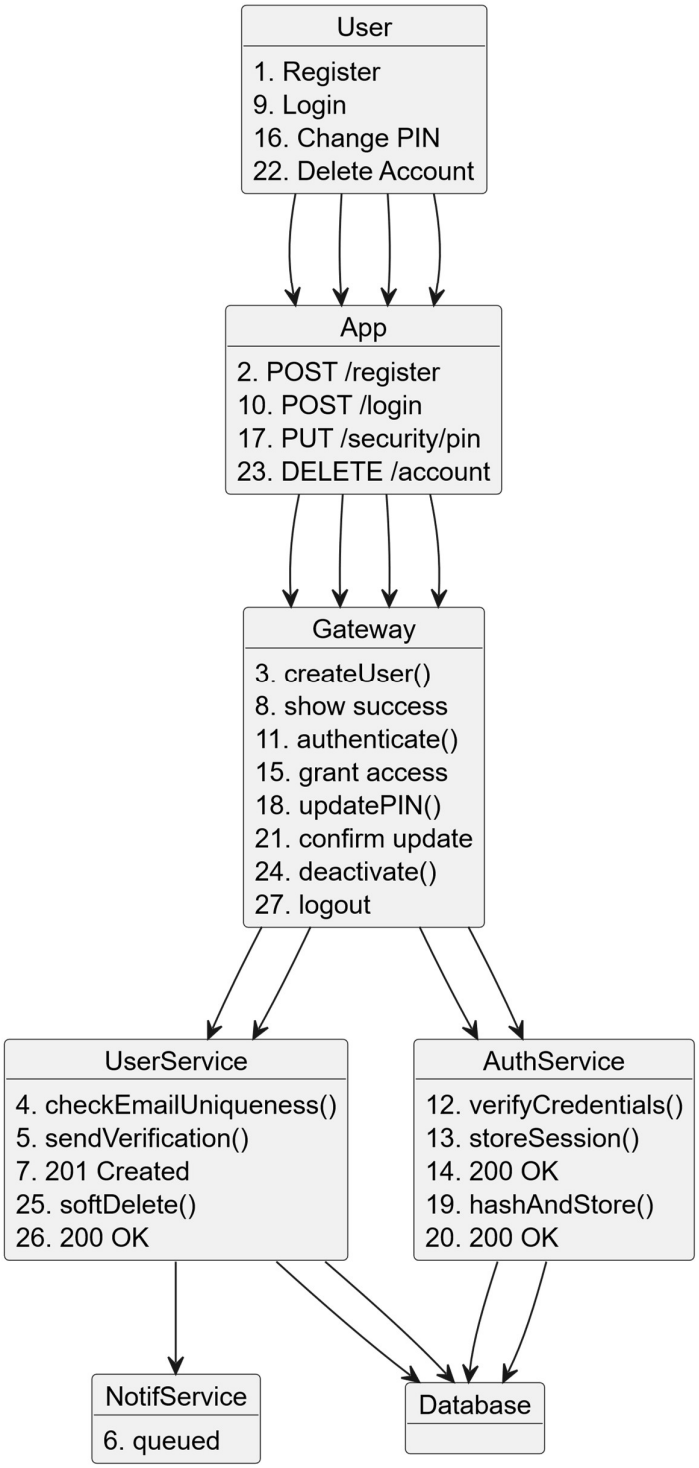


d) Collaboration Diagrams

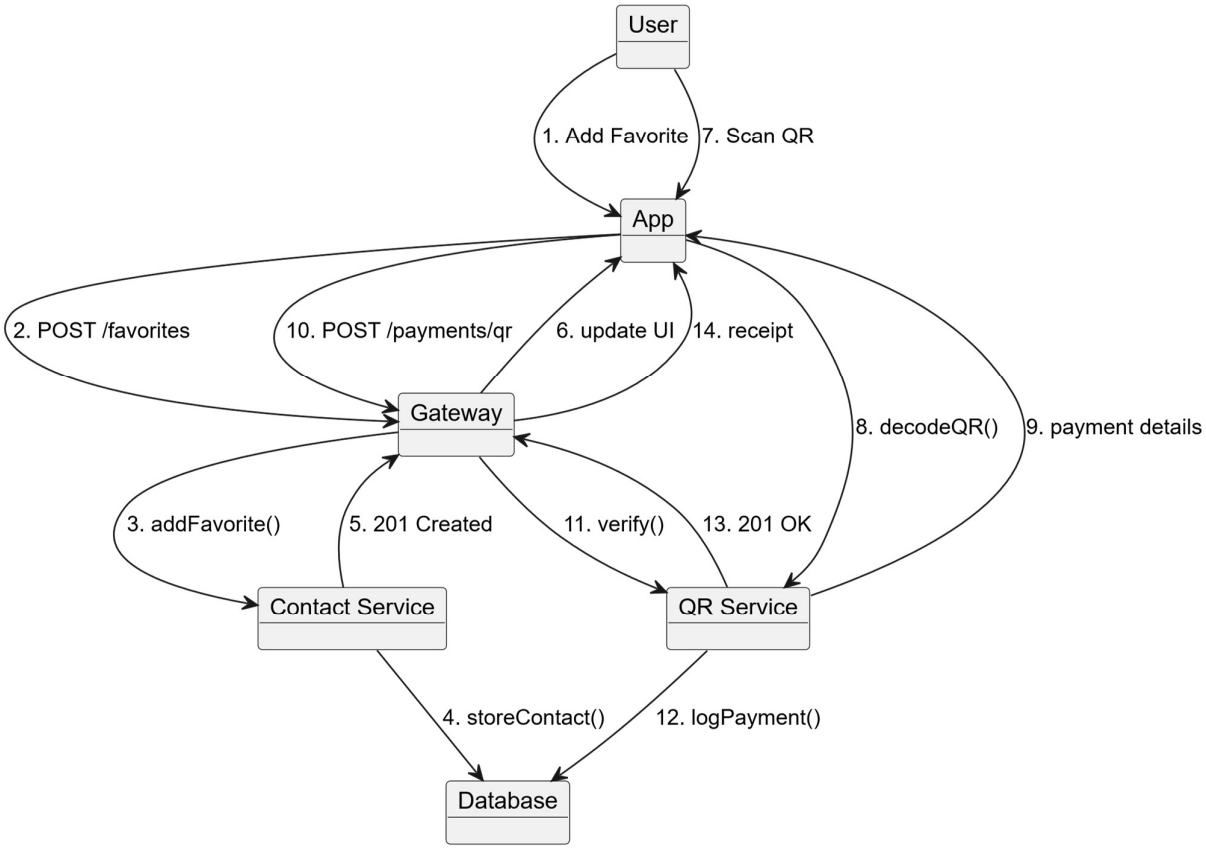
Collaboration Diagram: Admin & Reporting
(UC33, UC35-UC40)



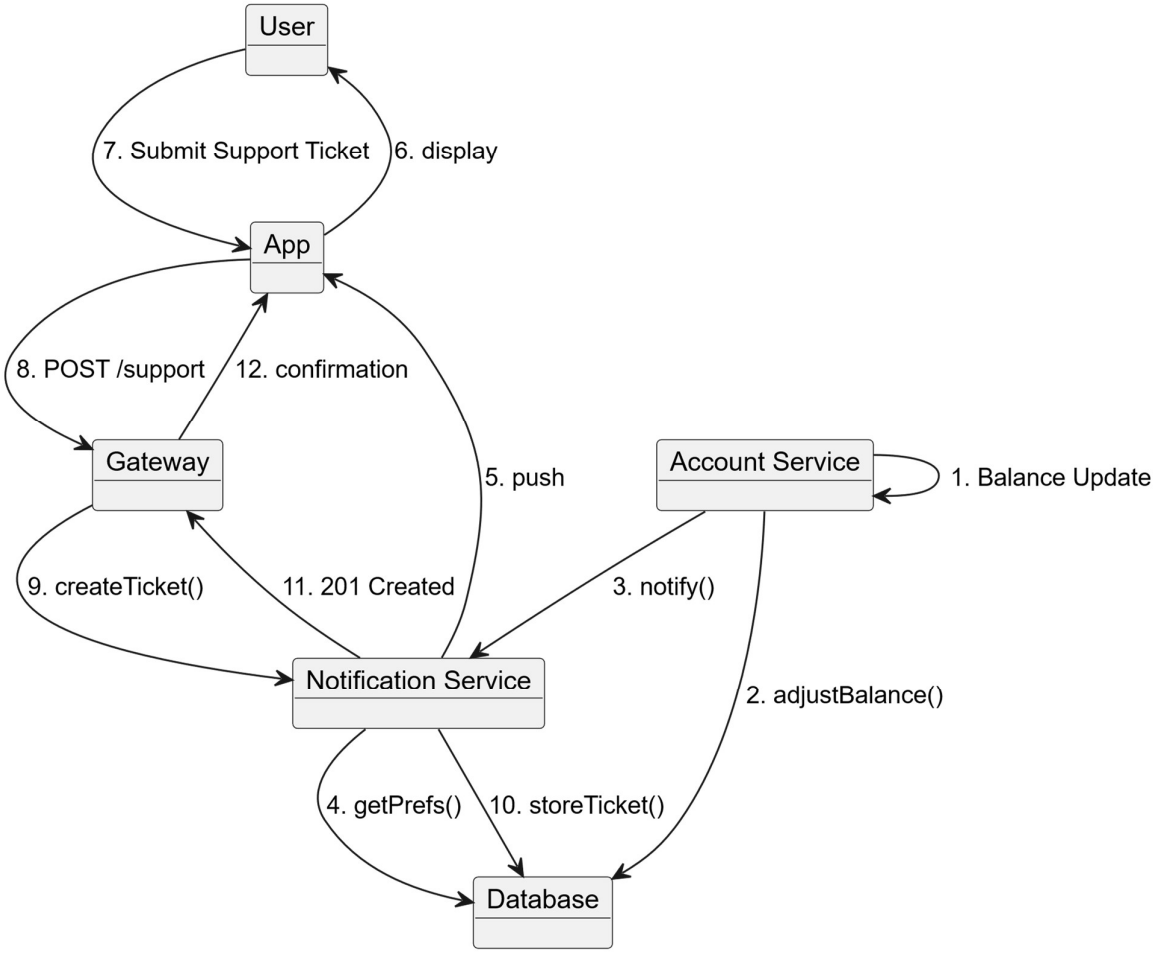
Collaboration Diagram: User Authentication & Profile Management (UC1, UC2, UC16, UC22)



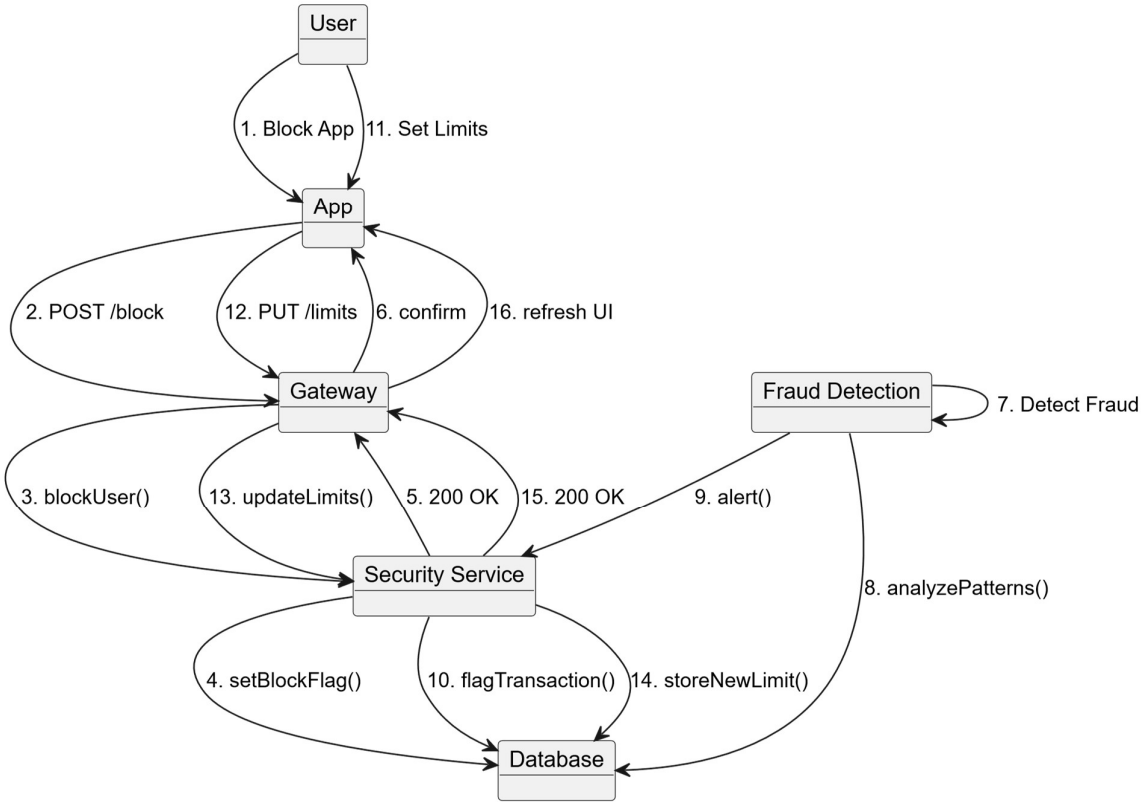
Collaboration Diagram: Contacts & QR Payments
(UC6, UC11, UC13-14, UC20-21, UC25-27)



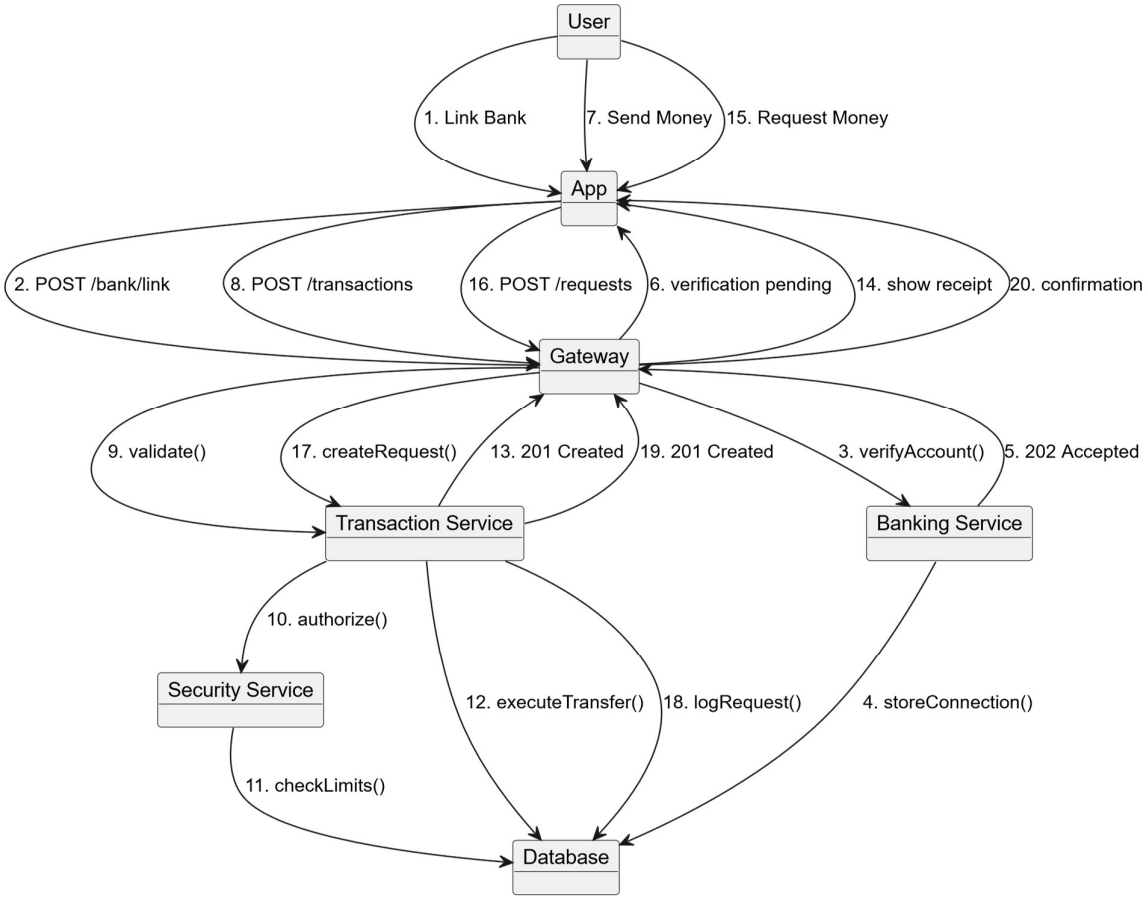
Collaboration Diagram: Notifications & Balances
(UC15, UC17, UC24, UC31)



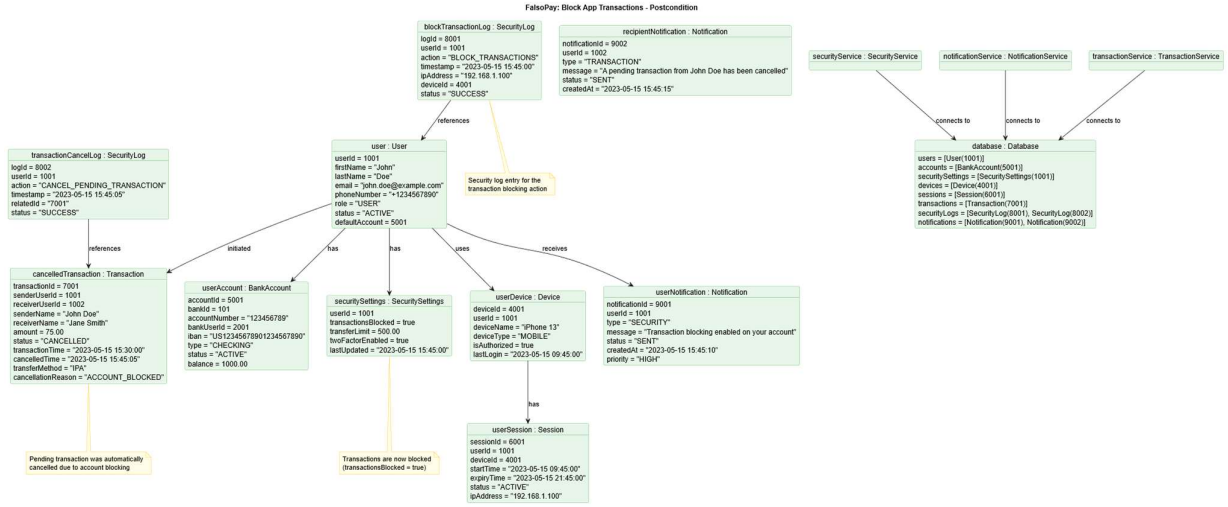
Collaboration Diagram: Security & System Controls
(UC19, UC23, UC28-30, UC32, UC34)

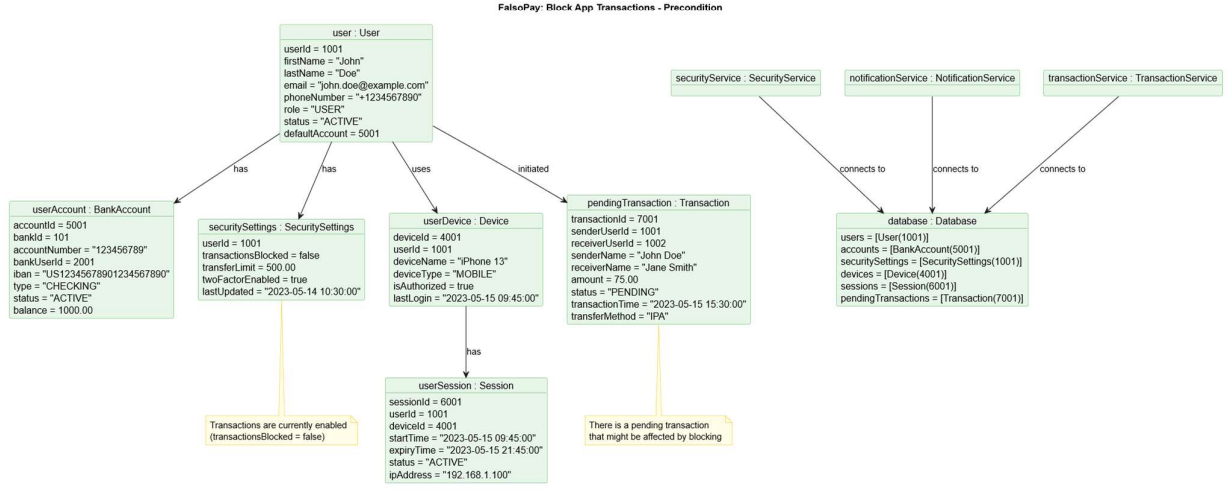


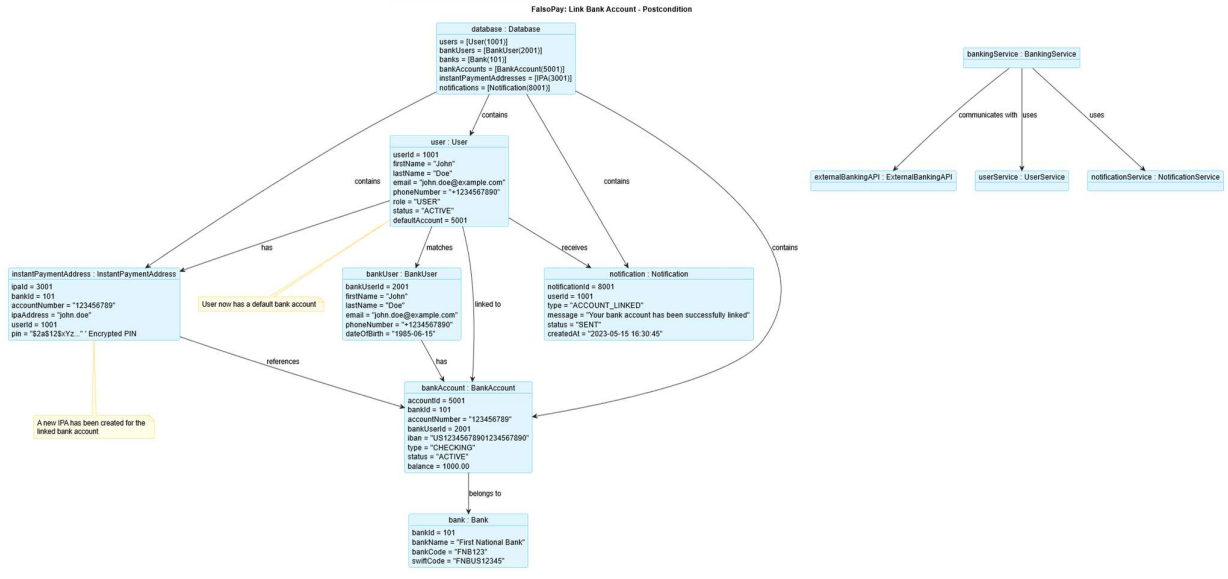
Collaboration Diagram (with Flow Arrows): Financial Transactions Core
(UC5, UC8-UC9, UC12)



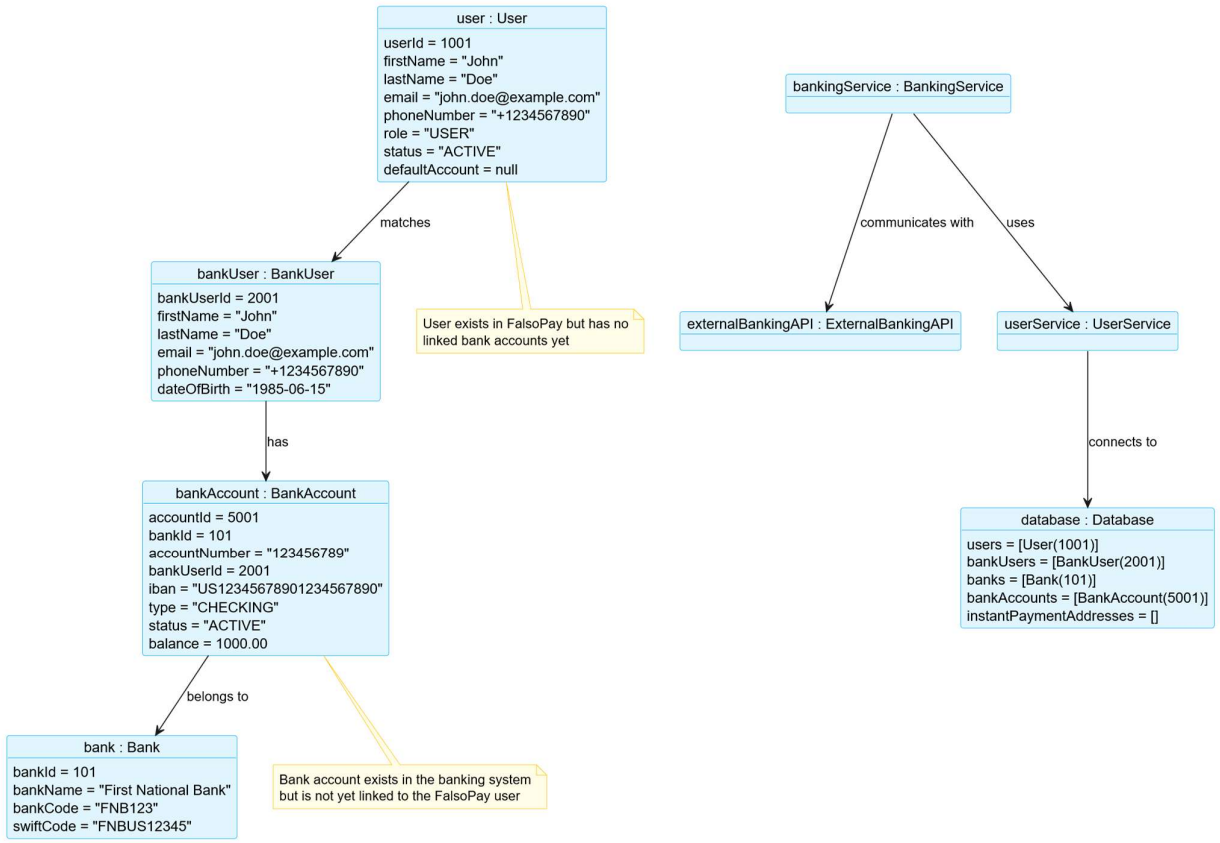
e) Object Diagrams



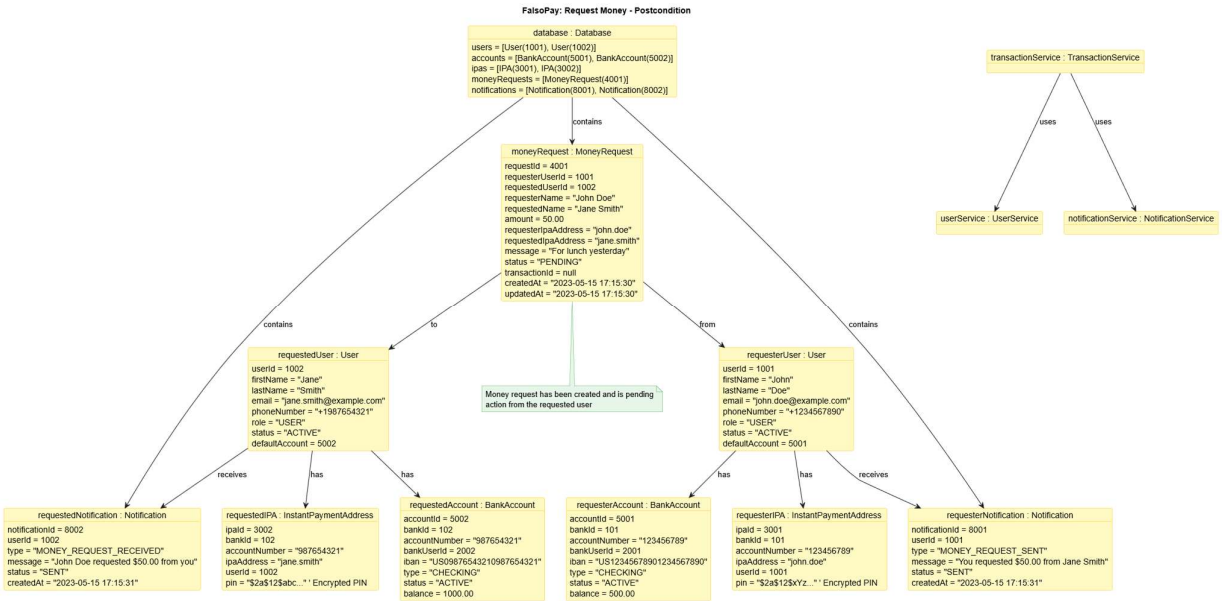


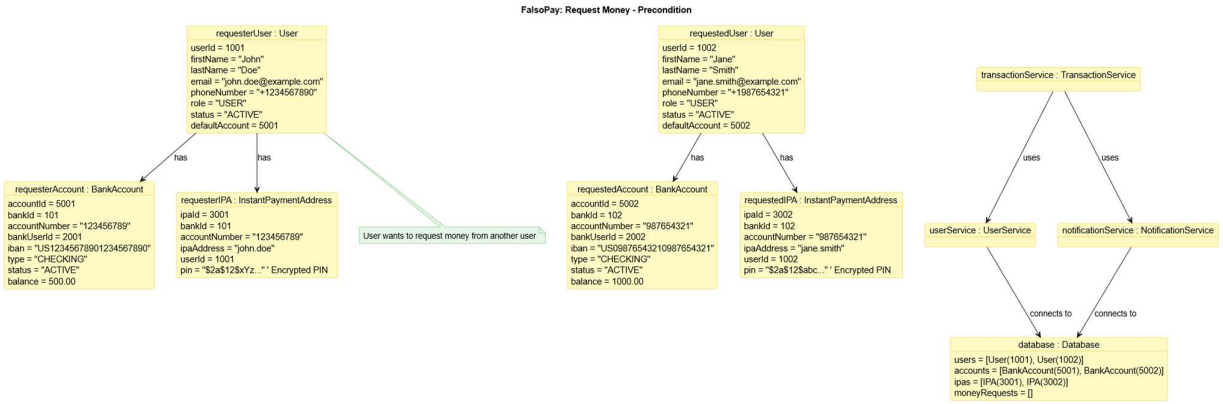


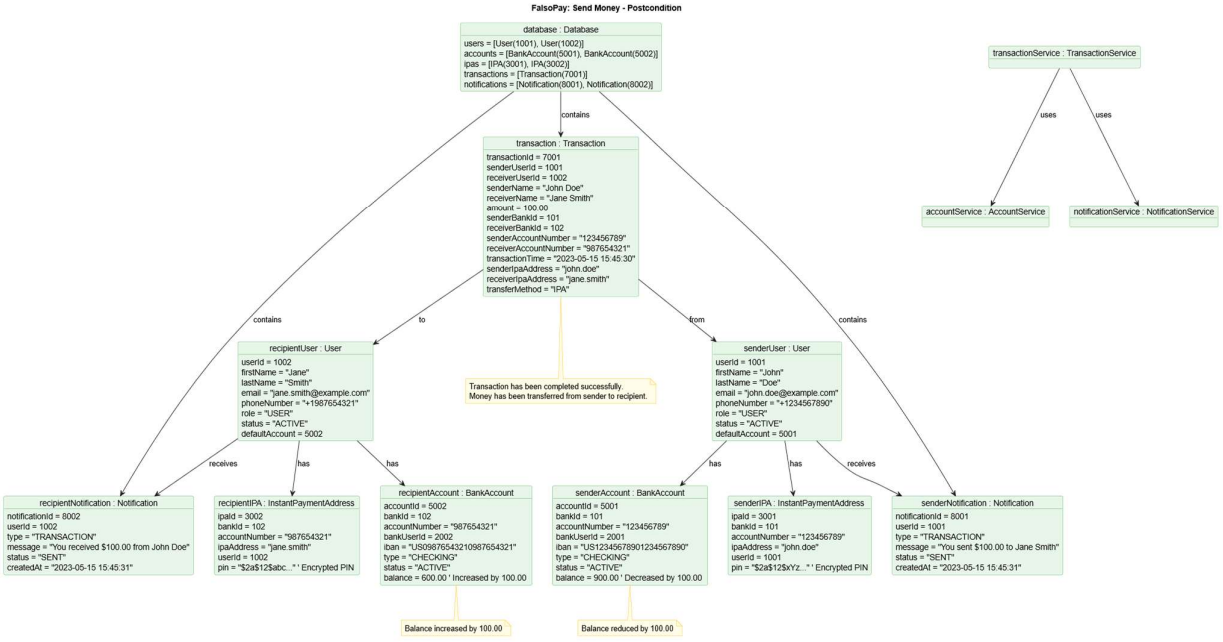
FalsoPay: Link Bank Account - Precondition

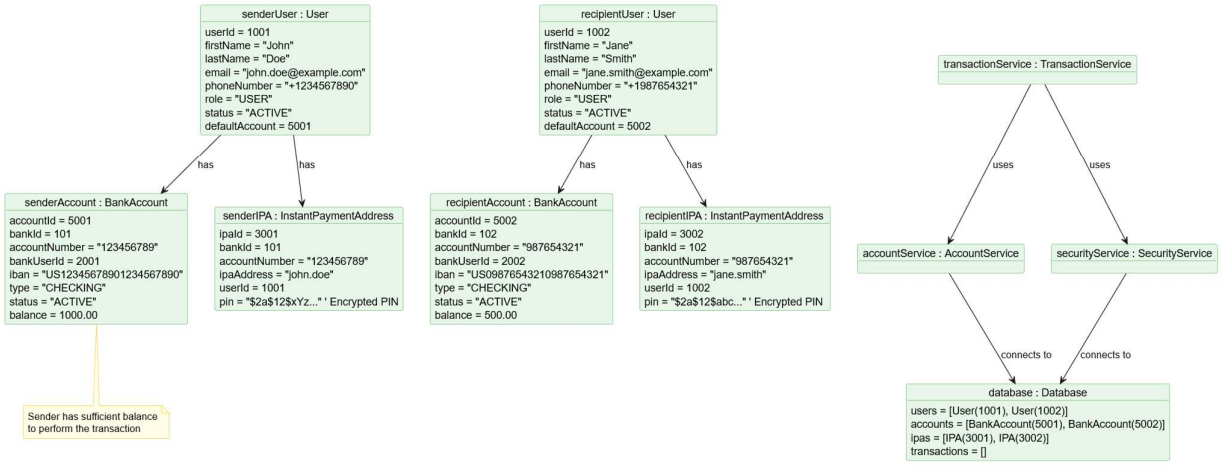


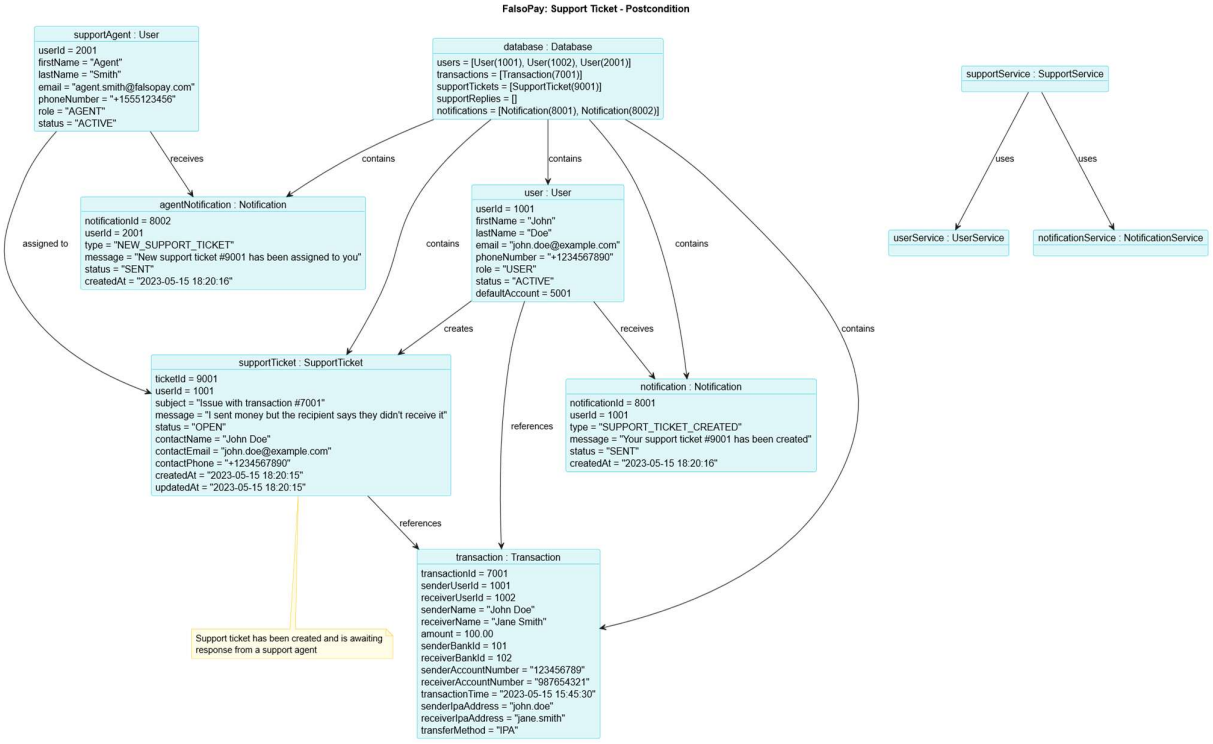
FalsoPay: Request Money - Postcondition



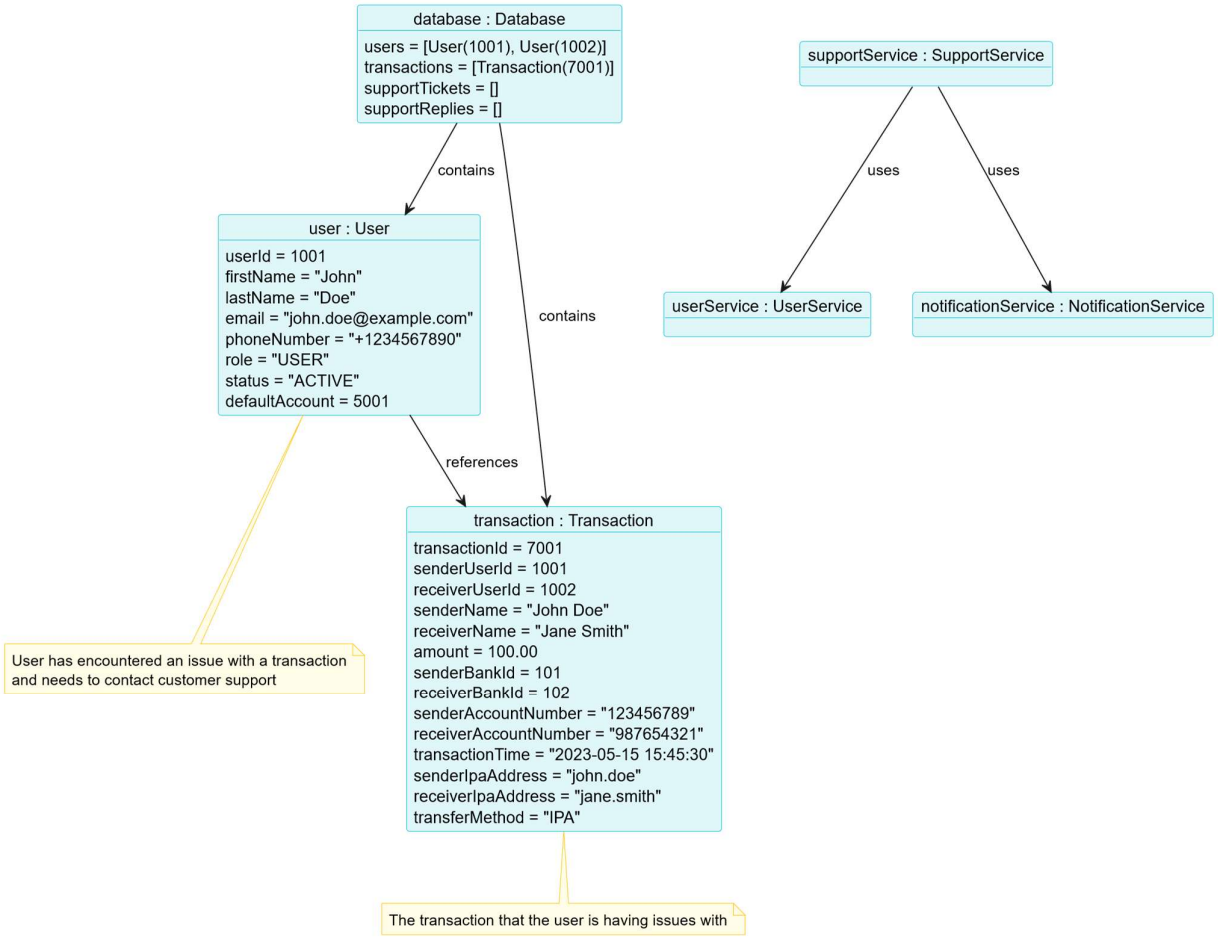

FalsoPay: Request Money - Precondition
requesterUser : User
userId = 1001
firstName = "John"
lastName = "Doe"
email = "john.doe@example.com"
phoneNumber = "+1234567890"
role = "USER"
status = "ACTIVE"
defaultAccount = 5001
requestedUser : User
userId = 1002
firstName = "Jane"
lastName = "Smith"
email = "jane.smith@example.com"
phoneNumber = "+1987654321"
role = "USER"
status = "ACTIVE"
defaultAccount = 5002
transactionService : TransactionService
has
has
has
has
uses
uses
requesterAccount : BankAccount
accountId = 5001
bankId = 101
accountNumber = "123456789"
bankUserId = 2001
iban = "US1234567890123456789O"
type = "CHECKING"
status = "ACTIVE"
balance = 500.00
requesterIPA : InstantPaymentAddress
ipaId = 3001
bankId = 101
accountNumber = "123456789"
ipaAddress = "john.doe"
userId = 1001
pin = "$2a$12$xYz..." ' Encrypted PIN
User wants to request money from another user
requestedAccount : BankAccount
accountId = 5002
bankId = 102
accountNumber = "987654321"
bankUserId = 2002
iban = "US0987654321098765432 1"
type = "CHECKING"
status = "ACTIVE"
balance = 1000.00
requestedIPA : InstantPaymentAddress
ipaId = 3002
bankId = 102
accountNumber = "98765432 1"
ipaAddress = "jane.smith"
userId = 1002
pin = "$2a$12$abc..." ' Encrypted PIN
userService : UserService
notificationService : NotificationService
connects to
connects to
database : Database
users = [User(1001), User(1002)]
accounts = [BankAccount(5001), BankAccount(5002)]
ipas = [IPA(3001), IPA(3002)]
moneyRequests = []


FalsoPay: Send Money - Postcondition
database : Database
users = [User(1001), User(1002)]
accounts = [BankAccount(5001), BankAccount(5002)]
ipas = [IPA(3001), IPA(3002)]
transactions = [Transaction(7001)]
notifications = [Notification(8001), Notification(8002)]
transactionService : TransactionService
contains
uses
uses
transaction : Transaction
transactionId = 7001
senderUserId = 1001
receiverUserId = 1002
senderName = "John Doe"
receiverName = "Jane Smith"
amount = 100.00
senderBankId = 101
receiverBankId = 102
senderAccountNumber = "123456789"
receiverAccountNumber = "987654321"
transactionTime = "2023-05-15 15:45:30"
senderIpaAddress = "john.doe"
receiverIpaAddress = "jane.smith"
transferMethod = "IPA"
accountService : AccountService
notificationService : NotificationService
contains
to
from
contains
recipientUser : User
userId = 1002
firstName = "Jane"
lastName = "Smith"
email = "jane.smith@example.com"
phoneNumber = "+1987654321"
role = "USER"
status = "ACTIVE"
defaultAccount = 5002
Transaction has been completed successfully.
Money has been transferred from sender to recipient.
senderUser : User
userId = 1001
firstName = "John"
lastName = "Doe"
email = "john.doe@example.com"
phoneNumber = "+1234567890"
role = "USER"
status = "ACTIVE"
defaultAccount = 5001
receives
has
has
has
has
receives
recipientNotification : Notification
notificationId = 8002
userId = 1002
type = "TRANSACTION"
message = "You received $100.00 from John Doe"
status = "SENT"
createdAt = "2023-05-15 15:45:31"
recipientIPA : InstantPaymentAddress
ipaId = 3002
bankId = 102
accountNumber = "98765432 1"
ipaAddress = "jane.smith"
userId = 1002
pin = "$2a$12$abc..." ' Encrypted PIN
recipientAccount : BankAccount
accountId = 5002
bankId = 102
accountNumber = "987654321"
bankUserId = 2002
iban = "US0987654321098765432 1"
type = "CHECKING"
status = "ACTIVE"
balance = 600.00 ' Increased by 100.00
senderAccount : BankAccount
accountId = 5001
bankId = 101
accountNumber = "123456789"
bankUserId = 2001
iban = "US1234567890123456789O"
type = "CHECKING"
status = "ACTIVE"
balance = 900.00 ' Decreased by 100.00
senderIPA : InstantPaymentAddress
ipaId = 3001
bankId = 101
accountNumber = "123456789"
ipaAddress = "john.doe"
userId = 1001
pin = "$2a$12$xYz..." ' Encrypted PIN
senderNotification : Notification
notificationId = 8001
userId = 1001
type = "TRANSACTION"
message = "You sent $100.00 to Jane Smith"
status = "SENT"
createdAt = "2023-05-15 15:45:31"
Balance increased by 100.00
Balance reduced by 100.00

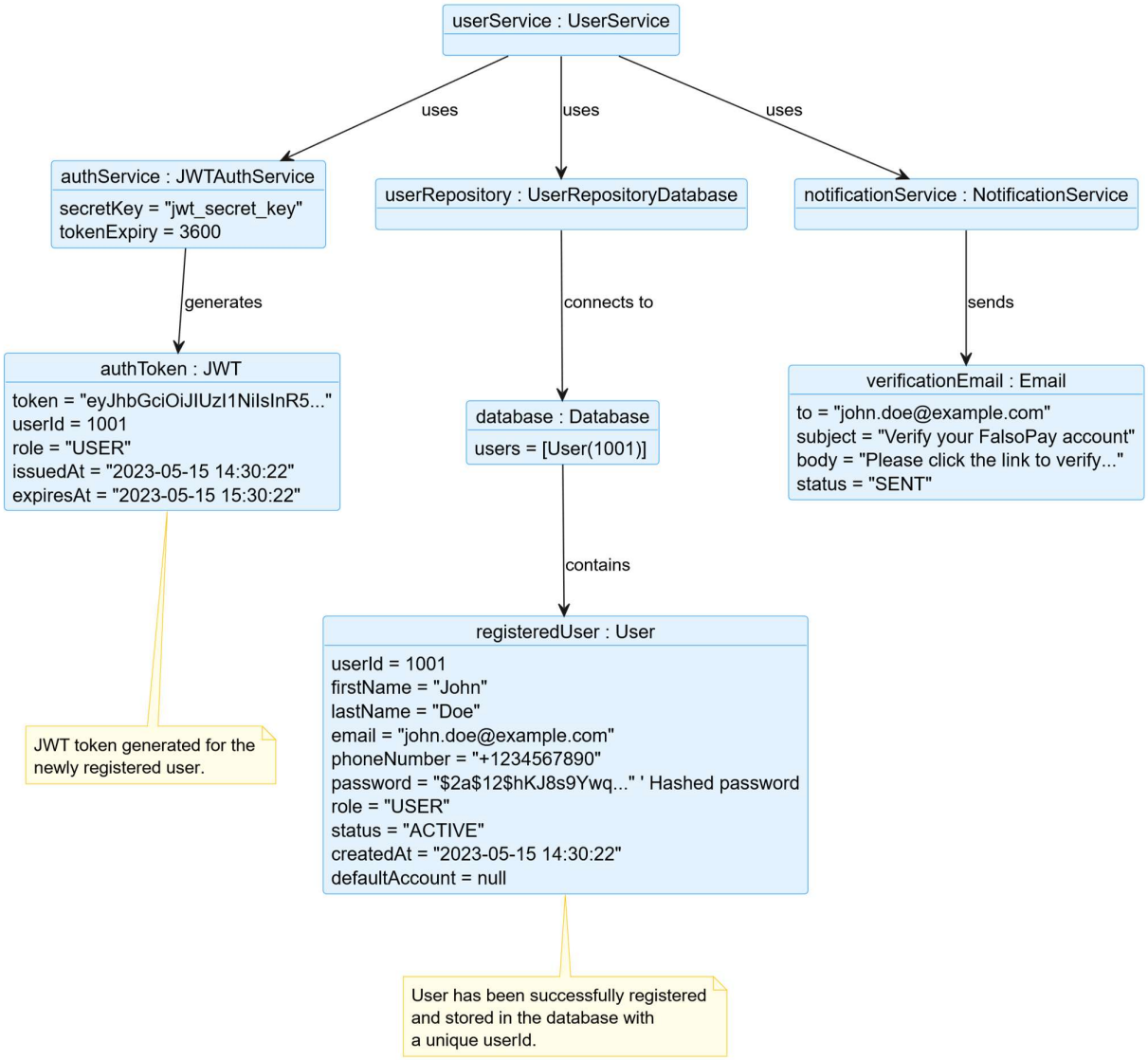

FalsoPay: Send Money - Precondition
senderUser : User
userId = 1001
firstName = "John"
lastName = "Doe"
email = "john.doe@example.com"
phoneNumber = "+1234567890"
role = "USER"
status = "ACTIVE"
defaultAccount = 5001
recipientUser : User
userId = 1002
firstName = "Jane"
lastName = "Smith"
email = "jane.smith@example.com"
phoneNumber = "+1987654321"
role = "USER"
status = "ACTIVE"
defaultAccount = 5002
transactionService : TransactionService
has
has
has
has
uses
uses
senderAccount : BankAccount
accountId = 5001
bankId = 101
accountNumber = "123456789"
bankUserId = 2001
iban = "US1234567890123456789O"
type = "CHECKING"
status = "ACTIVE"
balance = 1000.00
senderIPA : InstantPaymentAddress
ipaId = 3001
bankId = 101
accountNumber = "123456789"
ipaAddress = "john.doe"
userId = 1001
pin = "$2a$12$xYz..." ' Encrypted PIN
recipientAccount : BankAccount
accountId = 5002
bankId = 102
accountNumber = "987654321"
bankUserId = 2002
iban = "US0987654321098765432 1"
type = "CHECKING"
status = "ACTIVE"
balance = 500.00
recipientIPA : InstantPaymentAddress
ipaId = 3002
bankId = 102
accountNumber = "98765432 1"
ipaAddress = "jane.smith"
userId = 1002
pin = "$2a$12$abc..." ' Encrypted PIN
accountService : AccountService
securityService : SecurityService
Sender has sufficient balance
to perform the transaction
connects to
connects to
database : Database
users = [User(1001), User(1002)]
accounts = [BankAccount(5001), BankAccount(5002)]
ipas = [IPA(3001), IPA(3002)]
transactions = []

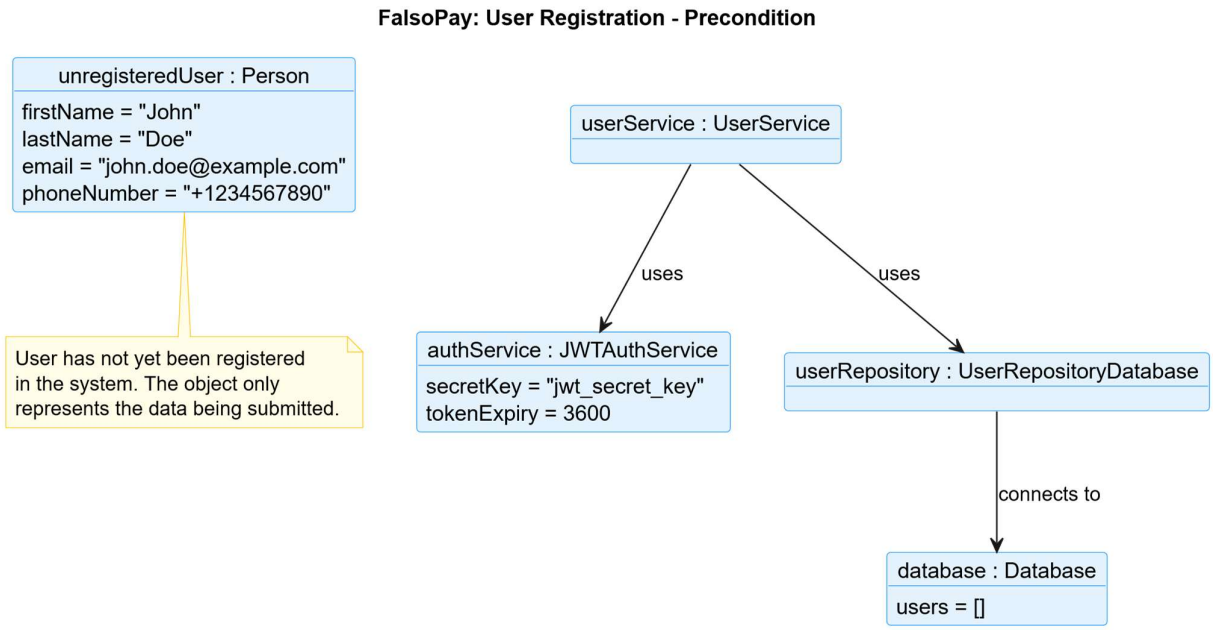

FalsoPay: Support Ticket - Postcondition
supportAgent : User
userId = 2001
firstName = "Agent"
lastName = "Smith"
email = "agent.smith@falsopay.com"
phoneNumber = "+1555123456"
role = "AGENT"
status = "ACTIVE"
database : Database
users = [User(1001), User(1002), User(2001)]
transactions = [Transaction(7001)]
supportTickets = [SupportTicket(9001)]
supportReplies = []
notifications = [Notification(8001), Notification(8002)]
supportService : SupportService
userService : UserService
notificationService : NotificationService
agentNotification : Notification
notificationId = 8002
userId = 2001
type = "NEW_SUPPORT_TICKET"
message = "New support ticket #9001 has been assigned to you"
status = "SENT"
createdAt = "2023-05-15 18:20:16"
user : User
userId = 1001
firstName = "John"
lastName = "Doe"
email = "john.doe@example.com"
phoneNumber = "+1234567890"
role = "USER"
status = "ACTIVE"
defaultAccount = 5001
supportTicket : SupportTicket
ticketId = 9001
userId = 1001
subject = "Issue with transaction #7001"
message = "I sent money but the recipient says they didn't receive it"
status = "OPEN"
contactName = "John Doe"
contactEmail = "john.doe@example.com"
contactPhone = "+1234567890"
createdAt = "2023-05-15 18:20:15"
updatedAt = "2023-05-15 18:20:15"
notification : Notification
notificationId = 8001
userId = 1001
type = "SUPPORT_TICKET_CREATED"
message = "Your support ticket #9001 has been created"
status = "SENT"
createdAt = "2023-05-15 18:20:16"
transaction : Transaction
transactionId = 7001
senderUserId = 1001
receiverUserId = 1002
senderName = "John Doe"
receiverName = "Jane Smith"
amount = 100.00
senderBankId = 101
receiverBankId = 102
senderAccountNumber = "123456789"
receiverAccountNumber = "987654321"
transactionTime = "2023-05-15 15:45:30"
senderIpaAddress = "john.doe"
receiverIpaAddress = "jane.smith"
transferMethod = "IPA"
Support ticket has been created and is awaiting response from a support agent
receives
contains
contains
uses
uses
assigned to
contains
contains
creates
receives
contains
references
references

FalsoPay: Support Ticket - Precondition

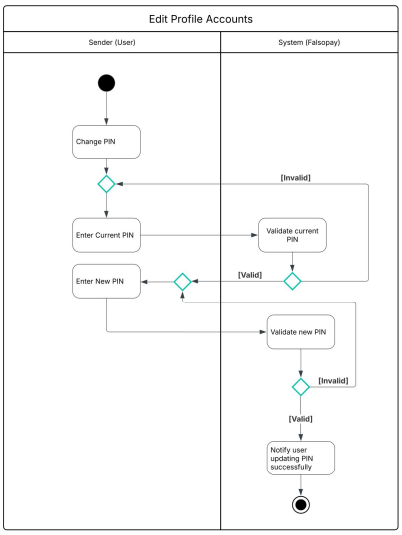


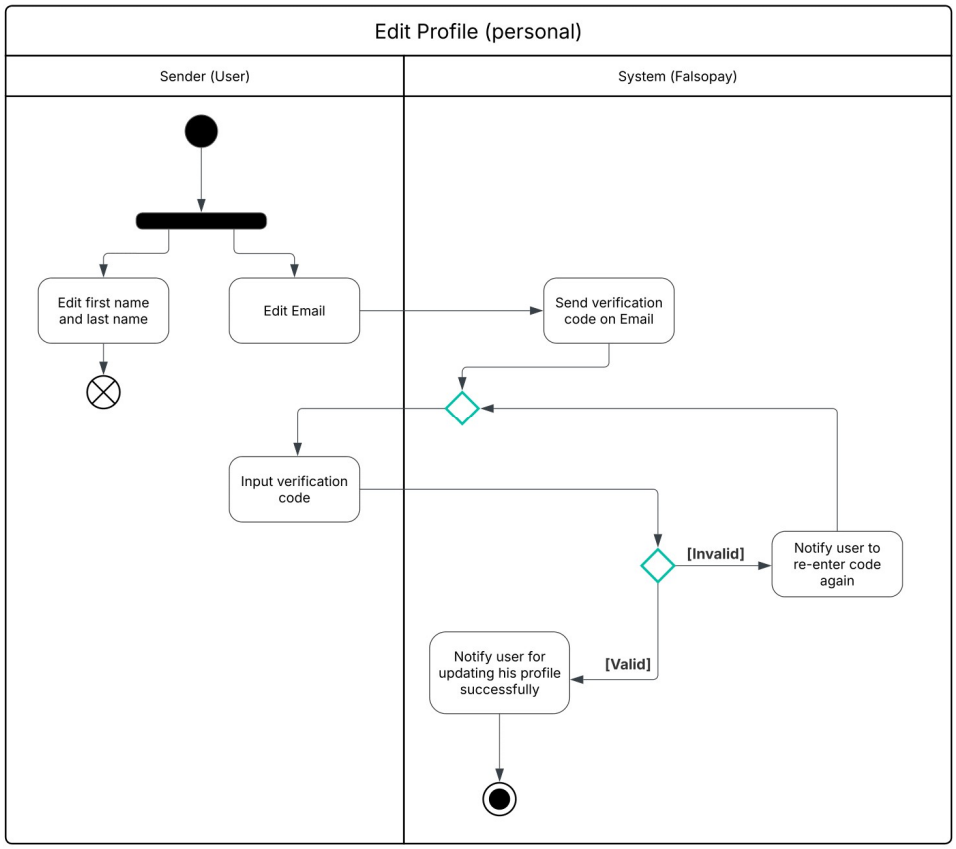
FalsoPay: User Registration - Postcondition

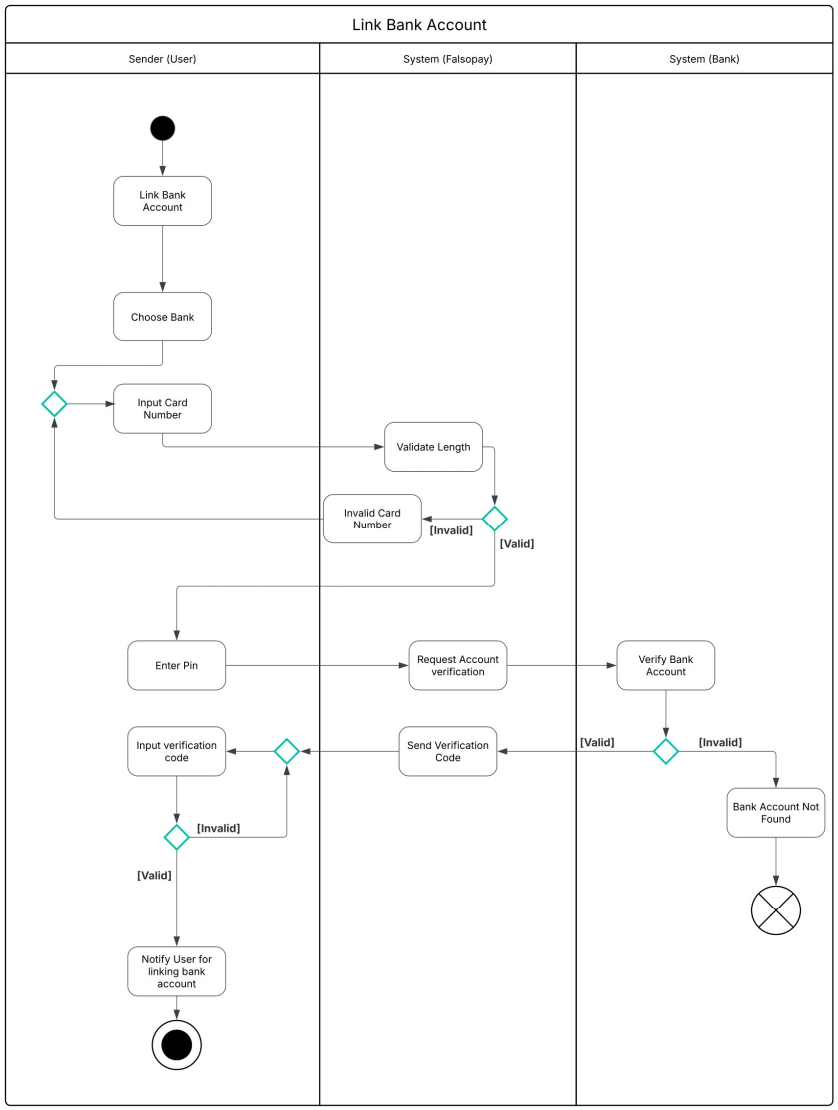


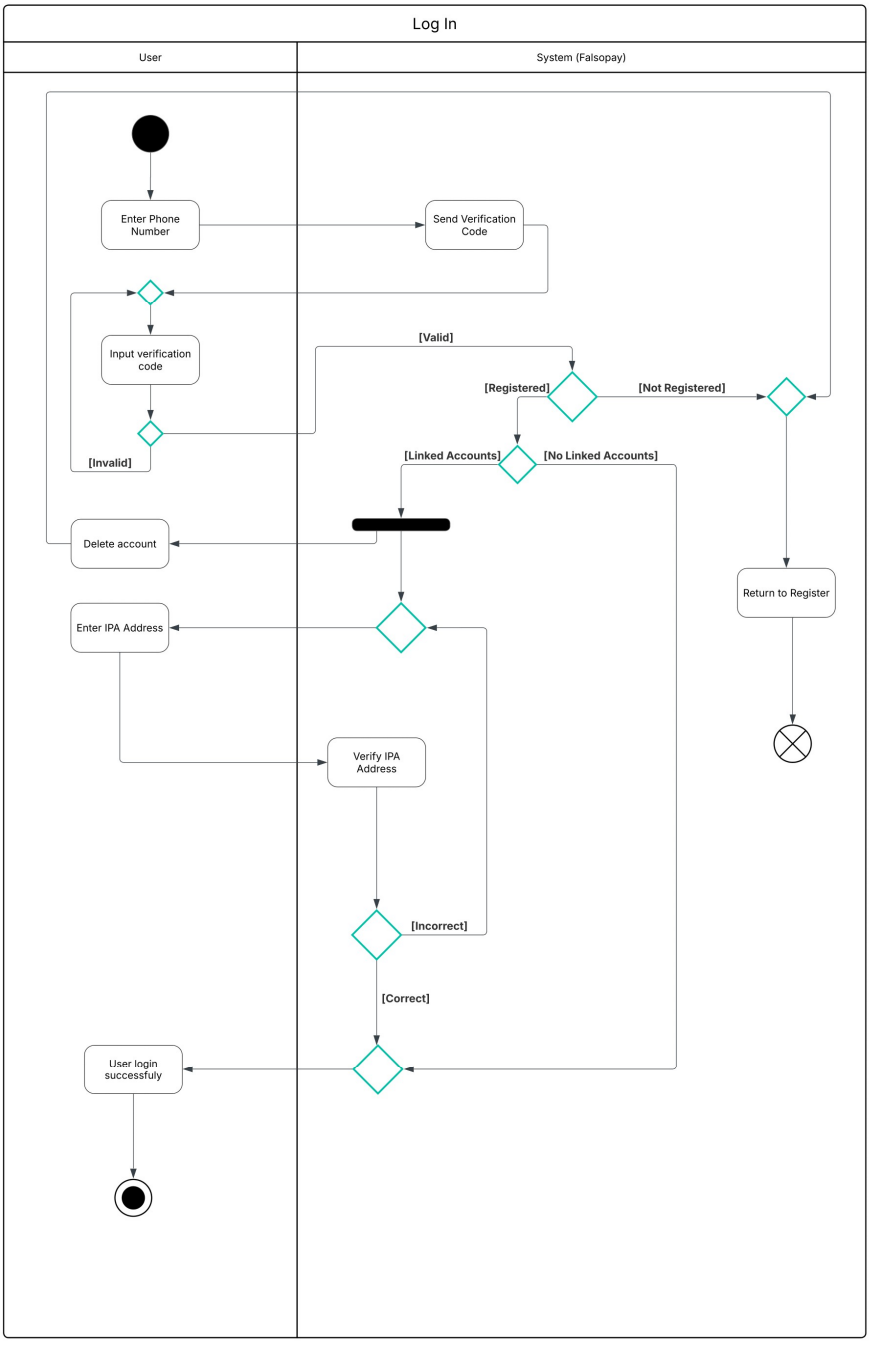


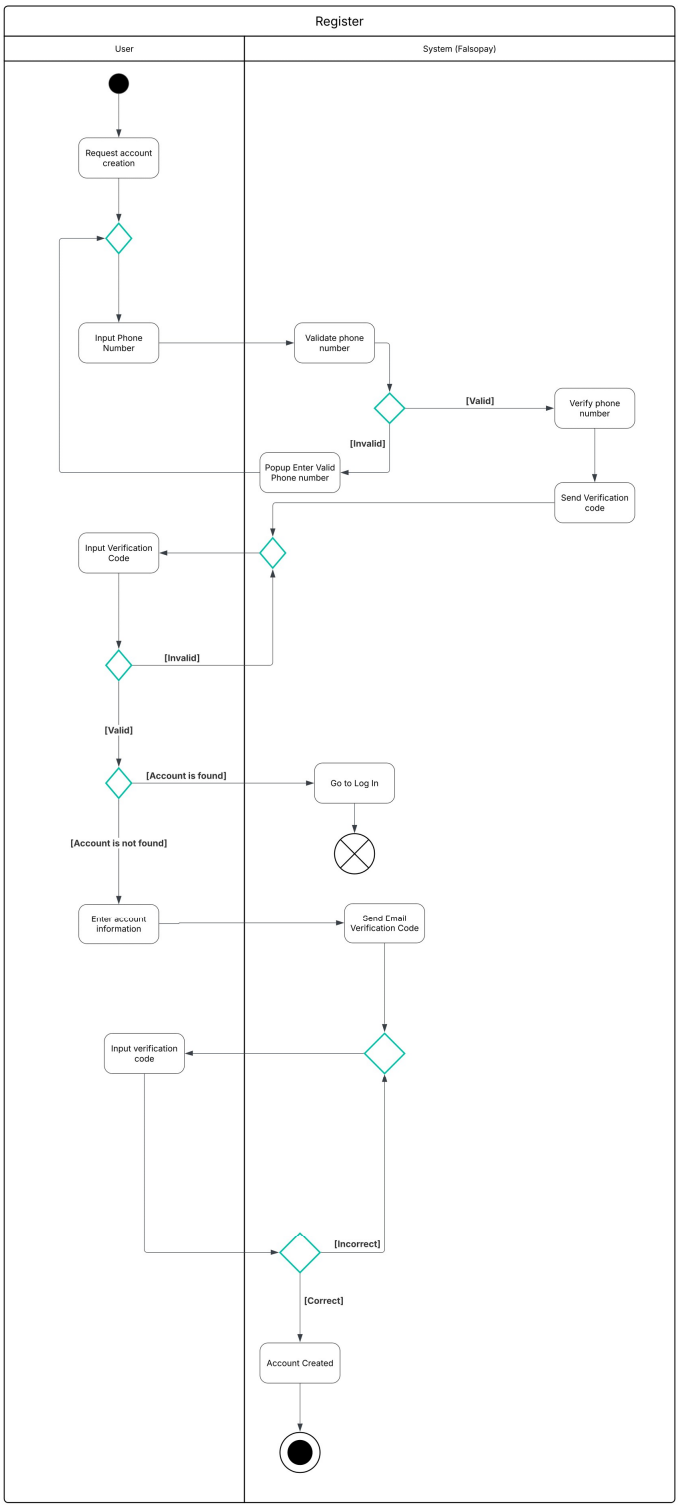
f) Activity Diagram

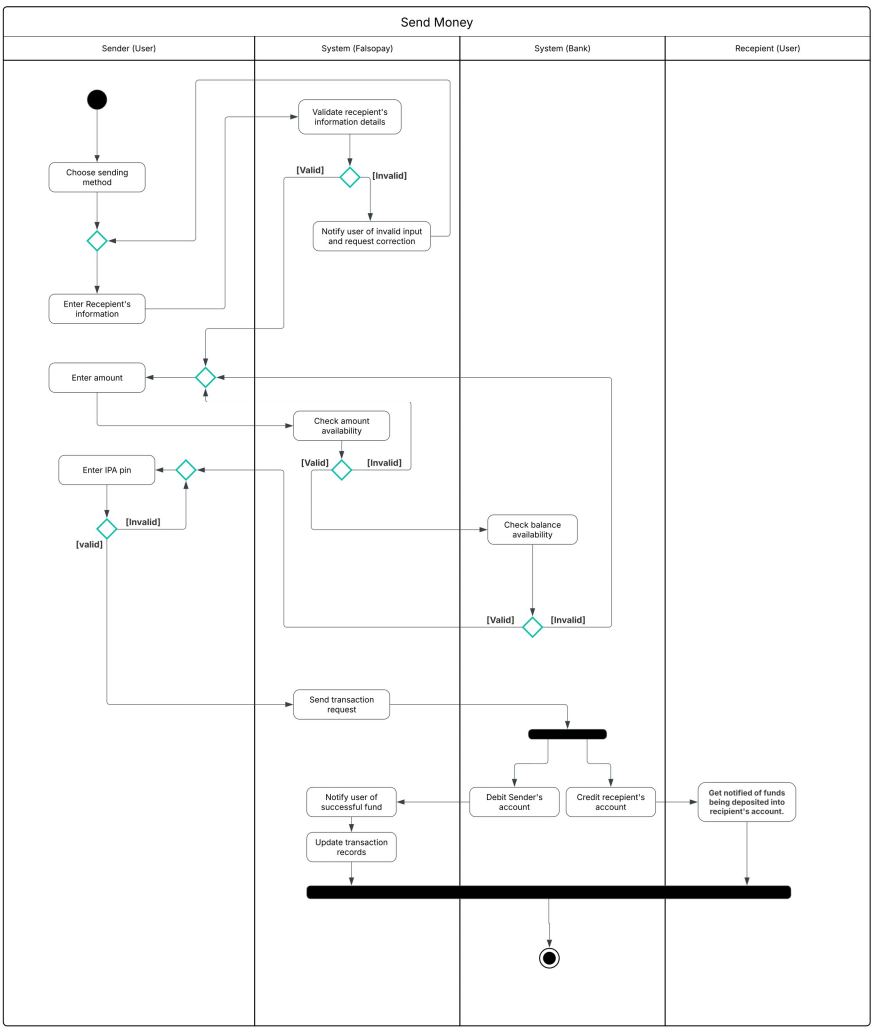



Edit Profile (personal)
Sender (User)
System (Falsopay)
Edit first name and last name
Edit Email
Send verification code on Email
Input verification code
[Invalid]
Notify user to re-enter code again
Notify user for updating his profile successfully
[Valid]


Link Bank Account
Sender (User)
System (Falsopay)
System (Bank)
Link Bank Account
Choose Bank
Input Card Number
Validate Length
Invalid Card Number
[Invalid]
[Valid]
Enter Pin
Request Account verification
Verify Bank Account
Input verification code
Send Verification Code
[Valid]
[Invalid]
Bank Account Not Found
[Invalid]
[Valid]
Notify User for linking bank account


Log In
User
System (Falsopay)
Enter Phone Number
Send Verification Code
Input verification code
[Valid]
[Registered]
[Not Registered]
[Invalid]
[Linked Accounts]
[No Linked Accounts]
Delete account
Return to Register
Enter IPA Address
Verify IPA Address
[Incorrect]
[Correct]
User login successfuly


Register
User
System (Falsopay)
Request account creation
Input Phone Number
Validate phone number
[Valid]
Verify phone number
[Invalid]
Popup Enter Valid Phone number
Send Verification code
Input Verification Code
[Invalid]
[Valid]
[Account is found]
Go to Log In
[Account is not found]
Enter account information
Send Email Verification Code
Input verification code
[Incorrect]
[Correct]
Account Created



f) Database Specification