Software Requirements and Design Document

for

ServEase

Prepared by

Minahil Ali
Haleema Tahir
Jawairia Waseem

24/11/2024

Table of Contents

1. Introduction

1.1 Purpose

This document outlines the software requirements specification (SRS) for the **servEase** application, which provides on-demand household management and personal services. This document covers the full scope of the **servEase** application, detailing the features, functionalities, and interactions of its components. The software's release version is **1.0**, and it is designed to facilitate the management of services like babysitting, laundry, tutoring, home cleaning, and more.

1.2 Product Scope

The **servEase** application allows users to request, book, and manage a variety of household services. The system connects customers with service providers based on service type, and availability. The application provides functionalities for both customers and service providers to interact, manage bookings, and make payments. Additionally, it allows admins to manage users and services The **servEase** platform is expected to improve user convenience, streamline household management, and provide a seamless experience for service requests and fulfillment.

1.3 Title

servEase: On-Demand Home Services App

This project aims to provide a comprehensive solution for managing household services. **servEase** allows users to request services like babysitting, laundry, tutoring, and home cleaning through a mobile application, connecting them with service providers available in their local area. The system is designed to ensure a smooth, user-friendly experience while promoting transparency and trust between service providers and customers.

1.4 Objectives

- To provide a platform that enables customers to easily browse, book, and pay for a variety of household services.
- To offer service providers the ability to manage their profiles, availability, and bookings.
- To enable the admin to manage users, services, and service requests, ensuring smooth operations.
- To facilitate efficient payment processing and feedback collection for each service rendered.
- To develop a scalable and reliable backend infrastructure to handle bookings, payments, and user management.

1.5 Problem Statement

The current domestic help market is fragmented, with many households struggling to find reliable and trustworthy service providers for their day-to-day needs. While some individuals may rely on traditional agencies or word of mouth, the process is time-consuming, inefficient, and often lacks transparency in pricing and service quality. The lack of a central platform that aggregates and manages these services can result in inconsistent user experiences and a lack of trust between service providers and customers.

The **servEase** app addresses this issue by creating a seamless, easy-to-use platform where users can request services, book appointments, make payments, and review service providers all in one place. The platform also offers service providers the opportunity to grow their businesses by gaining access to a wider customer base. Furthermore, the application's admin functionality ensures that service providers are verified and that the service process is smooth and transparent for both customers and providers.

In terms of **feasibility**, the project leverages existing technology and cloud-based infrastructure to ensure scalability and security. The app is designed to be compatible with mobile platforms (iOS and Android) and provides real-time updates for both users and service providers. With proper implementation, **servEase** can cater to diverse user needs while offering a robust, efficient solution to the fragmented domestic help service market.

2. Overall Description

2.1 Product Perspective

servEase is a new, self-contained product designed to meet the growing demand for efficient, on-demand household services. It is not a replacement for any existing system but a fresh approach to managing domestic help needs through technology. **servEase** serves as an intermediary platform between customers and service providers, offering a centralized, user-friendly interface for both parties to interact seamlessly. The application is designed to be scalable and accessible to a broad range of users, from individuals seeking domestic help to service providers looking to expand their business.

This product is designed as a mobile app that functions as both a service booking platform and a marketplace for household services, making it easier for customers to find qualified, available service providers nearby. **servEase** integrates several components, including user registration, service management, booking, payments, and feedback systems. The admin component manages the overall platform, including user roles (customer, service provider, and admin) and service offerings.

The system interacts with external payment gateways for processing secure transactions, and the user interface (UI) is built to be intuitive for non-technical users. The app will run on iOS and

Android platforms, leveraging cloud-based storage for scalability and reliability. Below is a simple diagram showing the major components of the system and their interactions:

- **Customer**: Requests services, makes bookings, provides feedback.
- **Service Provider**: Receives service requests, manages availability, and updates their profile.
- **Admin**: Oversees the operations, manages users, services, and service requests.

2.2 Product Functions

servEase must perform several key functions to ensure that customers can easily book services and service providers can manage their offerings. The major functions of the product include:

User Registration and Login

- Allows customers, service providers, and admins to register an account.
- Enables secure login and authentication to access their respective functionalities.

Service Browsing and Booking

- Customers can browse through available services, check service provider profiles, and make bookings.
- The system must allow customers to select service types or by name and date/time.

Service Provider Management

- Service providers can create, update, and manage their profiles.
- They can update their availability for various services and accept or reject bookings.

Admin Panel

- Admins can manage the platform by overseeing user activities, handling disputes, verifying service providers, and ensuring the overall integrity of the platform.
- Admins can also monitor and adjust service offerings as required.

Payment Processing

- Customers can make secure payments through integrated payment gateways.
- The system must handle transaction processing, including invoicing and receipt generation.

Feedback and Rating System

- Customers can rate and provide feedback on services received.

2.3 List of Use Cases

1. Register Account
2. Login/Logout
3. Browse Services (Customer)
4. Book Service
5. Cancel Booking (By Customer)
6. Make Payment
7. Provide Feedback/Rate Service
8. View Bookings/History
9. Update Profile
10. Manage Services
11. Manage Users
12. Accept/Reject Request
13. Offer services by service provider

2.4 Extended Use Cases

Use case name	Register Account
Scope	ServEase
Level	User Goal
Primary actor	Customer, Service Provider
Stakeholders and interests	- **Customer**: Wants to create an account to access services. - **Service Provider**: Wants to create an account to offer services. - **Admin**: Wants accurate user information for account management and service monitoring.
Preconditions	- User has internet access. - The user does not already have an account.

Postconditions	- New account is created and stored in the system database. - User is logged in upon successful registration..	
Main success scenario	1. The user selects "Register" on the app. 3. The user provides name, email, password, etc. and submits.	2. The system displays the registration form for input. 4. The system validates the inputs. 5. The system creates the account and logs the user in.
Extensions	3a. The user enters invalid information.	3a1. The system prompts the user to correct the information. 5a. The system detects that the email is already registered. 5a1. The system prompts the user to log in instead of registering.

Use case name	Login/Logout	
Scope	ServEase	
Level	User Goal	
Primary actor	Customer, Service Provider, Admin	
Stakeholders and interests	- **Customer**: Wants to securely log in to access services. - **Service Provider**: Wants to log in to manage service requests. - **Admin**: Monitors user access and ensures the security of accounts.	
Preconditions	- User has a registered account. - User has internet access.	
Postconditions	- User is logged in or logged out of the system.	
Main success scenario	1. The user selects "Login" from the home screen. 3. The user enters credentials and submits.	2. The system prompts for the username and password. 4. The system validates credentials and grants access if correct.

		5. The user is logged into the system.
Extensions	3a. The user enters incorrect credentials. 4a. The user attempts multiple failed logins.	3a1. The system displays an error and prompts for re-entry. 4a1. The system locks the account and requests a password reset.

Use case name	Browse Services	
Scope	ServEase	
Level	User Goal	
Primary actor	Customer	
Stakeholders and interests	- **Customer**: Wants to explore available services in their area. - **Service Provider**: Wants their services to be easily discoverable by customers. - **Admin**: Ensures the list of services is accurate and up to date.	
Preconditions	- Customer is logged in. - Services are available in the system.	
Postconditions	- Customer is able to view a list of services. - Customer selects a service.	
Main success scenario	1. The customer selects "Browse Services". 3. The customer filters or searches for services. 5. The customer selects a service to view details.	2. The system displays the list of available services. 4. The system updates the displayed services based on the search/filter. 6. The system displays detailed service information.
Extensions	2a. No services are available in the region.	2a1. The system notifies the customer of unavailability.

Use Case Name:

Book Service

Scope
ServEase

Level
User-goal level

Primary Actor
Customer

Stakeholders and Interests

1. **Customer**:
    - Wants to easily book a service, view service provider details, ensure secure payment, and receive timely service.
2. **Service Provider**:
    - Wants to be promptly notified about bookings, receive clear instructions, and ensure timely payment.
3. **Admin**:
    - Manages bookings, ensures smooth platform operations, verifies provider availability, and resolves disputes if needed.
4. **Payment Gateway**:
    - Ensures secure payment processing, fraud prevention, and smooth transactions for both customer and provider.

Precondition:
The customer is registered and logged in on the app.
The service provider is available and registered on the platform.
The customer has a valid payment method linked to their account.
The system and payment gateway are operational.

Postcondition:
The service is successfully booked, payment is processed, and both customer and service provider receive confirmation notifications.
The provider is prepared to deliver the service at the agreed time.

Main Success Scenario:

Customer, Service provider	System
1. The customer opens the app and navigates to the "Book Service" section.	3. The system displays a list of available service providers, along with ratings, pricing, and availability.
2. The customer selects the desired service (e.g., babysitting) from a list of available services.	6. The system prompts the customer to confirm the booking and initiate payment.
4. The customer selects a service provider and inputs the details (date, time, location, and any specific instructions).	8.The system processes the payment via the payment gateway.
5. The customer reviews the summary, including the booking request details and pricing.	9. Upon successful payment, the system confirms the booking and sends notifications to both the customer and the service provider.
7. The customer confirms and authorizes payment through their linked payment method.	
10. The service provider acknowledges the booking request and prepares to fulfill the service.	

Extensions

1: Service Provider Not Available
- 4a. If no service provider is available for the selected time, the system suggests alternative dates or providers.
- 4b. The customer chooses one of the suggested options, or cancels the booking.

2: Payment Fails
- 8a. If the payment fails due to insufficient funds or technical issues, the system prompts the customer to use an alternative payment method or retry the transaction.
- 8b. The customer updates the payment information or cancels the booking attempt.

3: Service Provider Declines Booking Request

- 9a. If the service provider declines the booking, the system notifies the customer and suggests alternative providers or times.
- 9b. The customer chooses to either accept the alternative or cancel the booking.

4: Booking Cancellation by Customer
- 10a. The customer may cancel the booking prior to the service date. The system processes the cancellation and initiates a refund (if applicable) according to the platform's policies.

5: Admin Override
- 10b. The admin may step in to resolve disputes, reassign bookings, or make adjustments if required, ensuring smooth customer and provider satisfaction.

Use Case Name:
Cancel Booking (By Customer)

Scope:
ServEase

Level:
User-goal level

Primary Actor:
Customer

Stakeholders and Interests:
1. Customer:
Wants the ability to cancel a service booking easily, receive a refund (if applicable), and avoid being charged for services not rendered.

2. Service Provider:
Wants to be informed of cancellations promptly to avoid unnecessary scheduling conflicts and missed income.

3. Admin:
Wants to ensure smooth processing of cancellations, manage refund policies, and maintain the app’s integrity in handling disputes or complaints.

4. Payment Gateway:
Handles the processing of refunds, ensuring compliance with payment policies, and avoiding fraudulent refund requests.

Preconditions

1. The customer has already booked a service via the app.
2. The service booking is still eligible for cancellation as per the platform's policies (e.g., within the allowed time window).
3. The customer is logged into their account on the app.

Postconditions:

The service is successfully canceled, the customer is notified, and the system initiates a refund (if applicable) based on the cancellation policy. The service provider is also notified.

Main Success Scenario:

Customer	System
1. The customer logs into the app and navigates to the "My Bookings" section.	2. The system displays the list of active bookings, including details like date, time, and service provider information.
3. The customer selects the booking they wish to cancel.	4. The system checks if the booking is still eligible for cancellation based on the platform's policies (e.g., time window for free cancellation).
6. The customer confirms the cancellation.	5. The system prompts the customer to confirm the cancellation.
	7. The system updates the booking status to "Canceled" and sends notifications to both the customer and the service provider.
	8. The payment gateway confirms the refund, and the system updates the customer's account with the refund details.

Extensions

1. Booking Not Eligible for Cancellation

- 4a. If the booking is not eligible for cancellation (e.g., it's too close to the service date), the system displays a message explaining the reason and offers the option to contact support.
- 4b. The customer may choose to contact the support team or exit the process.

2. Admin Override
- 10a. The admin may intervene to approve a late cancellation or refund in special circumstances (e.g., a customer complaint or dispute).
- 10b. The system reflects the admin's decision, and both the customer and service provider are notified of the resolution.

Use Case Name:
Make Payment

Scope:
ServEase

Level:
User-goal level

Primary Actor:
Customer

Stakeholders and Interests:
1. Customer:
Wants to securely pay for services using a preferred payment method and ensure the payment is processed promptly.

2. Payment Gateway:
Facilitates the payment, ensures secure transactions, and handles the transfer of funds between the customer and the service provider while protecting against fraud.

3. Admin:
Oversees payment processes, ensures smooth transactions, and handles any disputes or payment failures.

4. Service Provider:
Wants to be paid promptly for services provided, ensuring the payment is processed before or after service delivery as per policy.

Preconditions:

1. The customer has an active service booking.

2. The customer has a valid payment method linked to their account.
3. The system is integrated with a payment gateway that is operational.
4. The service price has been agreed upon by both the customer and the service provider.

Postconditions:

The payment is successfully processed, the customer and service provider are notified, and the service is confirmed or completed based on the payment.

Main Success Scenario:

Customer	System
1. The customer reviews the service details, including the total cost, before proceeding to payment.	3. The system communicates with the payment gateway to process the transaction.
2. The customer initiates the payment by selecting a linked payment method (e.g., credit card, debit card, digital wallet).	4. The payment gateway verifies the payment details and completes the transaction.
	5. Upon successful payment, the payment gateway sends confirmation to the system.
	6. The system updates the booking status to "Paid" and sends notifications to both the customer and the service provider.

Extensions

1.Insufficient Funds

- 4a. If the payment fails due to insufficient funds or an expired payment method, the payment gateway informs the system.
- 4b. The system prompts the customer to update their payment information or use an alternative payment method.
- 4c. The customer provides new payment details, and the system retries the payment.

2. Admin Override
- 8a. If there are disputes or issues with the payment, the admin can override the system to approve or reverse a payment manually.
- 8b. The system reflects the admin's decision, and all parties involved are notified.

Use Case Name:
Provide Feedback/Rate Service

Scope:
ServEase

Level:
User-goal level

Primary Actor:
Customer

Stakeholders and Interests:

1. Customer:
Wants to provide feedback on their experience with the service, including any concerns or praise, and rate the service provider based on satisfaction.

2. Service Provider:
Wants to receive feedback to improve service quality and maintain a positive rating that could influence future bookings.

3. Admin:
Wants to monitor service quality, address any customer or provider complaints, and use feedback to manage service providers' performance.

4. Future Customers:
Want to read accurate and helpful reviews/ratings to make informed decisions about booking services in the future.

Preconditions:

1. The customer has completed a service booking.
2. The system prompts the customer to provide feedback after the service is completed, or the customer initiates the process from their account.
3. The customer is logged into their account.

Postconditions:

The feedback and rating are successfully submitted and saved in the system. The service provider is notified, and the feedback is visible to other users (if applicable).

Main Success Scenario:

Customer	System
2. The customer navigates to the "Rate Service" section from the app or via the prompt.	1. The system sends a prompt or notification asking the user to rate the service after its completion.
4. The customer selects a star rating (e.g., 1-5 stars) to indicate their level of satisfaction.	3. The system displays the relevant service details and the name of the service provider.
6. The customer submits the rating and feedback.	5. The system prompts the customer to provide written feedback in a text box (optional).
9. The customer receives a confirmation of the successful submission.	7. The system saves the rating and feedback and updates the service provider’s profile with the new rating.
	8. The system notifies the service provider about the feedback.

Extensions

1. Customer Skips Feedback
   - 5a. If the customer chooses to skip writing feedback and only provides a star rating, the system still records and updates the service provider’s profile with the rating.

Use Case Name:
View Bookings/History

Scope:
ServEase

Level:
User-goal level

Primary Actor:
Customer

Stakeholders and Interests:

1. Customer:
   Wants to easily access and view details of upcoming and past bookings, including service information, dates, payment status, and provider details.

2. Service Provider:
   Wants to ensure that customers have access to service details to avoid miscommunication about upcoming appointments and past services.

3. Admin:
   Wants to ensure that all booking records are accurate and accessible to the customer for potential disputes, reference, or audits.

Preconditions:

1. The customer has booked one or more services via the app.
2. The customer is logged into their account.

Postconditions:

The customer successfully views their booking history, including details of completed, canceled, and upcoming services.

Main Success Scenario

Customer	System

1. The customer logs into the app and navigates to the "My Bookings" or "Booking History" section.	2. The system displays a list of upcoming bookings, along with basic details such as service type, date, time, and provider information.
3. The customer can select an upcoming booking to view additional details (e.g., service instructions, payment status, location).	5. For each past booking, the system displays the service details, date, time, service provider, and status (completed or canceled).
4. The customer can switch to a "Past Bookings" tab to view previously completed or canceled services.	7. The system also provides an option to rebook a past service or contact the service provider from the history page.
6. The customer can view payment details for each booking, including amounts paid, payment method, and transaction ID.	
8. The customer exits the booking history after reviewing the desired information.	

Extensions

1. No Upcoming or Past Bookings
   - 2a. If the customer has no upcoming or past bookings, the system displays a message indicating that there are no records.
   - 2b. The customer is prompted to book a new service from the main booking page.

2. Service no Longer Available
   - 7a The customer may switch to another service.
   - 7b. The customer may retry, or contact support if the issue persists.

3. Filter and Sort Options
   - 4a. The system may allow the customer to filter bookings by status (e.g., upcoming, completed, canceled) or sort by date.
   - 4b. The customer selects the desired filter or sort option, and the system refreshes the list accordingly.

4. Admin Intervention

- 5a. If the customer notices discrepancies or missing records in their booking history, they may contact the admin.
- 5b. The admin investigates the issue, makes any necessary corrections, and updates the customer.

5. View Booking Receipts
- 6a. The system provides an option for the customer to download or view receipts for each completed booking.
- 6b. The customer selects this option, and the system generates a detailed receipt, including payment, service, and provider details.

MINAHIL ALI (22i-0849)

Use case name	Update Profile	
Scope	ServEase	
Level	User Goal	
Primary actor	Customer, Service Provider	
Stakeholders and interests	**Customer**: Wants to update personal information like contact details, preferences. **Service Provider**: Wants to update availability, service details, and personal information. **Admin**: Ensures the profile updates are accurate and valid.	
Preconditions	User must be logged in and have a valid profile.	
Postconditions	Profile information is updated in the system and reflects any new changes.	
Main success scenario	1. User logs in to their account. 2. User navigates to the "Profile" section. 4. User makes necessary changes and submits the update. 6. User receives confirmation that the profile is updated	3. System displays current profile information. 5. System validates the changes and saves the new information.
Extensions	4a. Invalid information (e.g., invalid email): 5a. If system update fails due to server error:	4a.1. System prompts the user to correct the errors. 5a.1. System notifies the user and suggests trying again later.

Use case name	Manage Services	
Scope	ServEase	
Level	User Goal	
Primary actor	Admin	
Stakeholders and interests	**Admin**: Needs to manage service listings, including adding, removing, or modifying services. **Service Providers**: Want their services to be accurately listed. **Customers**: Want reliable services to be available in the system.	
Preconditions	Admin must be logged in with appropriate permissions.	
Postconditions	Service listings are updated in the system.	
Main success scenario	1. Admin logs in to the system. 2. Admin navigates to the "Manage Services" section. 3. Admin views the current list of services. 4. Admin selects an option to add, remove, or modify a service.	5. System applies changes and updates the service listings. 6. System confirms the success of the action to the admin.
Extensions	4a. If the admin tries to add an invalid service (e.g., missing details): 5a. If the update fails (e.g., server issues):	4a.1. System prompts admin to complete all required fields. 5a.1. System informs admin of the failure and suggests trying again later.

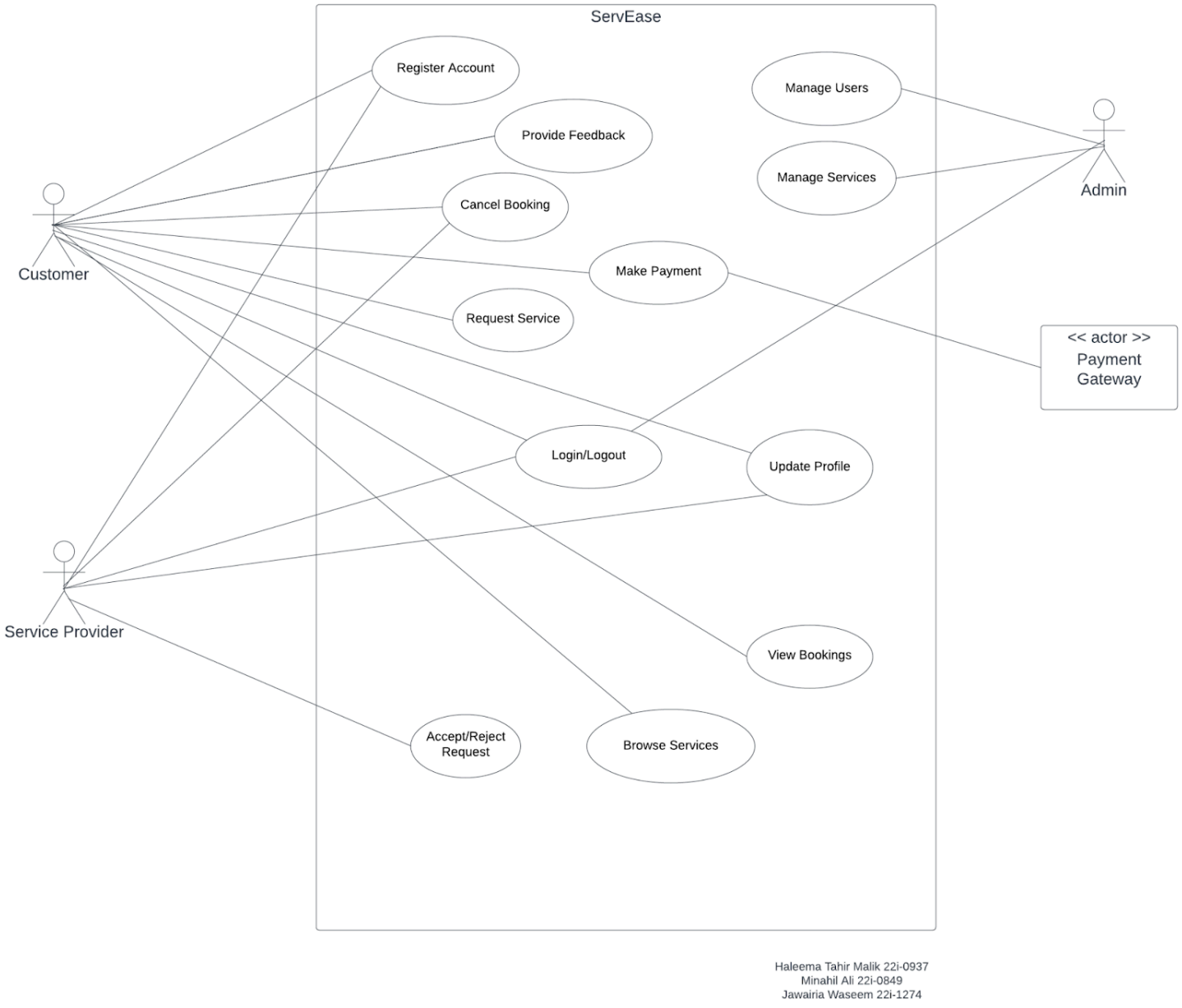
Use case name	Manage Users	
Scope	ServEase	
Level	User Goal	
Primary actor	Admin	
Stakeholders and interests	**Admin**: Needs to manage user accounts, including adding, removing, or modifying user information. **Users**: Want their accounts to be accurately maintained and secure.	
Preconditions	Admin must be logged in with appropriate permissions.	
Postconditions	User accounts are updated in the system.	
Main success scenario	1. Admin logs in to the system. 2. Admin navigates to the "Manage Users" section. 3. Admin views a list of all registered users. 4. Admin selects an option to add, remove, or modify a user. 6. Admin receives confirmation that the action was successful.	5. System applies changes and updates the user information.
Extensions	4a. Admin tries to modify an invalid user account (e.g., missing information): 4b. admin attempts to delete a user who has pending transactions 5a. If the update fails due to server error:	4a.1. System prompts admin to fill in the necessary fields. 4b.1. System notifies that user has active dependencies 5a.1. System informs admin of the failure and suggests trying again later.

(integrated Receive Service Request)

Use case name	Accept/Reject Request	
Scope	ServEase	
Level	User Goal	
Primary actor	Service Provider	
Stakeholders and interests	**Service Provider:** Wants to review new service requests and accept them based on availability or reject them if they are unable to fulfill them. **Customer:** Expects to receive confirmation or rejection of the service request in a timely manner. **Admin:** Oversees service requests and provider responses for quality and reliability of service.	
Preconditions	A service request must already be submitted by the customer and delivered to the service provider's account. The provider is logged in and able to access pending requests.	
Postconditions	The service request status is updated based on the provider's decision: accepted (and scheduled) or rejected (with notification sent to the customer).	
Main success scenario	1. Service provider logs in and navigates to the pending requests section. 2. Service provider views the details of the new service request, including the type of service, location, time, and any additional notes from the customer. 3. Service provider evaluates their availability and decides whether to accept or reject the request. 4. Service provider decides to accept or reject the request.	5. If the provider accepts the request, the system confirms the booking and notifies the customer of the acceptance. 6. If the provider rejects the request, the system updates the request status and sends a notification to the customer informing them of the rejection.
Extensions	2a. If the request details are	

	incomplete or unclear: 4a. If the provider accepts but needs to suggest a new time::	2a.1. Provider requests additional information from the customer through the app messaging system. 4a.1. Provider suggests an alternative time, and the customer is notified of the proposed time change.

2.5 Use Case Diagram



3. Other Nonfunctional Requirements

3.1 Performance Requirements

The **servEase** app must perform efficiently under normal operating conditions to ensure a seamless user experience. The system should be designed to handle a high volume of concurrent users,

including customers and service providers, without performance degradation. Specific performance requirements include:

1. **Response Time**:
    - The system should respond to user actions (e.g., searching for services, submitting a booking) within 2 seconds under normal conditions.
    - Complex queries (e.g., searching for services with specific filters) should return results within 5 seconds.
2. **System Availability**:
    - The app must be available 99.9% of the time, ensuring minimal downtime. Downtime should not exceed 8 hours annually.
3. **Scalability**:
    - The system should scale horizontally, ensuring that as the number of users increases, the system can handle additional traffic without performance loss. This scalability should support at least 10,000 concurrent users.
4. **Data Synchronization**:
    - The system should sync data between mobile devices and the backend in real-time. Any updates made by the customer or service provider (e.g., booking or service profile updates) should be reflected immediately for all relevant users.
5. **Transaction Throughput**:
    - The system should be able to process up to 1,000 transactions per minute during peak usage, including bookings and payments.

These performance requirements will guide developers in choosing appropriate technologies, database architectures, and design patterns to ensure the system operates efficiently even as user numbers and data complexity grow.

3.2 Safety Requirements

The **servEase** app operates in a context where customer safety is a priority, particularly when interacting with service providers. To mitigate any potential harm or damage during service provision, the following safety requirements apply:

1. **Service Provider Verification**:
    - All service providers must undergo background checks and verification before being allowed to register on the platform. This includes checking for criminal records, ensuring they are qualified to provide the services they list, and confirming their identity.
2. **Insurance and Liability**:
    - Service providers must carry liability insurance to protect customers in case of damage to property or personal injury during service delivery. The system will require proof of insurance before a service provider is allowed to take bookings.
3. **Safety Guidelines**:

- The platform will provide safety guidelines to both customers and service providers to minimize risks during service provision, such as maintaining safe distances and using appropriate equipment.

4. **Emergency Contact**:
   - The app will include a feature that allows customers to contact emergency services or a designated help center in case of an urgent issue during service provision.

By enforcing these safety measures, the system aims to ensure that users feel confident and protected while using the app.

3.3 Security Requirements

The **servEase** app will handle sensitive user data (e.g., payment information, personal addresses, booking history) and must meet high security and privacy standards:

User Authentication and Authorization:

Users (customers, service providers, and admins) must authenticate via secure login (using email, phone number, or third-party authentication, such as Google or Facebook).

Role-based access control (RBAC) will ensure that only authorized users can access certain features (e.g., admins can manage users, but customers and service providers cannot).

Data Encryption:

All sensitive data, including payment details and personal information, will be encrypted both at rest and in transit using AES-256 encryption for storage and SSL/TLS for communication.

Payment Security:

The app will integrate with PCI DSS-compliant payment gateways to handle transactions securely. Payment details will never be stored directly on the platform.

Privacy Protection:

The app will comply with data protection regulations such as GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act), ensuring that users have control over their data (e.g., rights to access, correction, and deletion).

Secure APIs:

The platform's APIs will be designed with security in mind, including authentication tokens and rate-limiting to protect against DDoS attacks and ensure secure access to external services.

3.4 Software Quality Attributes

The **servEase** app is expected to exhibit the following software quality attributes to ensure that the system is robust, adaptable, and user-friendly:

Usability:

The user interface must be intuitive and simple, with a goal of minimizing the learning curve for new users. The app should be usable without any technical knowledge, with clear instructions, tooltips, and responsive design.

Reliability:

The system should operate without critical bugs or crashes. It should be tested under various real-world conditions to ensure the platform remains stable even during peak usage.

Maintainability:

The app will be designed with modularity in mind, allowing for easy updates and bug fixes. The code will follow best practices for maintainable software, including clear documentation and unit testing.

Portability:

The platform should be deployable across different devices and operating systems, including iOS and Android. The app should also work across multiple screen sizes and orientations.

Interoperability:

The system must integrate with external services, such as payment gateways, mapping services (for location tracking), and email/SMS notification services.

Scalability:

The app should be scalable to handle increasing numbers of users, services, and transactions. This includes both horizontal and vertical scaling to ensure optimal performance as the user base grows.

3.5 Business Rules

Customer and Service Provider Roles:

- Only registered customers can book services. Customers must provide accurate personal and payment information during registration.
- Only verified service providers can offer services and must maintain an updated profile, including proof of their qualifications and insurance.

Booking Rules:

- Customers can only book services if the service provider is available at the requested time.
- A service provider can accept or reject bookings based on their availability and preferences.

Feedback and Ratings:

- Customers can only leave feedback after the service has been completed. Service providers can view their ratings and feedback but cannot delete or alter them.

3.6 Operating Environment

The **servEase** app will operate in the following environment:

1. **Hardware Platform**:
    - The app will be designed for mobile devices running on iOS and Android platforms. The backend will be cloud-based and should be accessible through internet-enabled devices.
2. **Operating System**:
    - The app will be compatible with iOS 11.0+ and Android 5.0+.
3. **Software Components**:
    - **Mobile App**: Developed using React Native for cross-platform compatibility, ensuring the app works on both iOS and Android.
    - **Backend Services**: The server-side application will be built using Node.js and Express, with a RESTful API for communication between the mobile app and the server.
    - **Database**: A relational database (e.g., PostgreSQL) will store user, service, booking, and payment data. The database will be hosted on a cloud provider like AWS or Google Cloud.
    - **Third-Party Services**: Payment gateway (e.g., Stripe or PayPal), email/SMS notification services (e.g., SendGrid, Twilio).

The **servEase** platform is designed to be scalable, secure, and compatible with modern mobile devices and cloud services, ensuring that it can accommodate the needs of customers, service providers, and admins alike.

3.7 User Interfaces

The **servEase** app features a user-friendly interface designed to streamline the interaction between customers, service providers, and admins. This section outlines the logical characteristics of each user interface (UI) in the system, focusing on common design standards, components, and functionality.

1. Customer Interface

- **Login/Signup Screen**:
  - **UI Components**:
    - **Username Field**: Input field for entering the email or phone number.
    - **Password Field**: Input field for entering the password.
    - **Login Button**: Button to log the customer into their account.
    - **Signup Link**: Link to create a new account if the customer does not have one.
    - **Forgot Password Link**: Link to reset the password.
    - **Standard Functionality**: If login fails, an error message is displayed, such as "Invalid username or password."
  - **Design Standards**:
    - Simple, minimalistic design.
    - Prominent call-to-action buttons (Login, Signup).
    - Error messages appear below the respective input fields.
    - Consistent font style with a modern, clean design.
- **Home Screen**:
  - **UI Components**:
    - **Service Search Bar**: A search bar to find services by name, location, or category.
    - **Service Categories**: Icons or dropdowns to select categories of services (e.g., Babysitting, Home Cleaning).
    - **Available Providers**: A list of service providers with their ratings and brief details.
    - **Profile Button**: To view and update customer profile.
    - **Navigation Bar**: Tabs for services, bookings, payments, and settings.
    - **Standard Buttons**: Filter button, search button.
  - **Design Standards**:
    - Clear sections with vertical scrolling.
    - Search bar at the top for easy access.
    - Each provider has a profile picture, a brief description, and rating information.
    - Use of bright, eye-catching colors for buttons to draw attention.
- **Booking Screen**:
  - **UI Components**:
    - **Provider Info**: Name, profile image, service details, ratings.
    - **Date & Time Picker**: Allows the customer to choose the booking date and time.
    - **Payment Information**: Section to add or choose payment methods.
    - **Book Button**: A button to confirm the booking and payment.
    - **Cancellation Policy**: A link or modal to view the cancellation terms before booking.
  - **Design Standards**:
    - Booking information should be laid out in a step-by-step manner.
    - Progress bar or steps indicator to show the current stage of booking.
    - Clear, large buttons to avoid errors during interaction.
- **Feedback and Rating Screen**:

    - **UI Components**:
        - **Rating Slider**: A slider or stars to rate the service provider.
        - **Comment Box**: Text field for writing comments or feedback.
        - **Submit Button**: To submit the feedback.
    - **Design Standards**:
        - Feedback form should appear once the service is completed.
        - Intuitive design with emphasis on ease of use for rating and commenting.

2. Service Provider Interface

- **Login/Signup Screen**:
    - Same as the customer interface, but with fields for service provider-specific details, like services offered, experience, and insurance status.
- **Dashboard Screen**:
    - **UI Components**:
        - **Service Listings**: Display of services currently being offered by the provider.
        - **Bookings List**: A list of upcoming, ongoing, and completed bookings.
        - **Profile Button**: To update personal details, including qualifications, experience, and services offered.
        - **Availability Calendar**: To set available dates and times for accepting bookings.
        - **Ratings & Reviews**: Display of received ratings and feedback from customers.
    - **Design Standards**:
        - Focus on functionality and organization of bookings and services.
        - Simple interface with clear tabs for service management, calendar, and ratings.
        - Notifications for new bookings should be prominent to avoid missed opportunities.
- **Booking Management Screen**:
    - **UI Components**:
        - **Customer Details**: Name, contact, and service request details.
        - **Accept/Reject Buttons**: To accept or reject booking requests.
        - **Booking Time & Date**: Clear display of the booking's scheduled time and date.
        - **Chat Option**: A chat feature for communication between customer and provider.
    - **Design Standards**:
        - Each booking should be clearly defined with easy-to-use action buttons (Accept, Reject).
        - Clean layout with emphasis on clarity of booking details.

3. Admin Interface

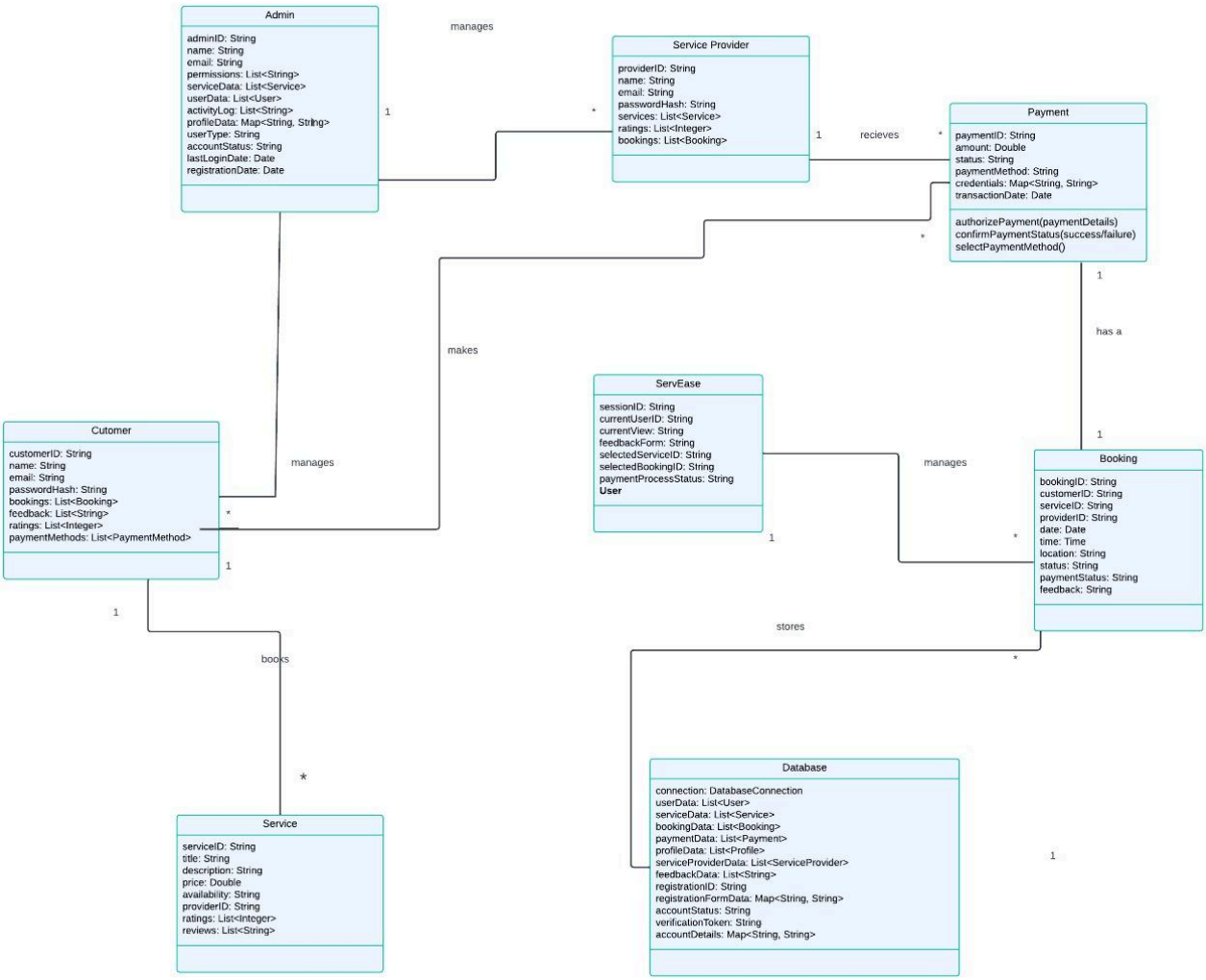
- **Login Screen**:
    - **UI Components**:
        - **Username and Password Fields**.

        - **Login Button**.
        - **Forgot Password Link**.
- **Admin Dashboard**:
    - **UI Components**:
        - **User Management**: A list of all customers and service providers with options to deactivate or activate accounts.
        - **Bookings Overview**: A summary of current, upcoming, and past bookings.
        - **Service Management**: Ability to view, approve, or reject service listings.
        - **Analytics Section**: Graphs and data on bookings, payments, user activity, etc.
        - **Notifications**: Notifications for any critical issues, like suspended accounts or pending service approvals.
    - **Design Standards**:
        - Data should be displayed in clear, organized tables or graphs for easy analysis.
        - Prominent action buttons for user account management and service approvals.
        - A navigation bar or sidebar for easy access to each section of the admin console.

4. General Interface Design Guidelines

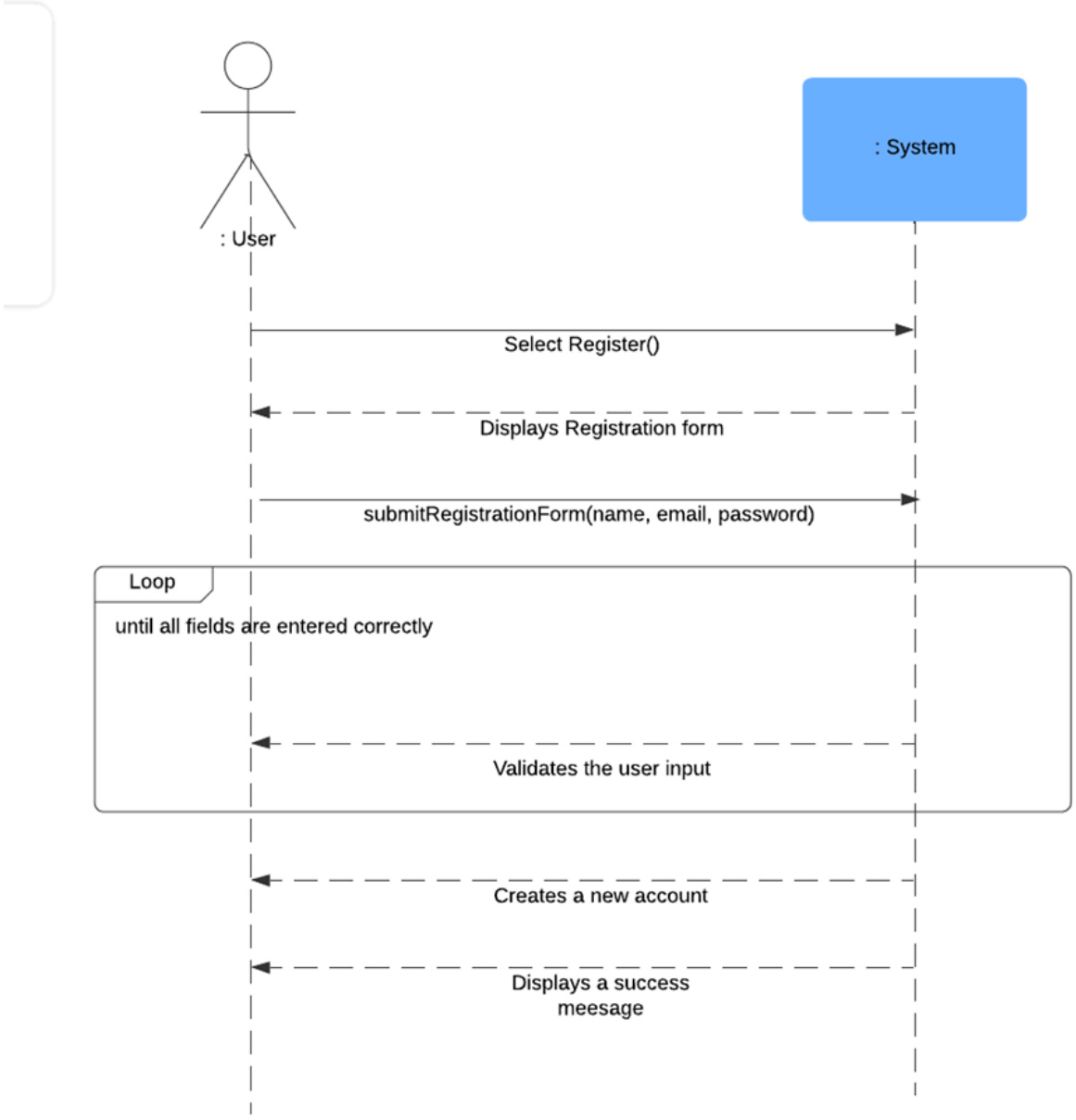
- **Consistency**:
    - All user interfaces will follow a consistent color scheme, font style, and layout principles to ensure that the user can easily transition between screens without confusion.
- **Accessibility**:
    - The design will meet accessibility standards, including text readability, alt text for images, color contrast for users with vision impairments, and keyboard navigability.
- **Help Section**:
    - A dedicated help icon/button will be present on every screen. It will lead to an FAQ page or customer support contact information.
- **Error Messages**:
    - Error messages will be shown in clear, concise language and placed near the field where the error occurred (e.g., incorrect login credentials, incomplete booking form).
    - All error messages will follow a color code (e.g., red for errors, green for success) and will be accompanied by a description of the problem and how to resolve it.
- **Mobile Responsiveness**:
    - All screens will be designed with a mobile-first approach, ensuring that the app is responsive across all screen sizes.

4. Domain Model



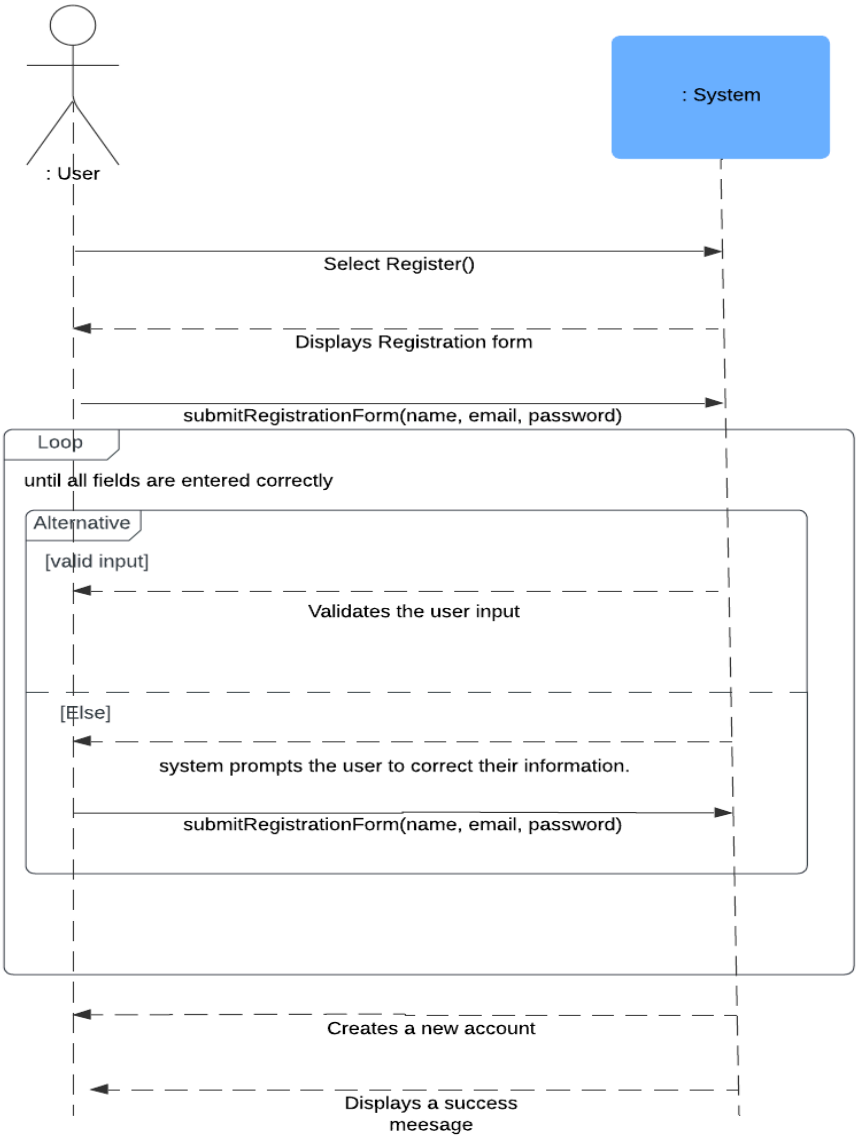
5. System Sequence Diagram

Main Success Scenario (MSS) - Register Account



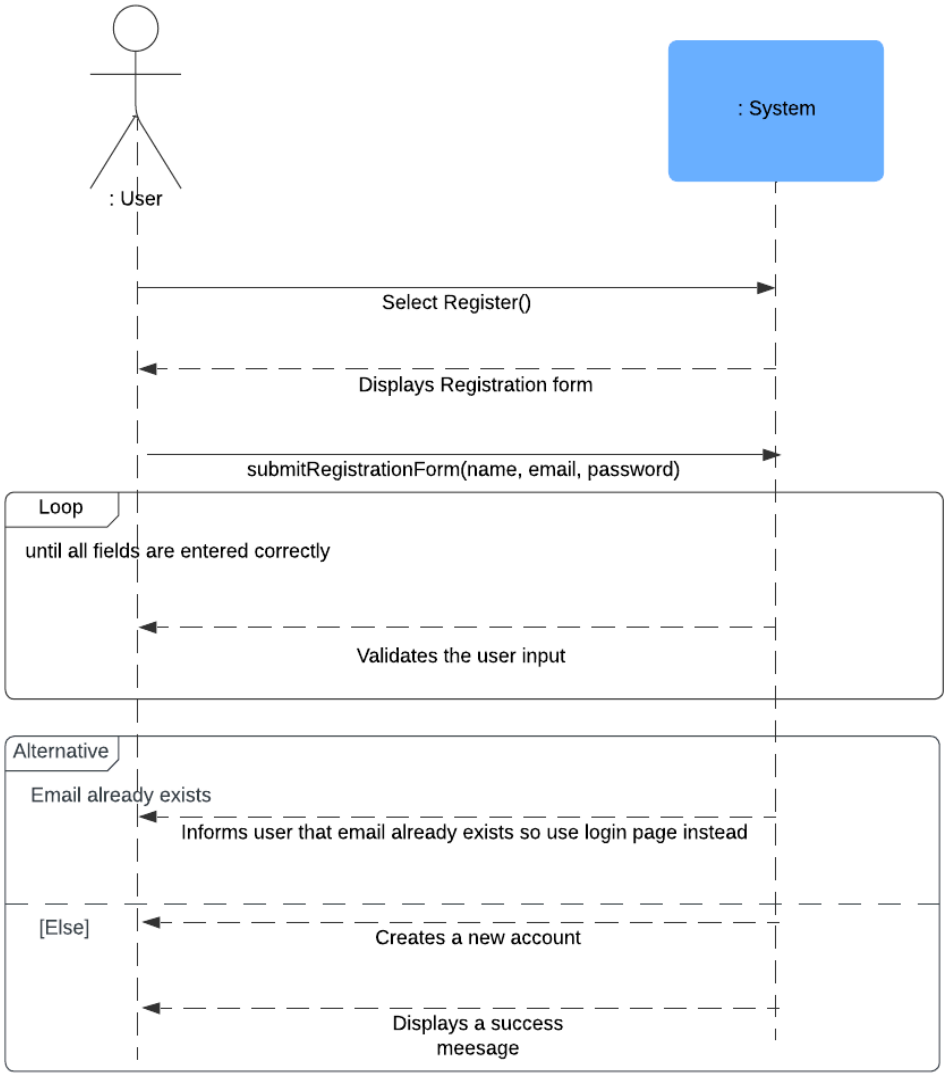
Alternative Scenario 3a - Invalid Information

Context: After the user submits the registration form in Step 3 of the MSS.

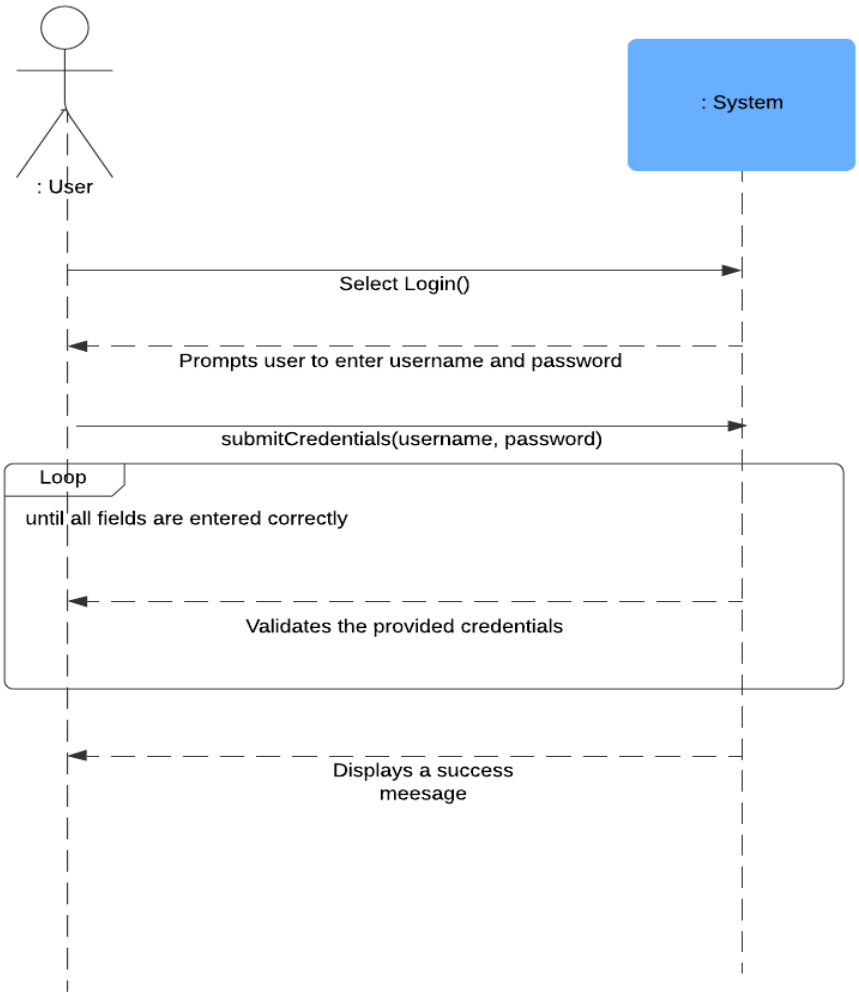


Alternative Scenario 5a - Email Already Registered

Context: After the system validates the input in Step 4 of the MSS.

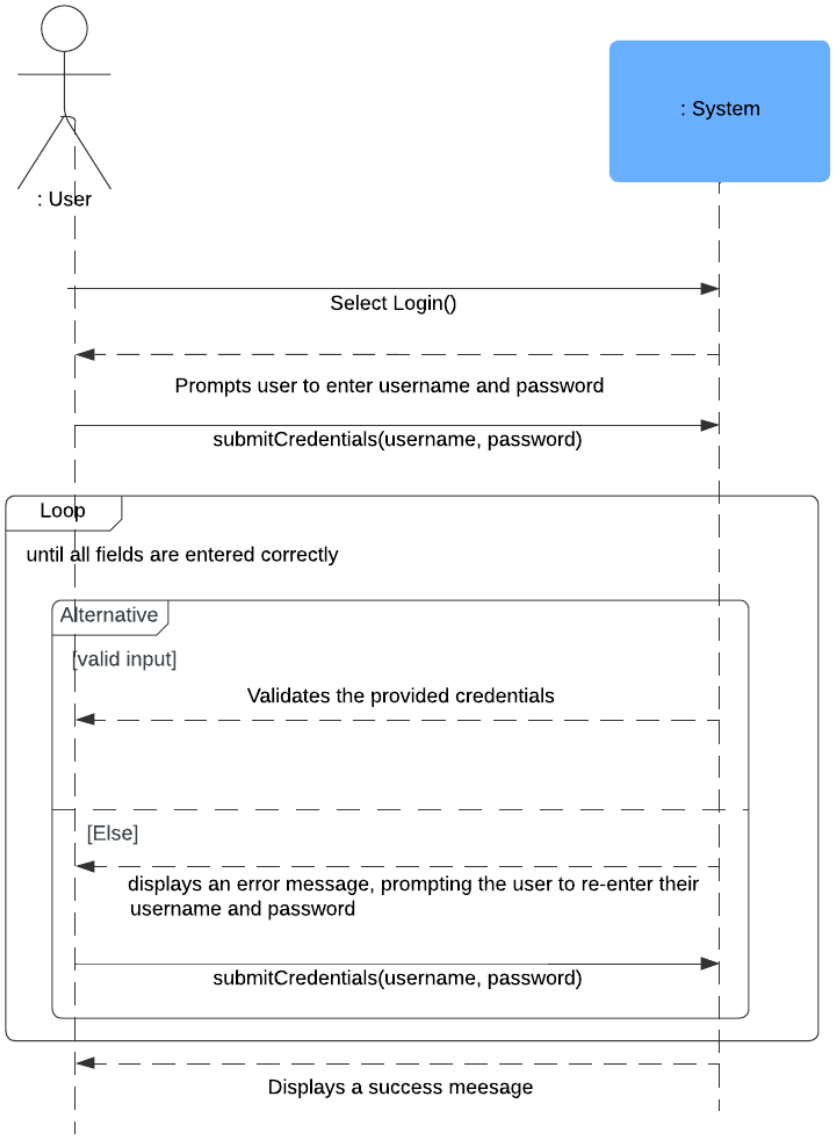


Main Success Scenario (MSS) - Login/Logout



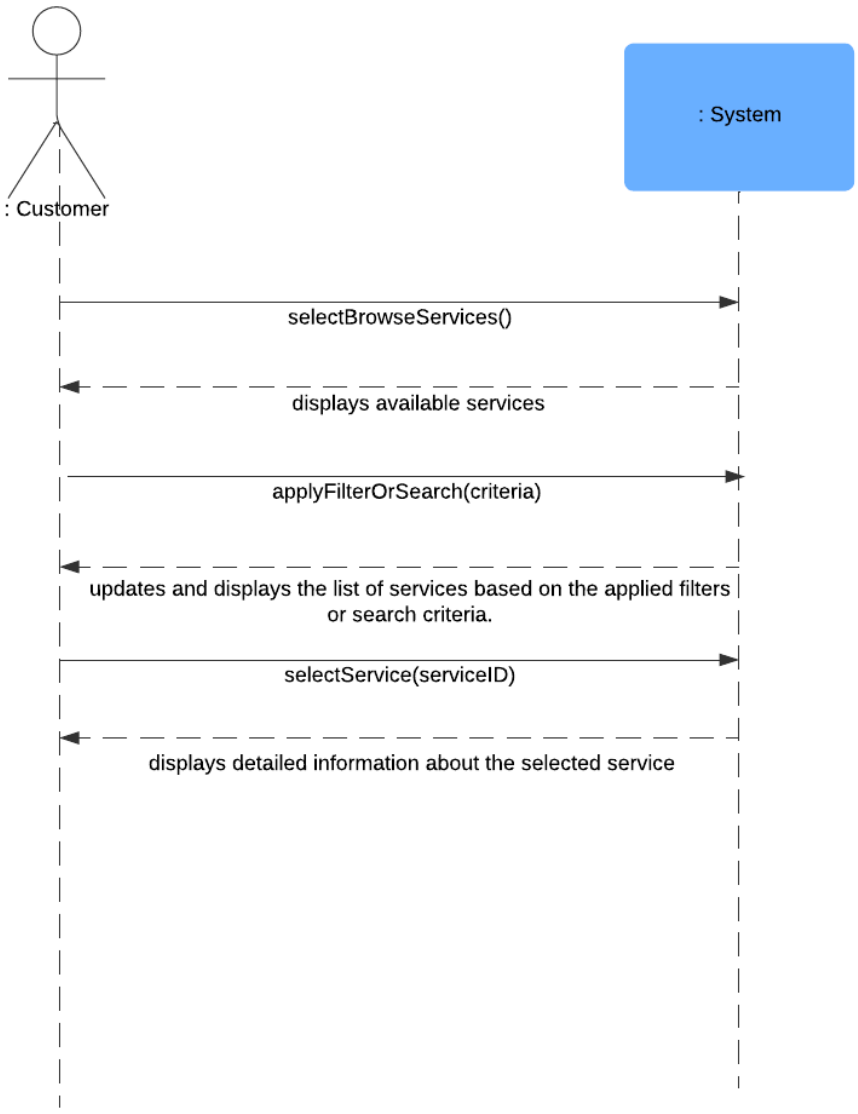
Alternative Scenario 3a - Incorrect Credentials

Context: After the user submits the credentials in Step 3 of the MSS.

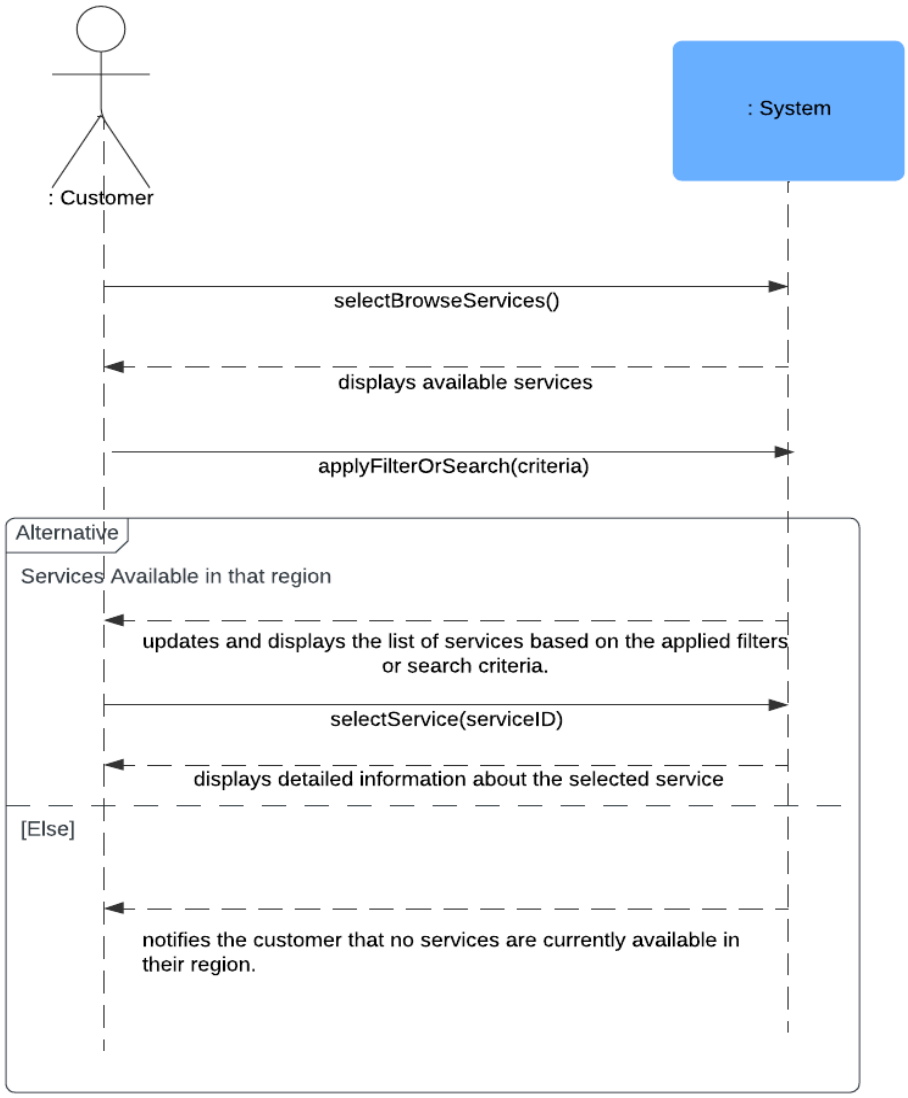


Use case 3: Browse Services

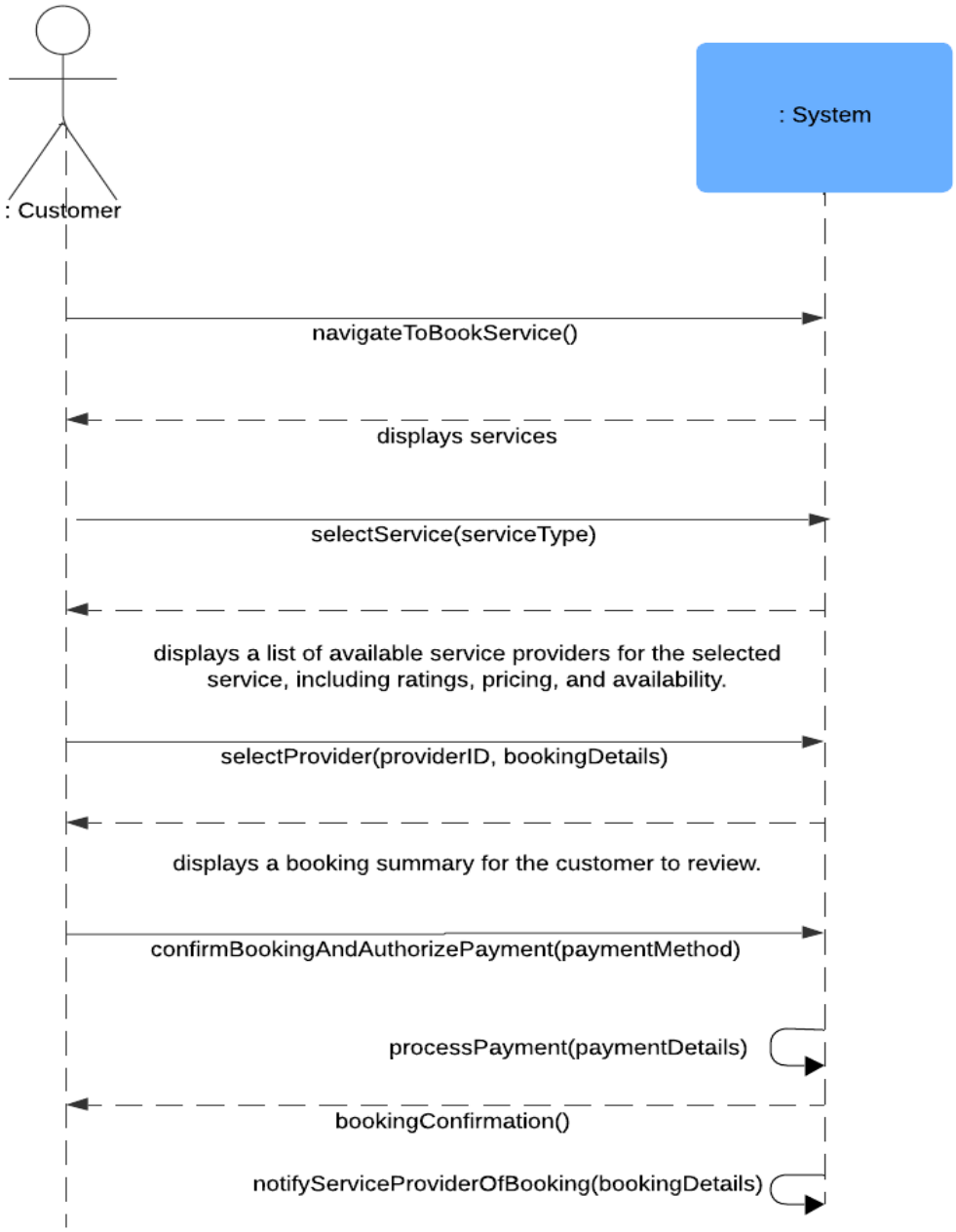
Main Success Scenario (MSS) - Browse Services



Context: After Step 2 in the MSS, if no services are available in the customer's region.



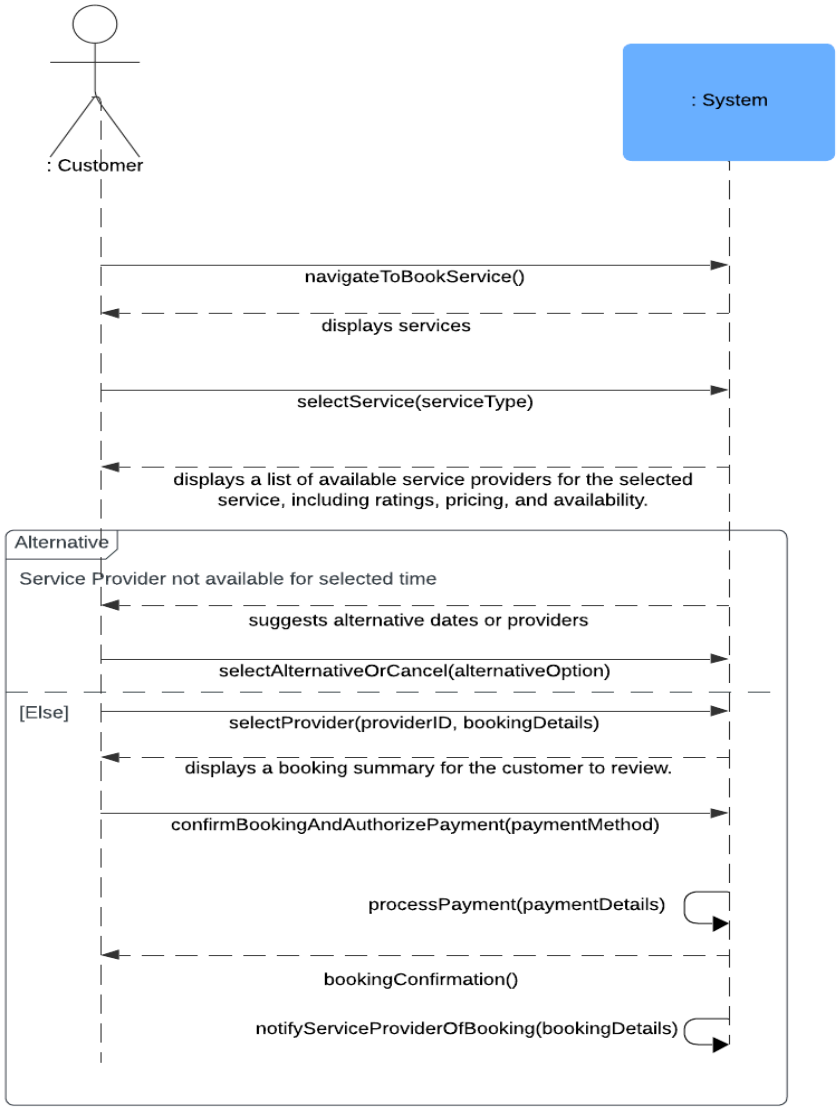
Main Success Scenario (MSS) - Book Service



Alternative Scenarios

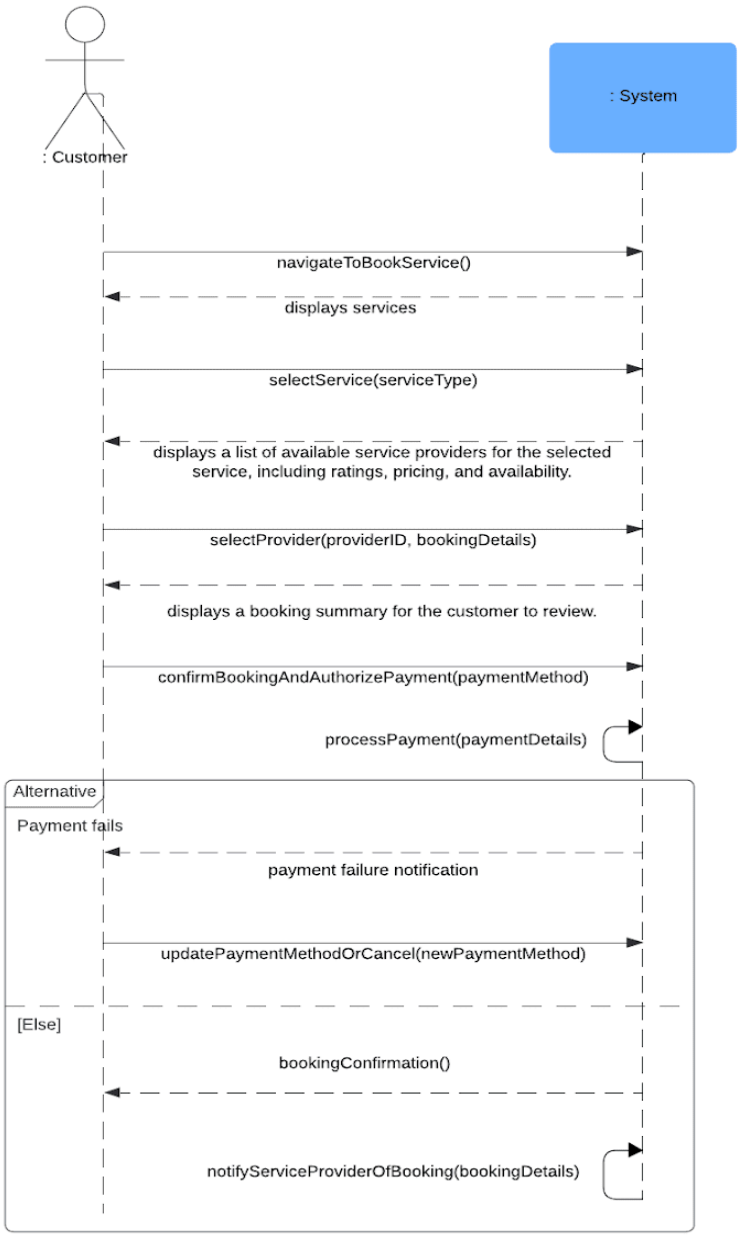
1. Service Provider Not Available

Context: After Step 4 in the MSS, if no service provider is available for the selected time.

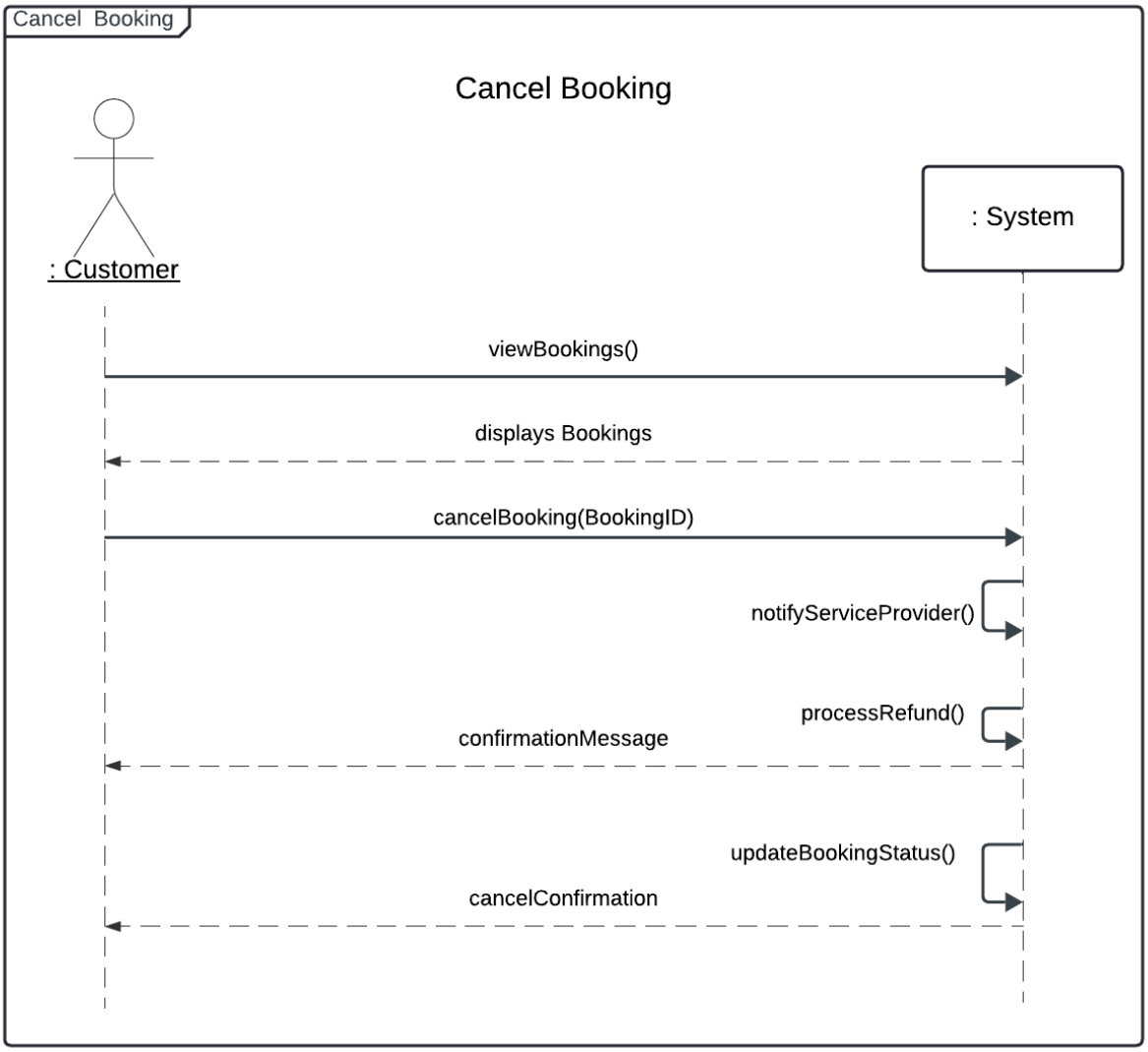


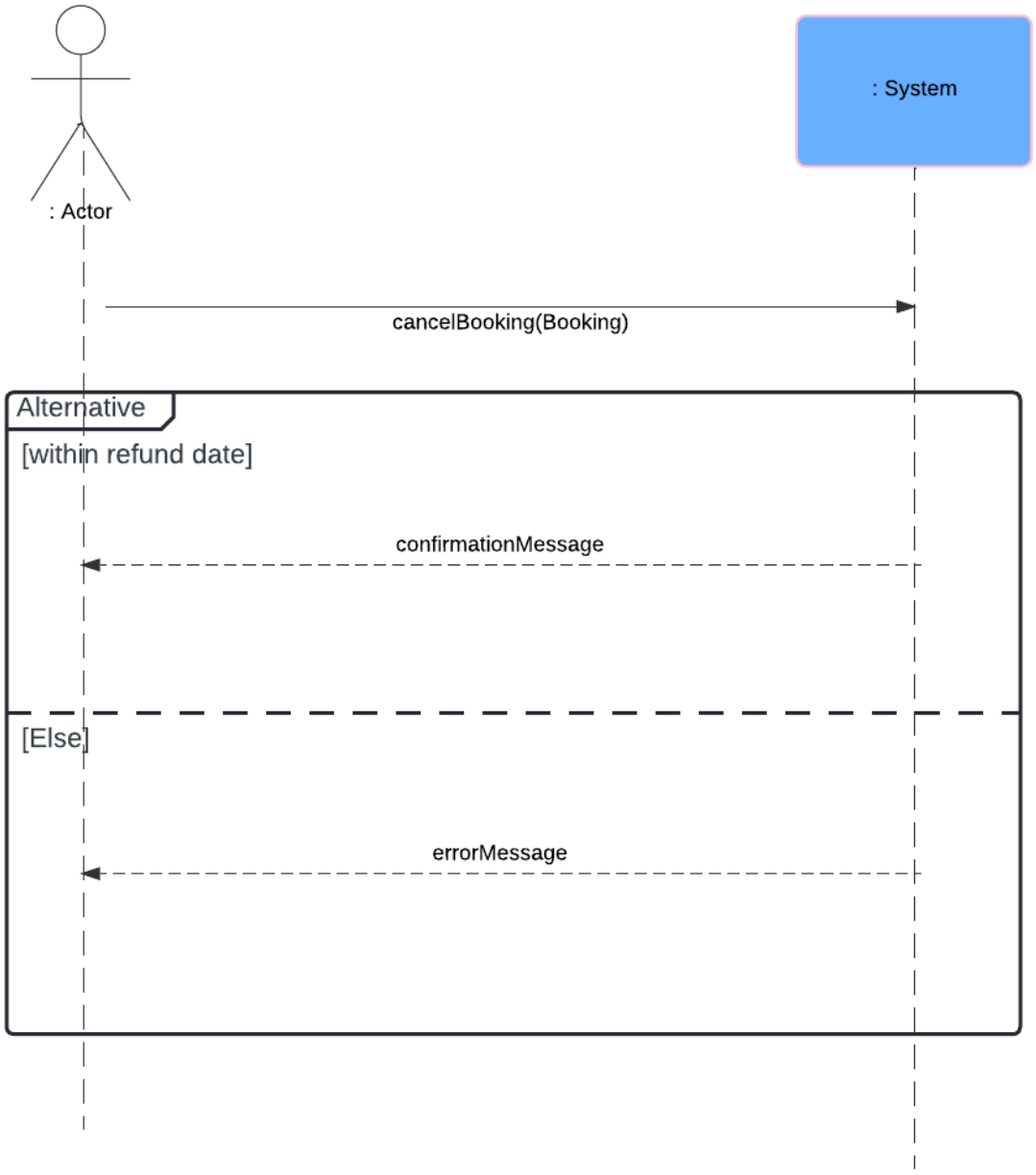
2. Payment Fails

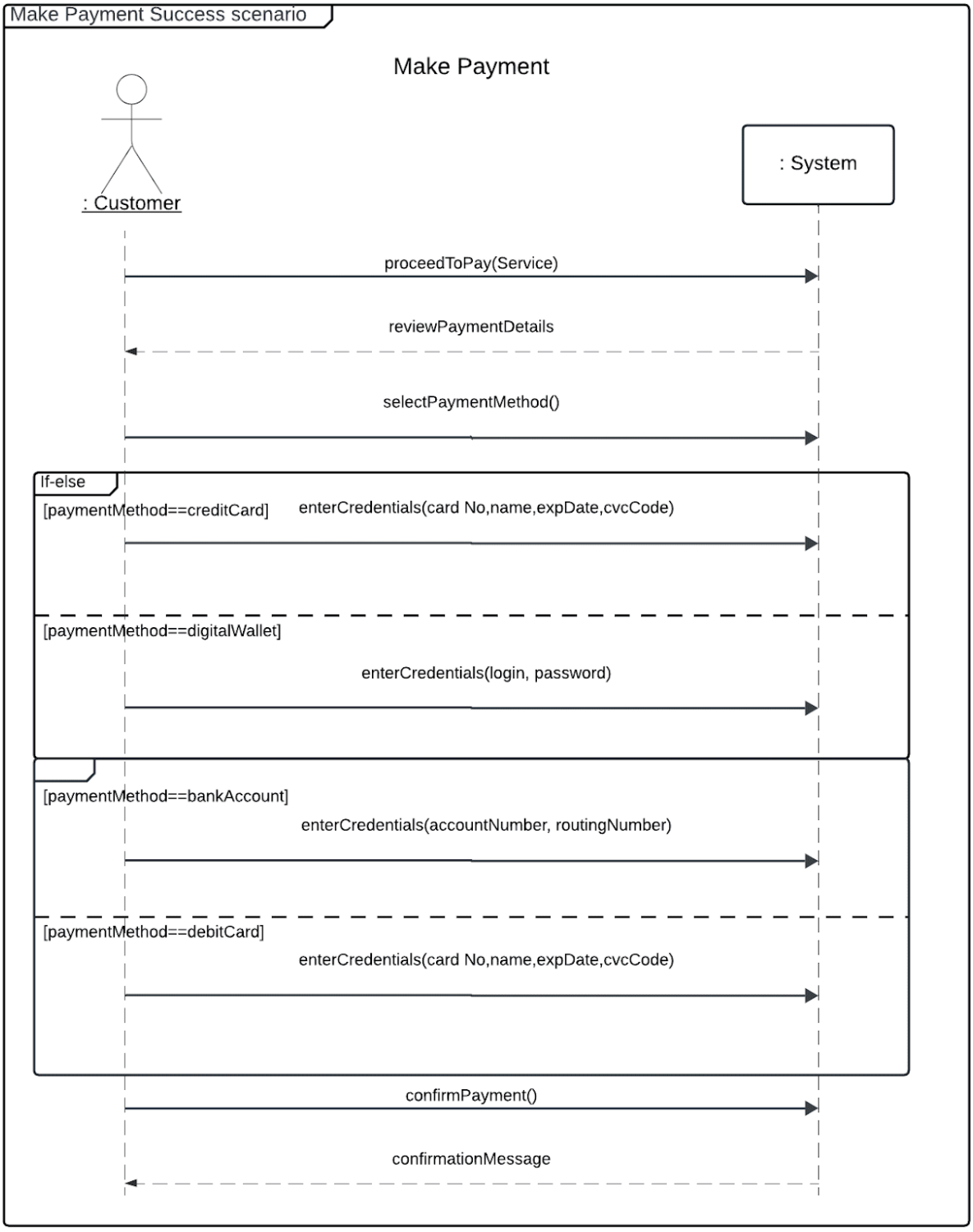
Context: After Step 8 in the MSS, if the payment fails due to insufficient funds or technical issues.

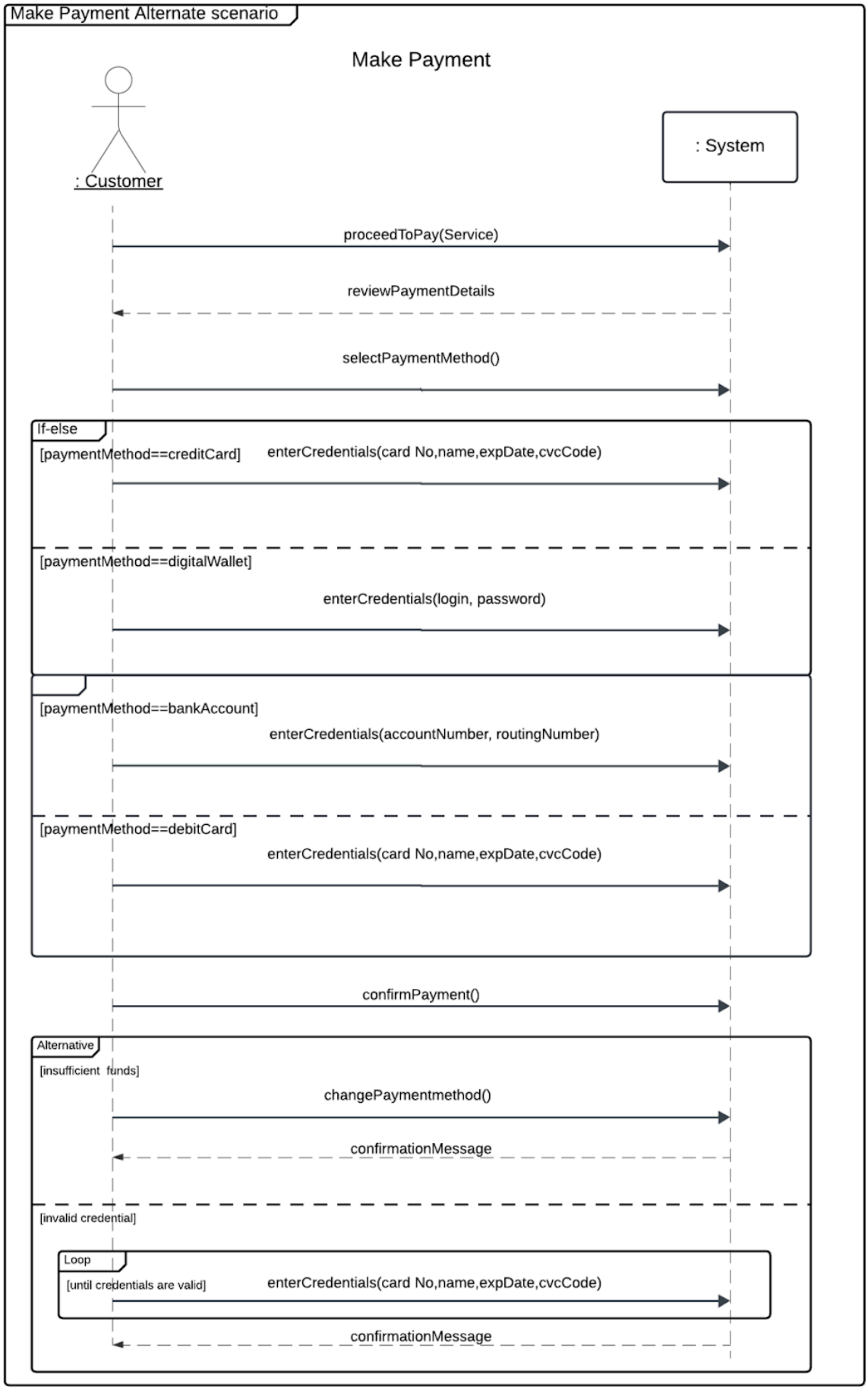


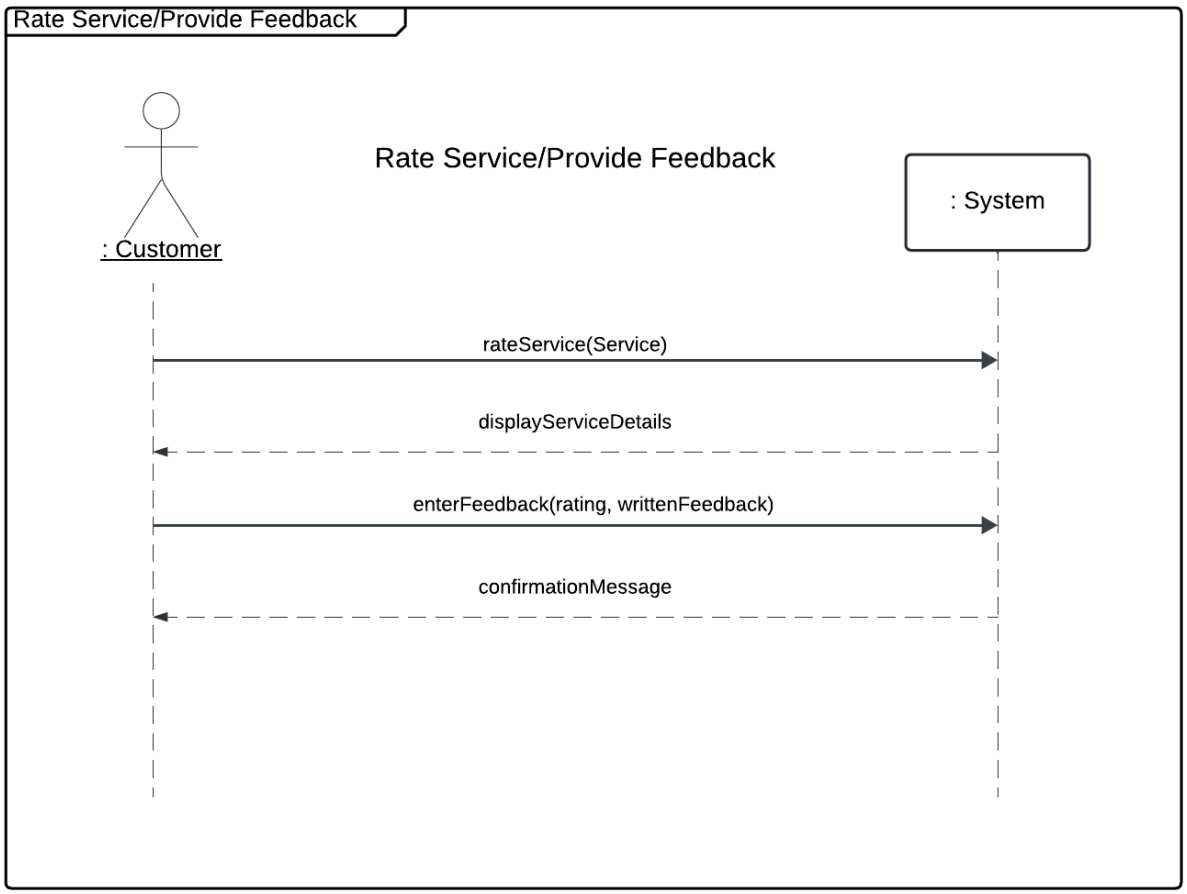
CANCEL BOOKING

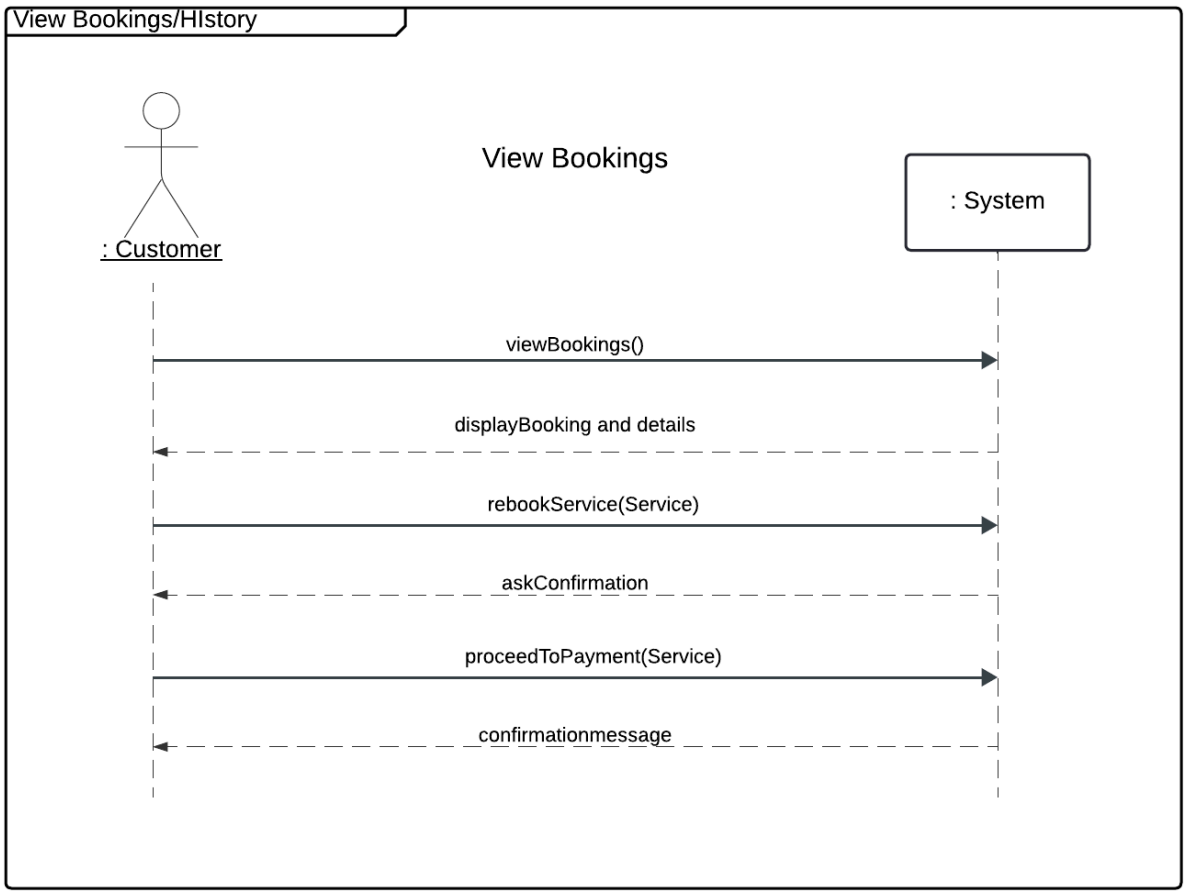


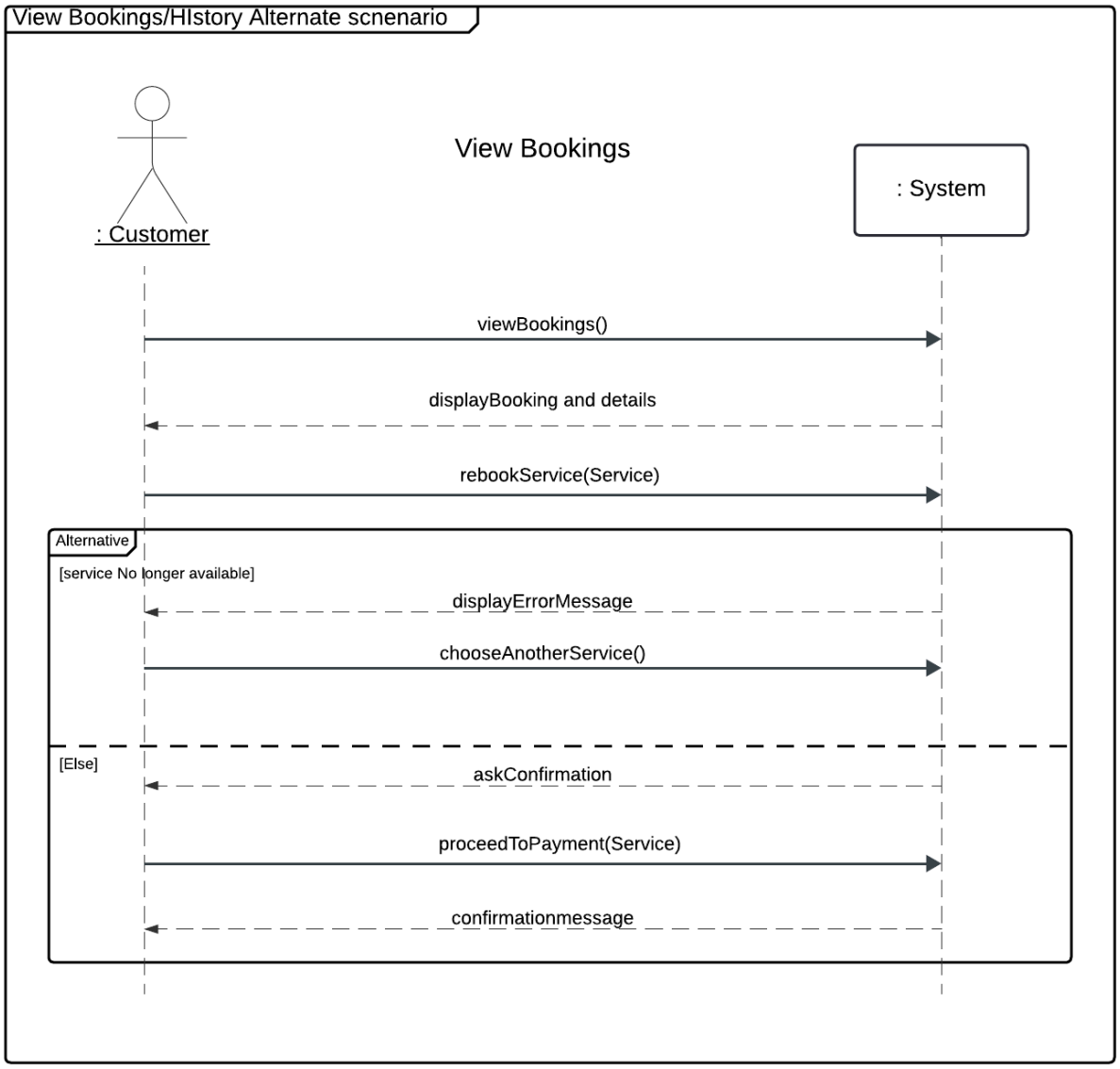

: Actor
: System
cancelBooking(Booking)
Alternative
[within refund date]
confirmationMessage
[Else]
errorMessage


Make Payment Success scenario
Make Payment
: Customer
: System
proceedToPay(Service)
reviewPaymentDetails
selectPaymentMethod()
If-else
[paymentMethod==creditCard]
enterCredentials(card No,name,expDate,cvcCode)
[paymentMethod==digitalWallet]
enterCredentials(login, password)
[paymentMethod==bankAccount]
enterCredentials(accountNumber, routingNumber)
[paymentMethod==debitCard]
enterCredentials(card No,name,expDate,cvcCode)
confirmPayment()
confirmationMessage

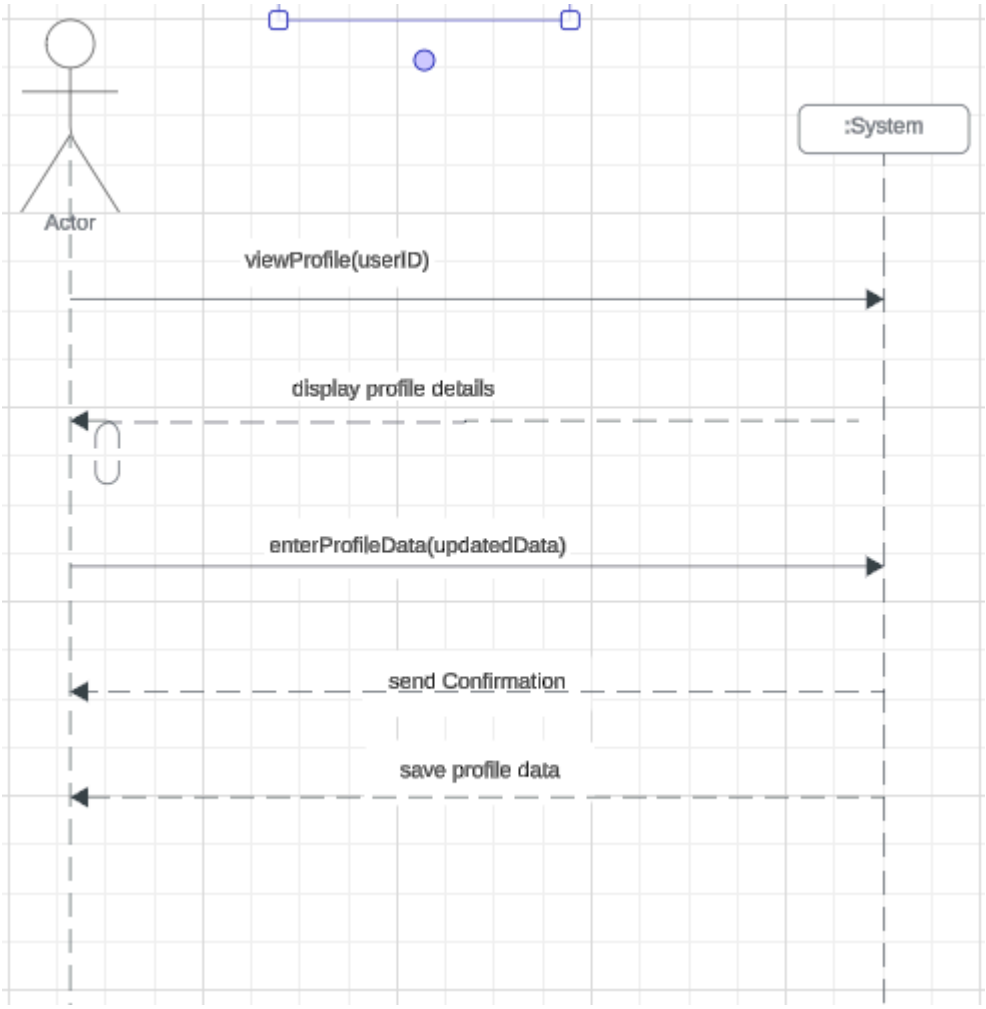

Make Payment Alternate scenario
Make Payment
: Customer
: System
proceedToPay(Service)
reviewPaymentDetails
selectPaymentMethod()
If-else
[paymentMethod==creditCard]
enterCredentials(card No,name,expDate,cvcCode)
[paymentMethod==digitalWallet]
enterCredentials(login, password)
[paymentMethod==bankAccount]
enterCredentials(accountNumber, routingNumber)
[paymentMethod==debitCard]
enterCredentials(card No,name,expDate,cvcCode)
confirmPayment()
Alternative
[insufficient funds]
changePaymentmethod()
confirmationMessage
[invalid credential]
Loop
[until credentials are valid]
enterCredentials(card No,name,expDate,cvcCode)
confirmationMessage


Rate Service/Provide Feedback
Rate Service/Provide Feedback
: Customer
: System
rateService(Service)
displayServiceDetails
enterFeedback(rating, writtenFeedback)
confirmationMessage

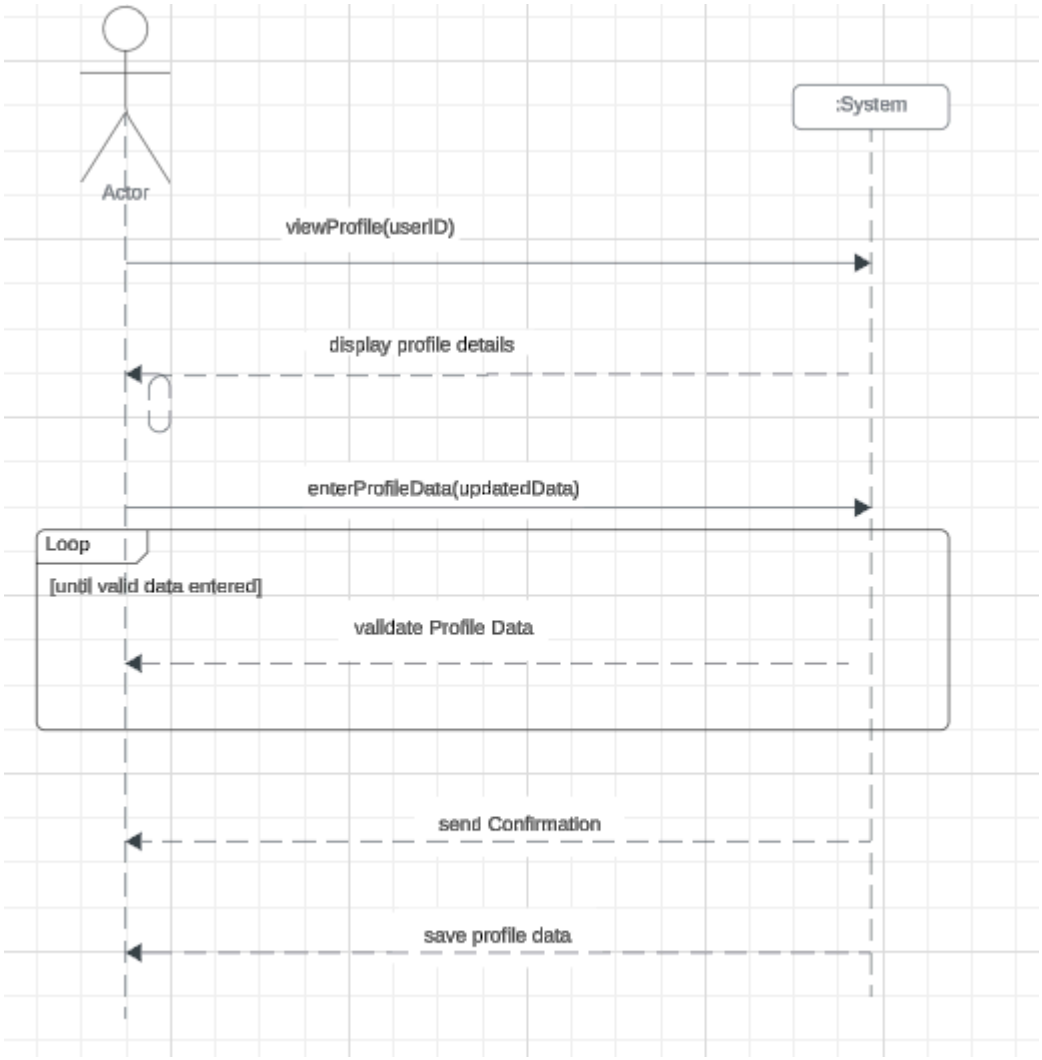

View Bookings/HIstory
View Bookings
: Customer
: System
viewBookings()
displayBooking and details
rebookService(Service)
askConfirmation
proceedToPayment(Service)
confirmationmessage


View Bookings/HIstory Alternate scnenario
View Bookings
: Customer
: System
viewBookings()
displayBooking and details
rebookService(Service)
Alternative
[service No longer available]
displayErrorMessage
chooseAnotherService()
[Else]
askConfirmation
proceedToPayment(Service)
confirmationmessage

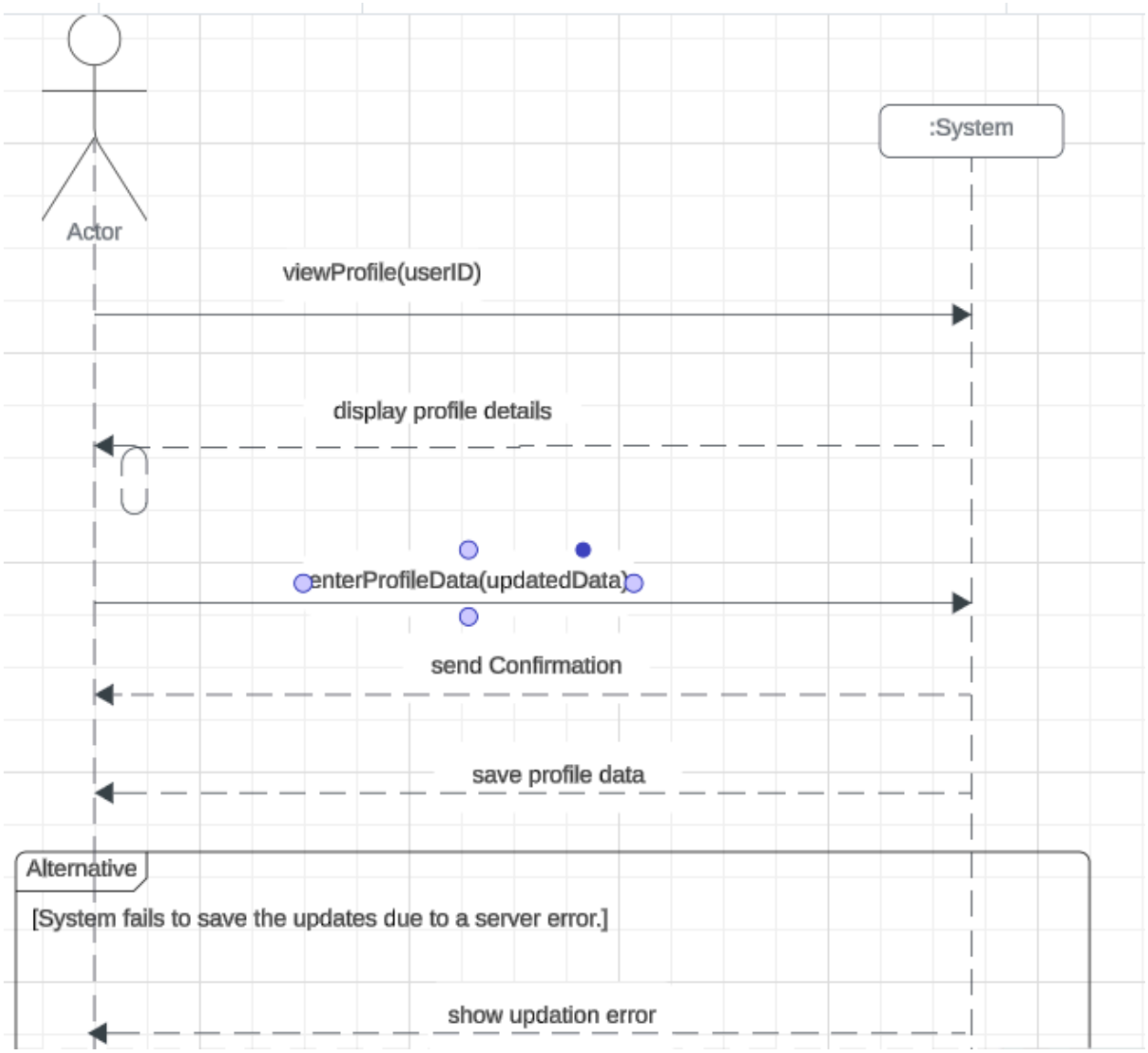
Update Profile MAIN SUCCESS SCENARIO



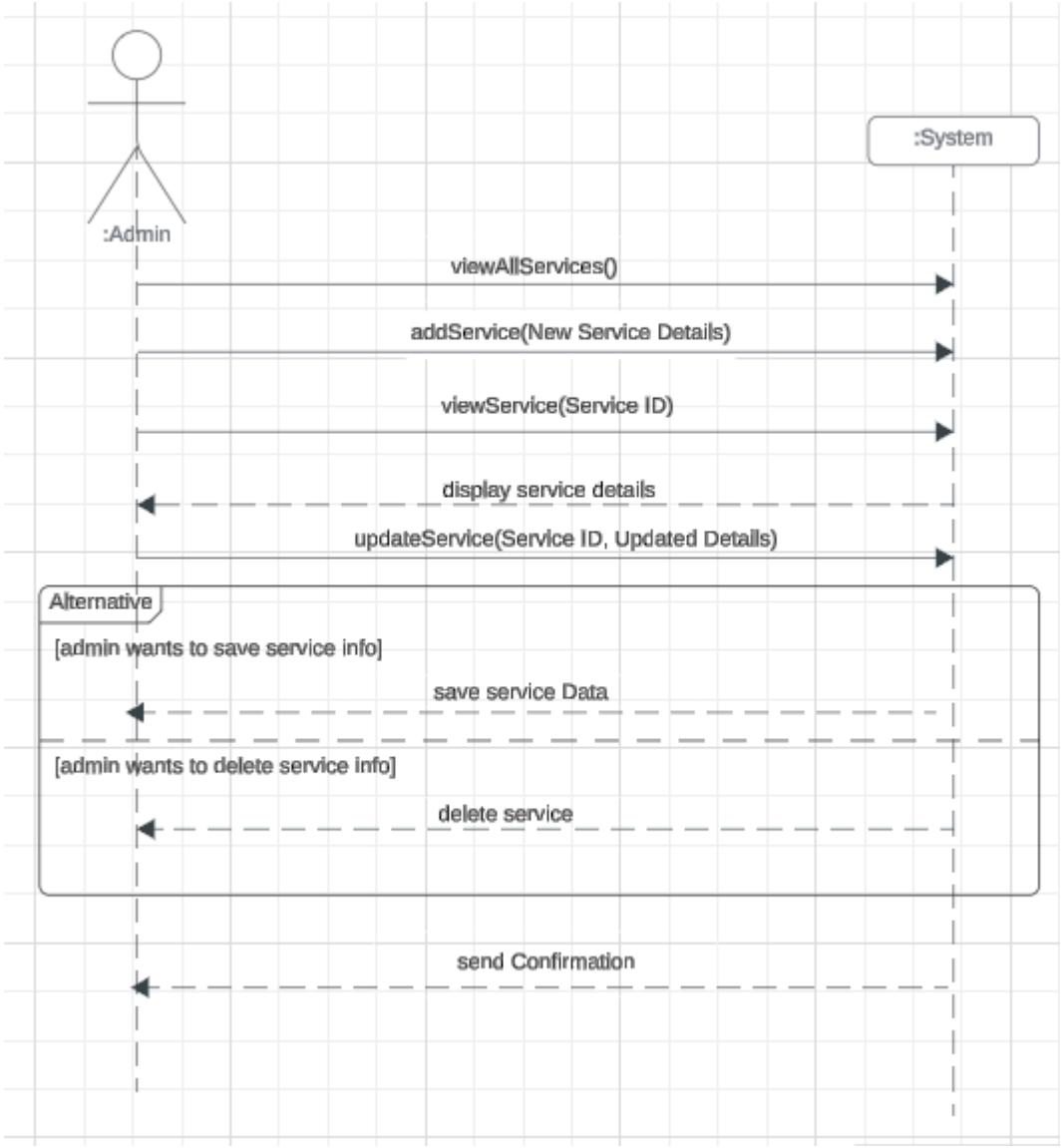
Update Profile: Invalid Information



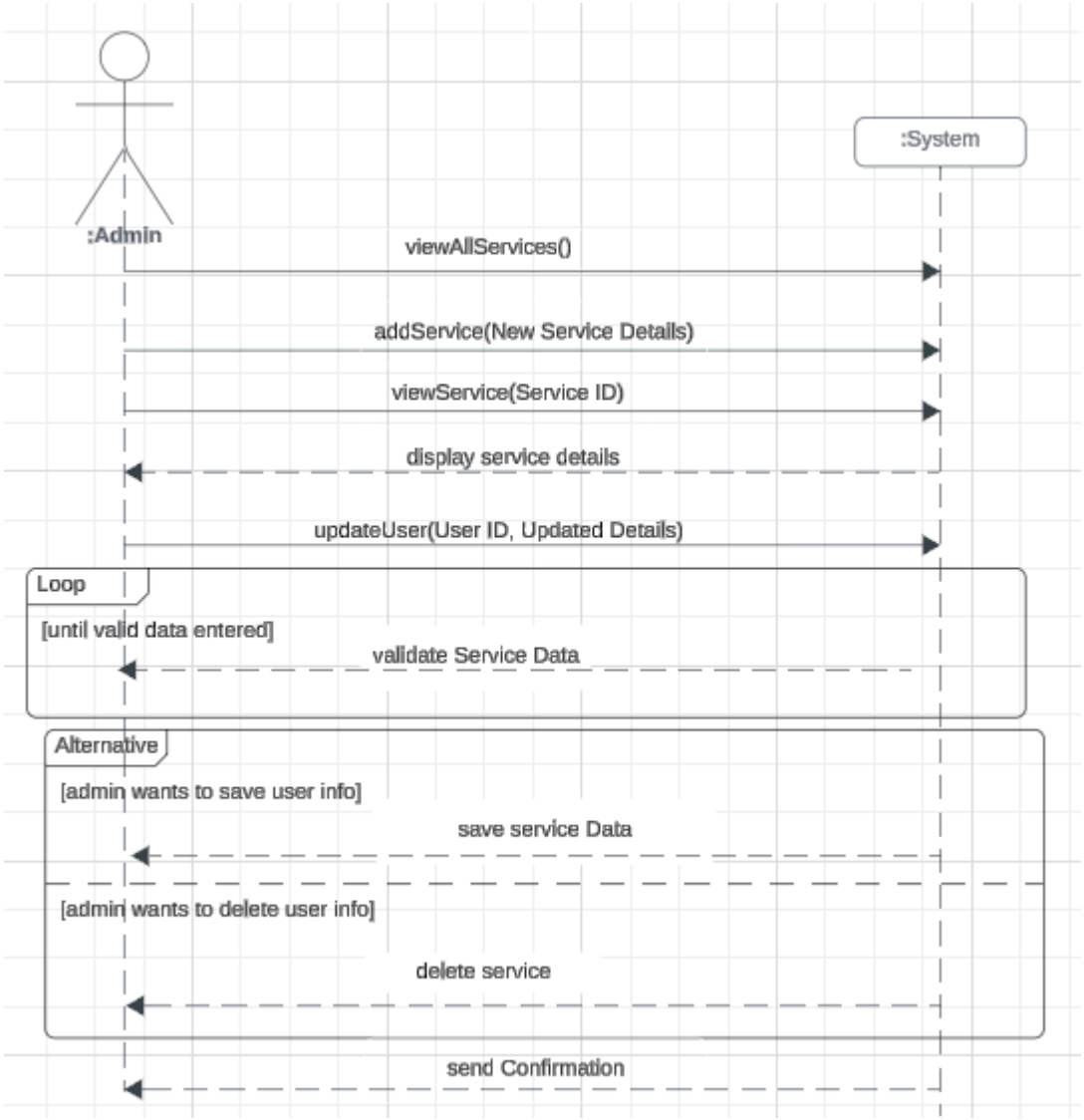
Update Profile: System Update Failure



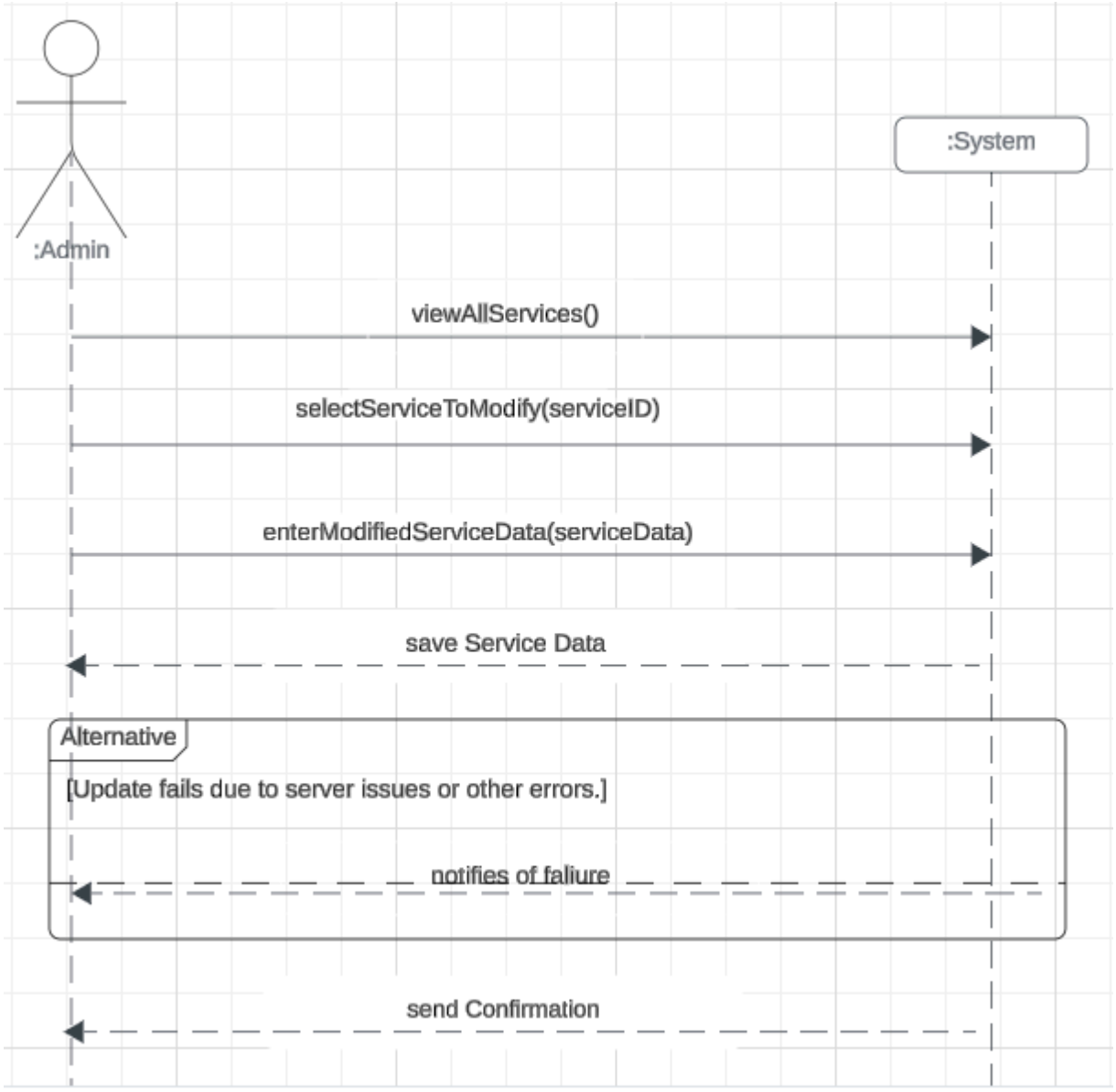
MANAGE SERVICES: MSS



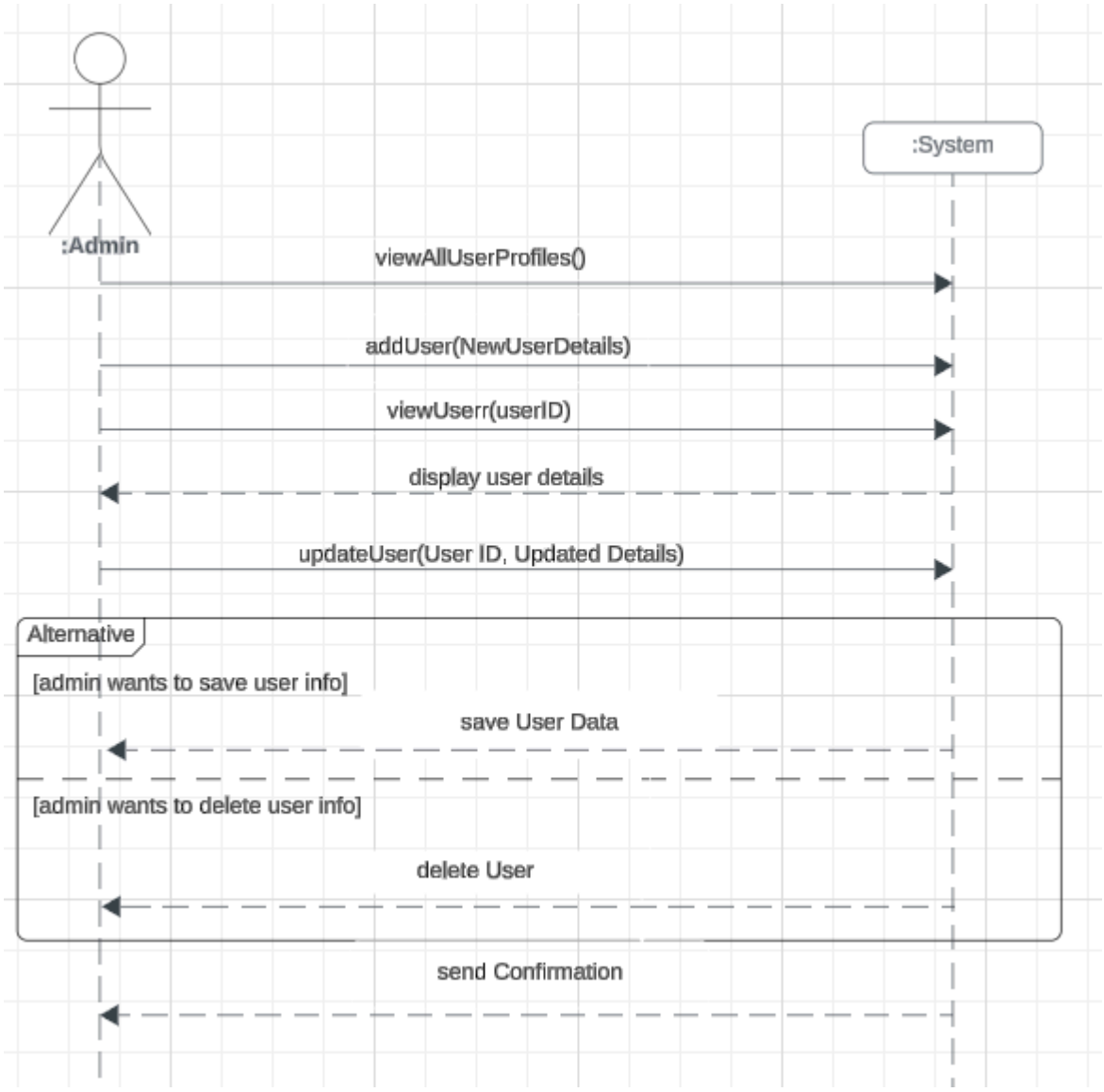
MANAGE SERVICES: invalid data



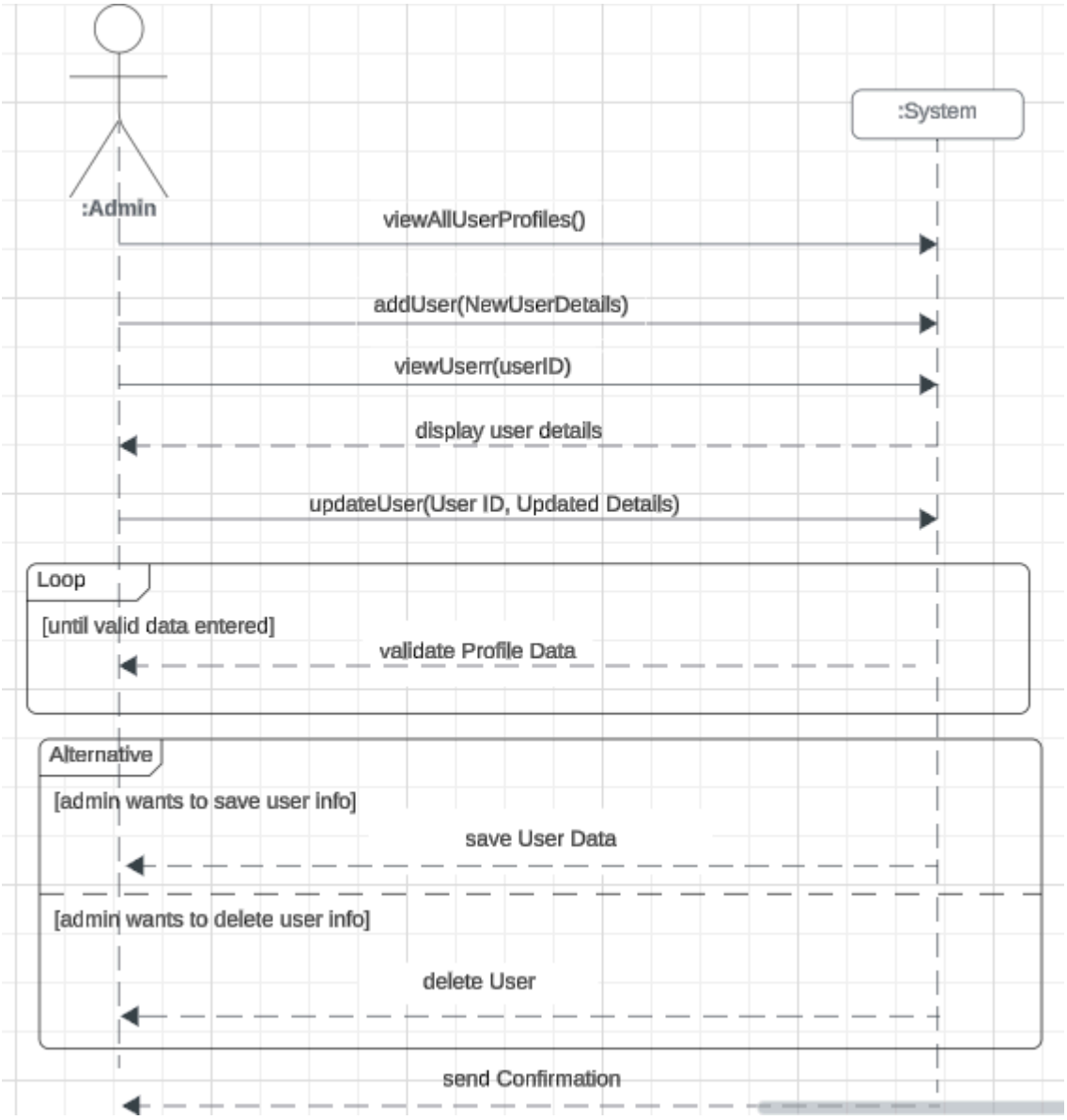
MANAGE SERVICES: System Update Failure



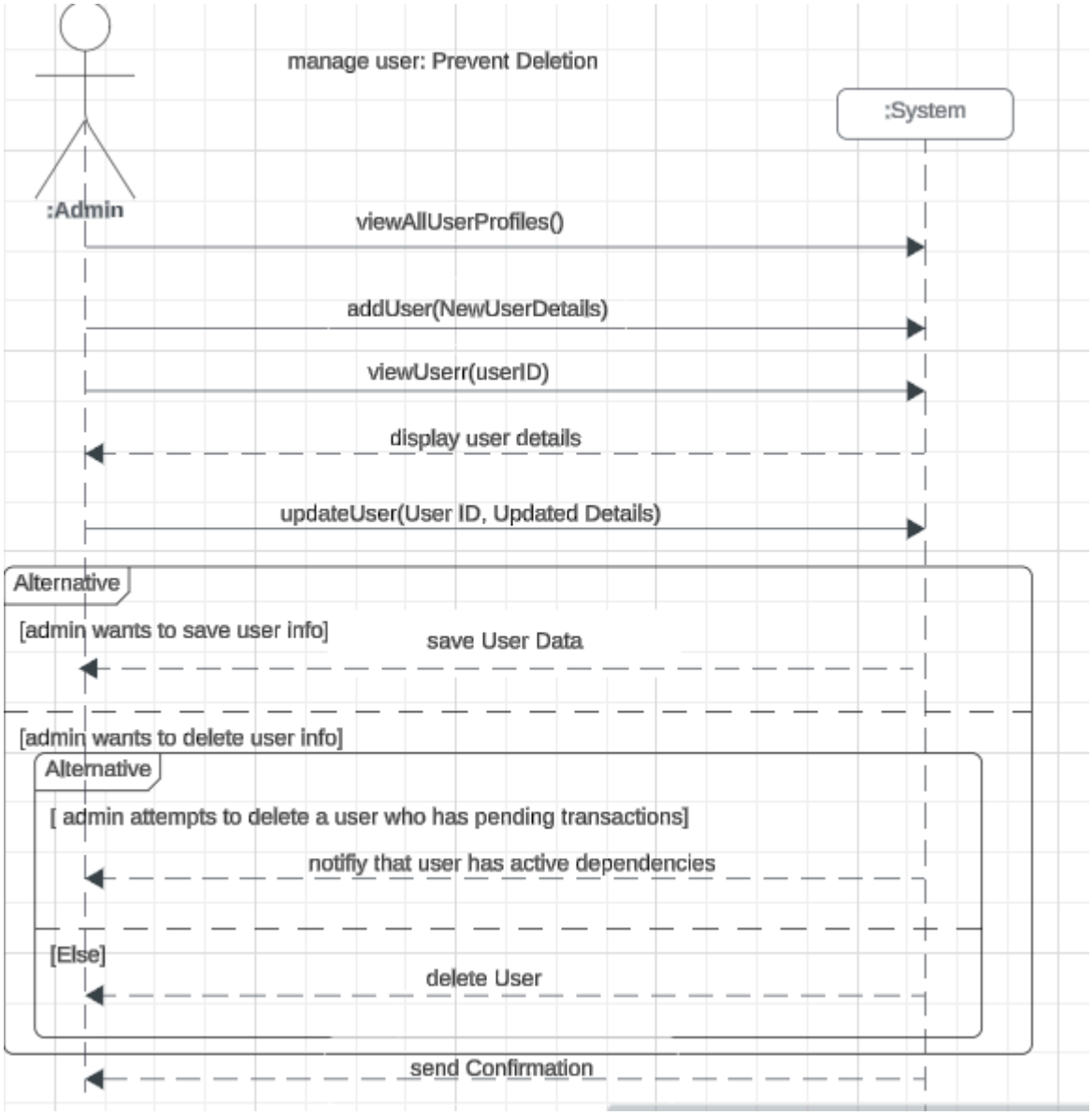
Manage Users: MSS



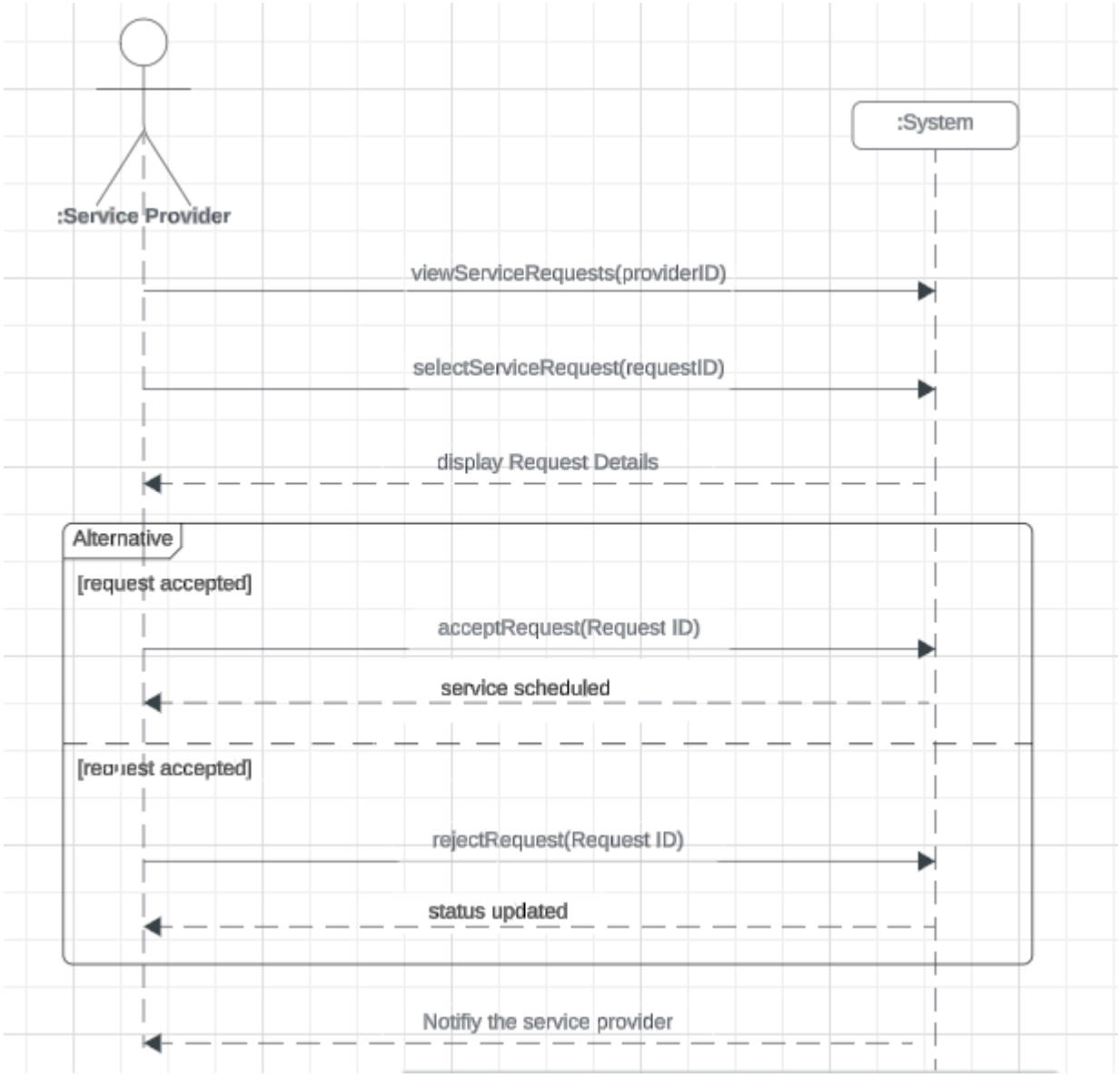
Manage Users:Invalid Service Details



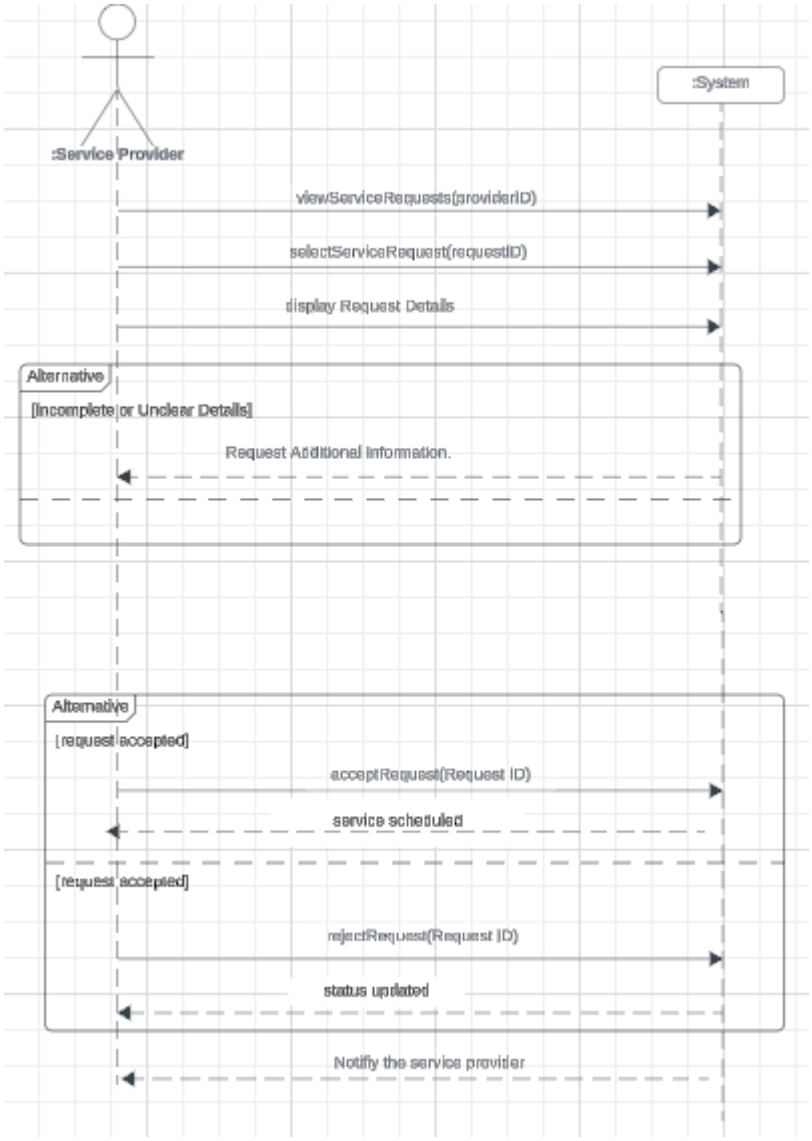
Manage Users: Prevent Deletion



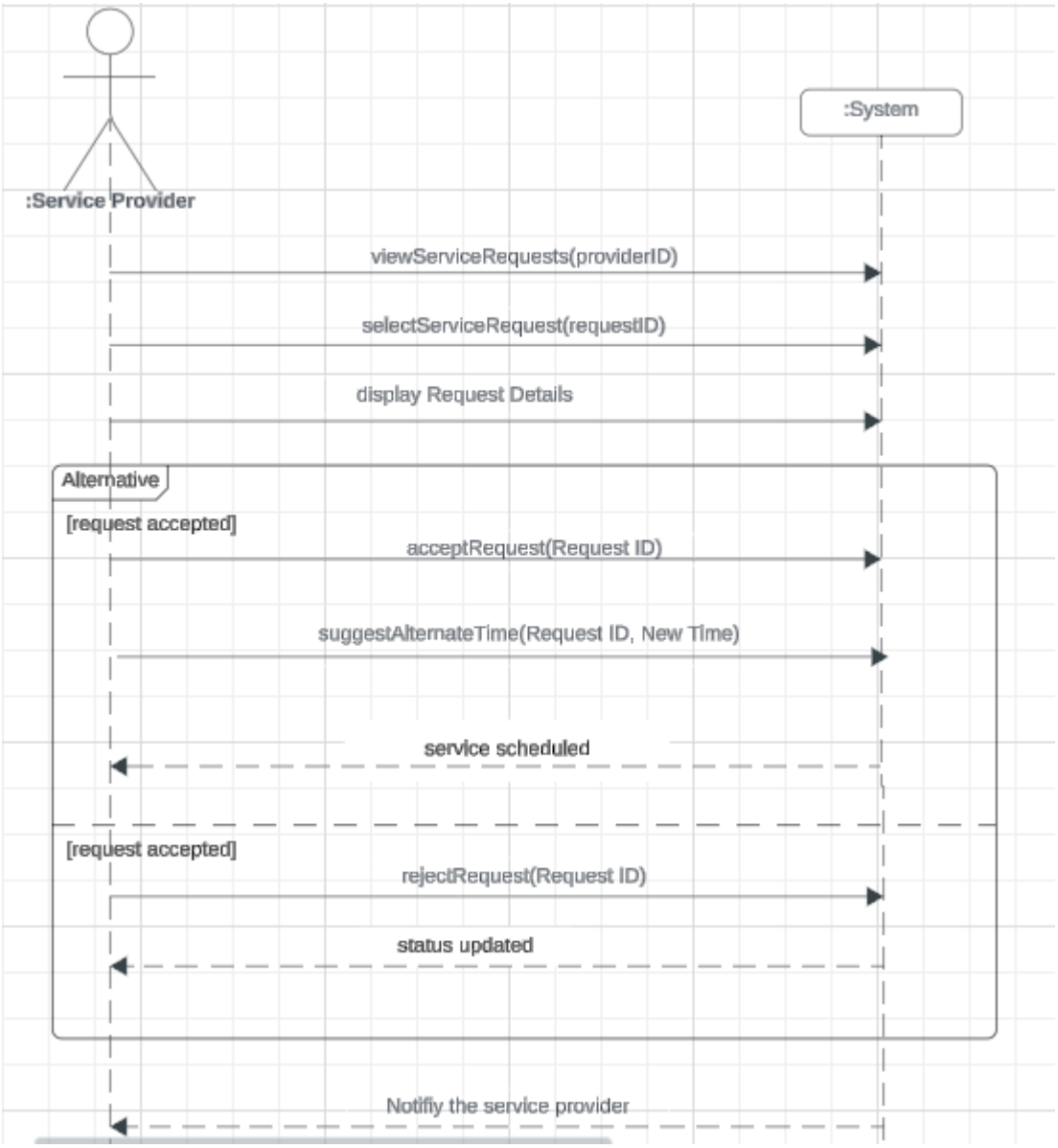
Accept/Reject Request: MSS



Accept/Reject Request: Incomplete or Unclear Details



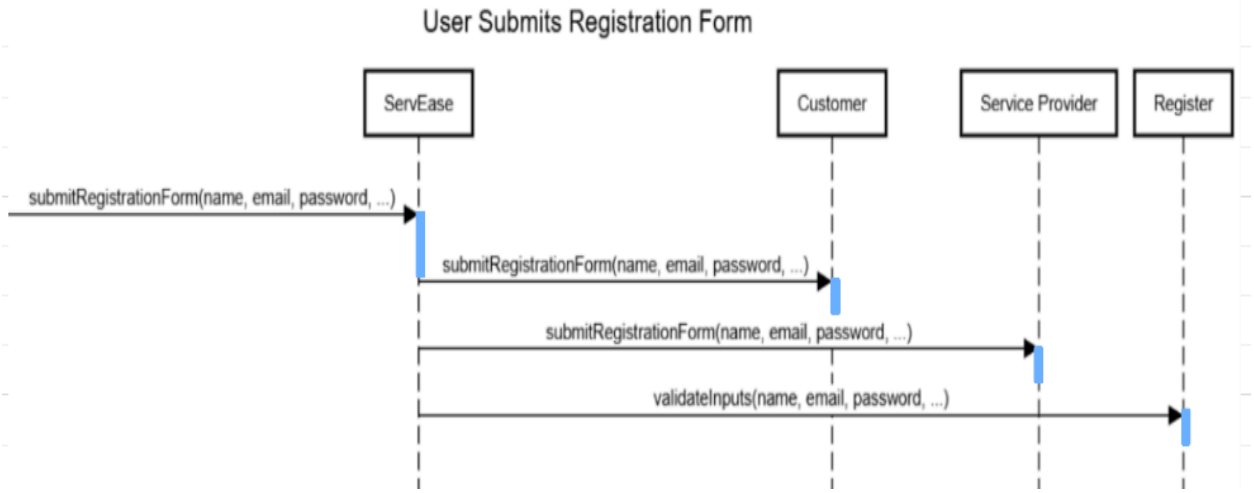
Accept/Reject Request: Accept with Suggested Time Change



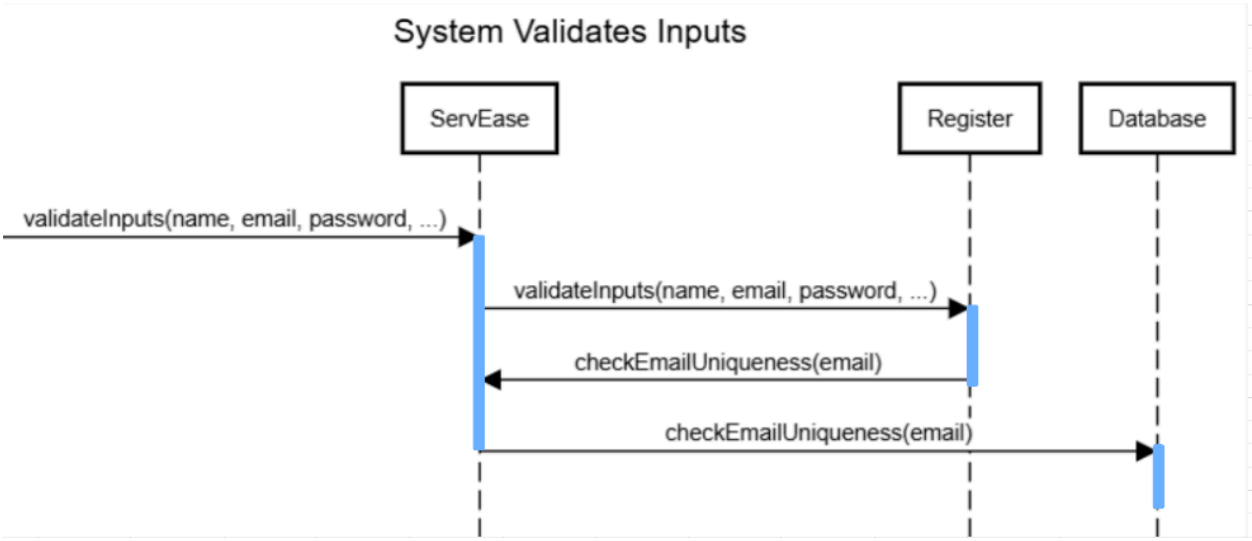
6. Sequence Diagram

Use case 1: Register account

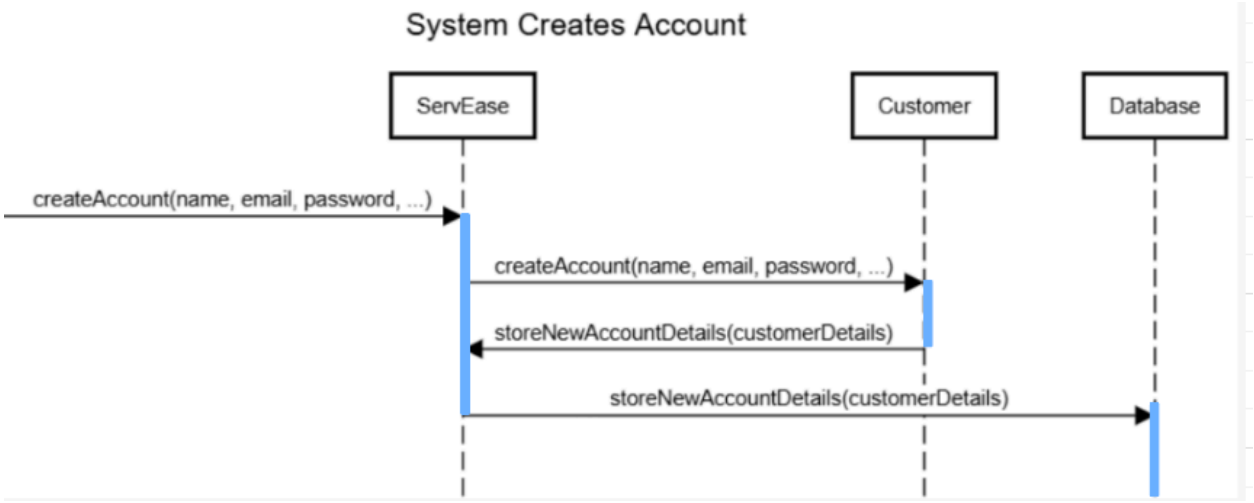
Operation: `submitRegistrationForm(name, email, password, etc.)`



Operation: `validateInputs(name, email, password, etc.)`

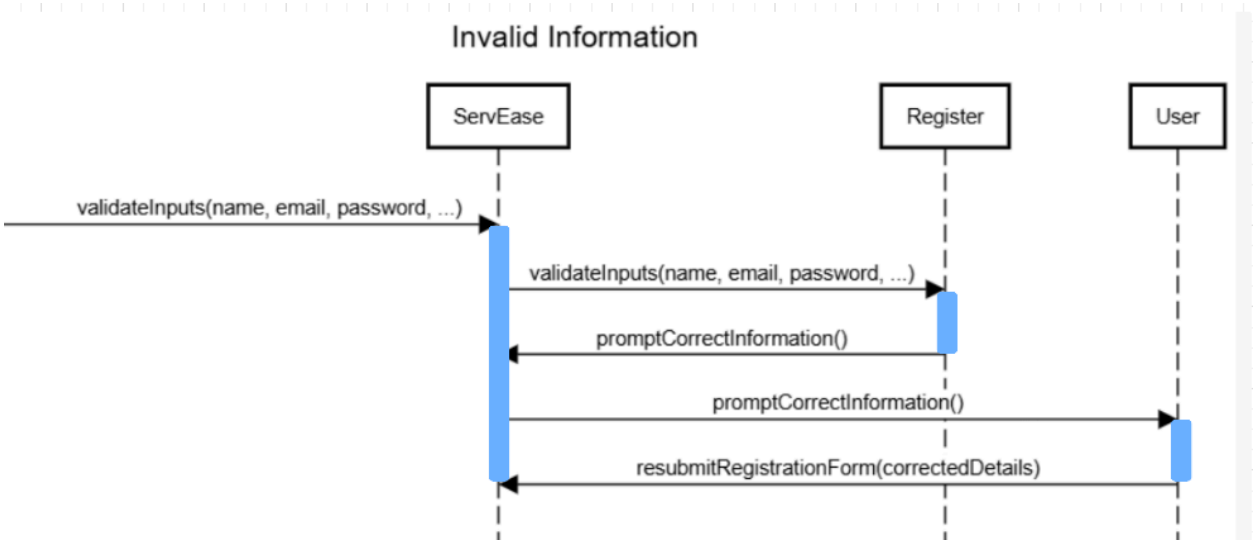


Operation: createAccount(name, email, password, etc.)



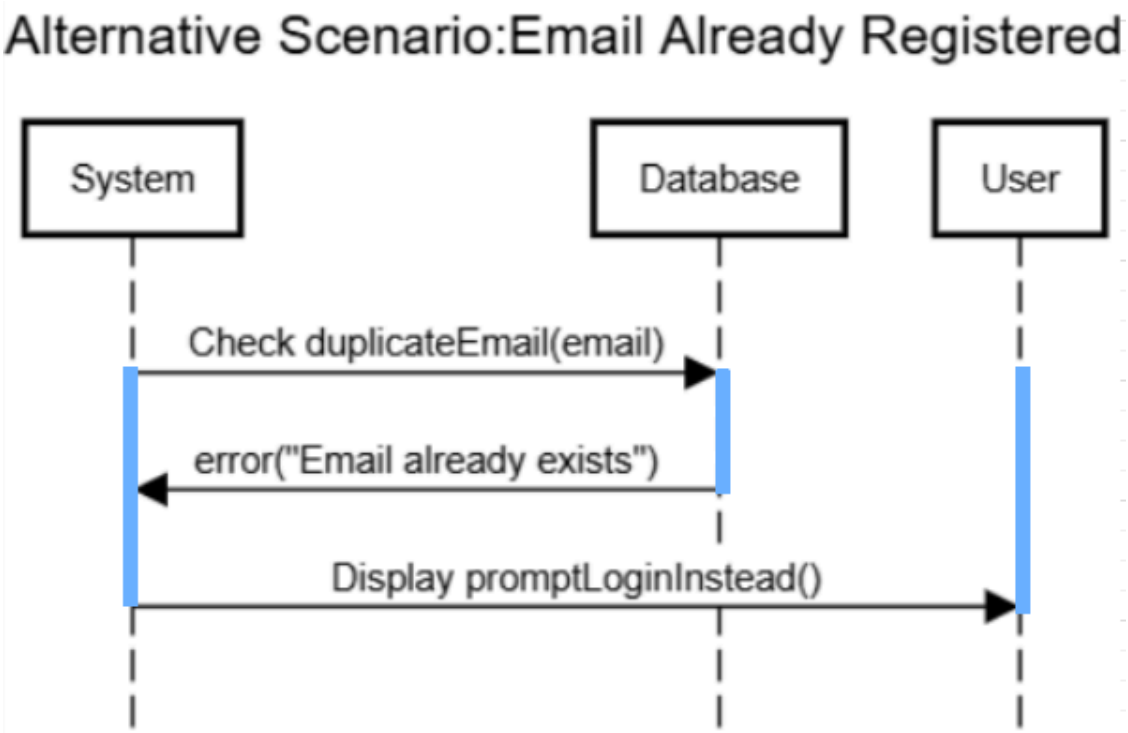
Alternative Scenario: User Enters Invalid Information

Operation: submitRegistrationForm(invalidData)



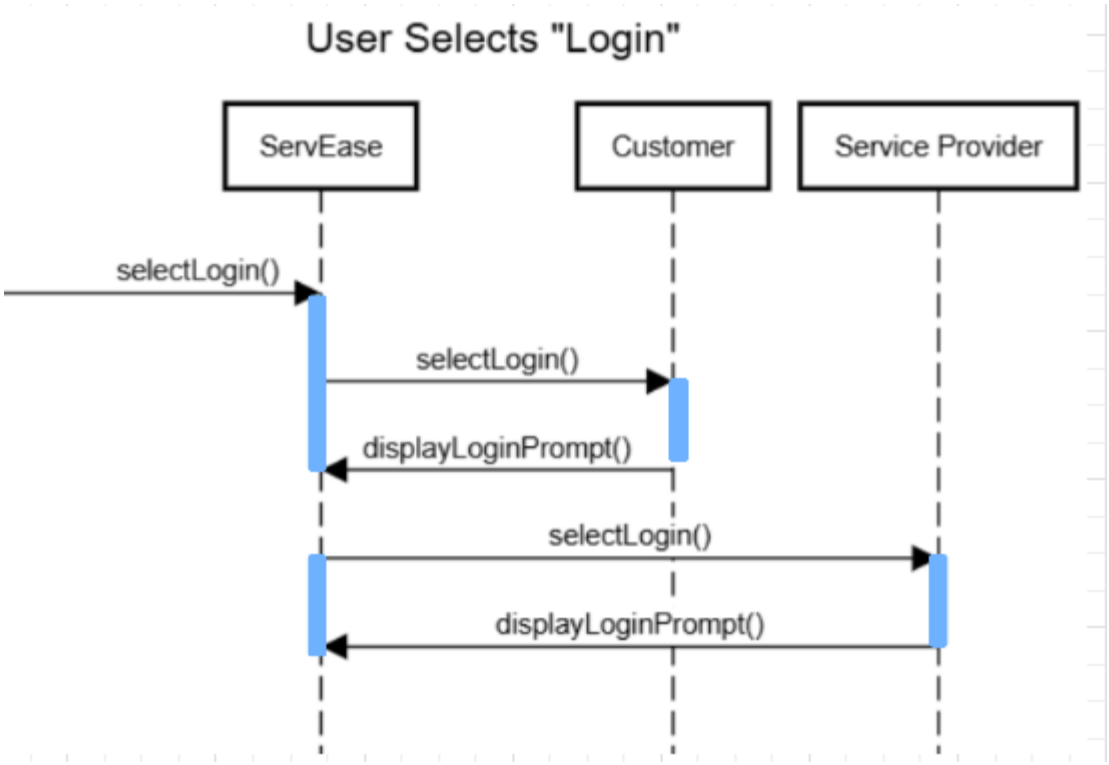
Alternative Scenario: Email Already Registered

Operation: createAccount(duplicateEmail)

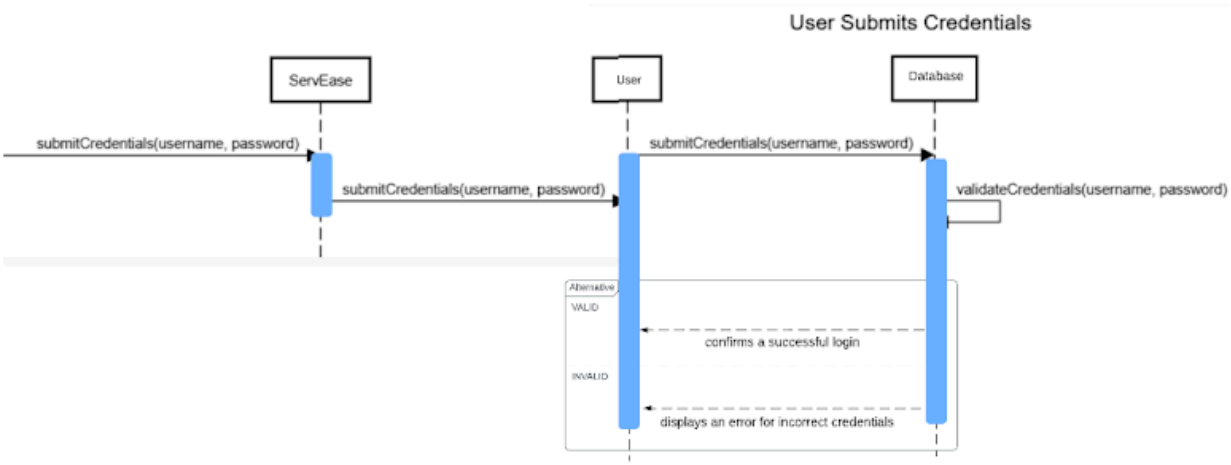


Use case 2: Login/Logout

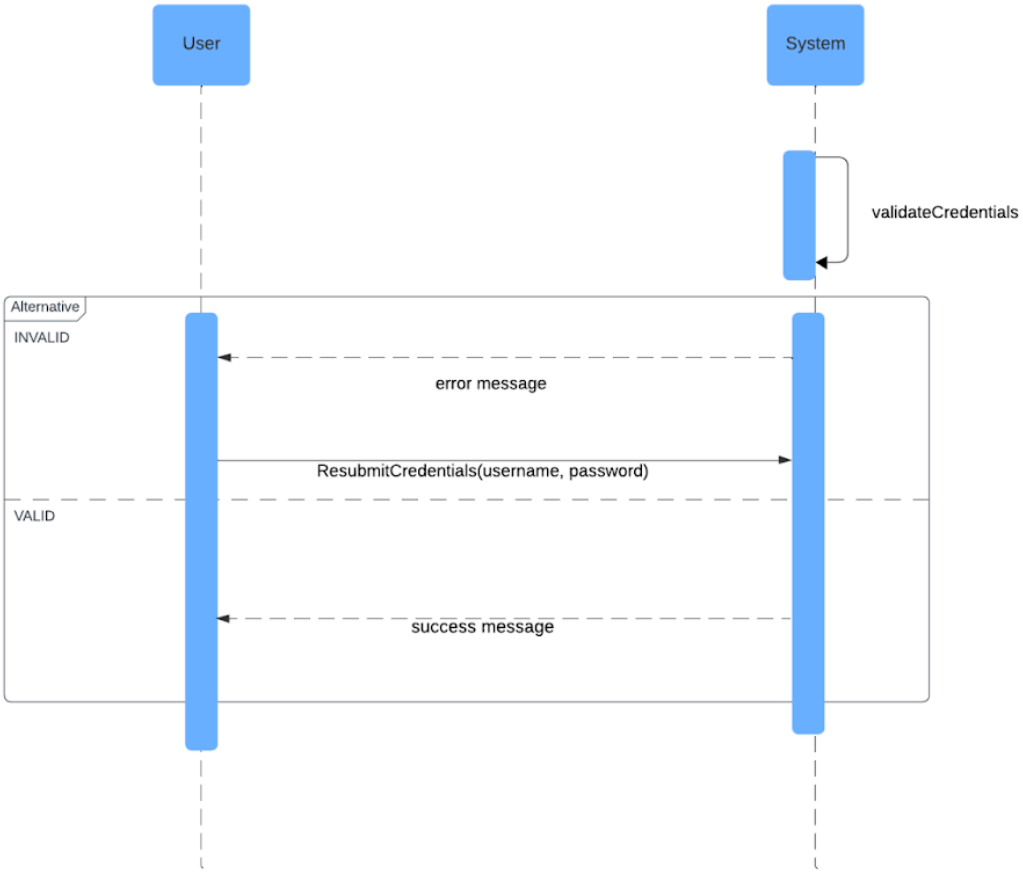
Sequence Diagram for **Event 1: User Selects "Login"**



Sequence Diagram for **`Event 2: User Submits Credentials`**



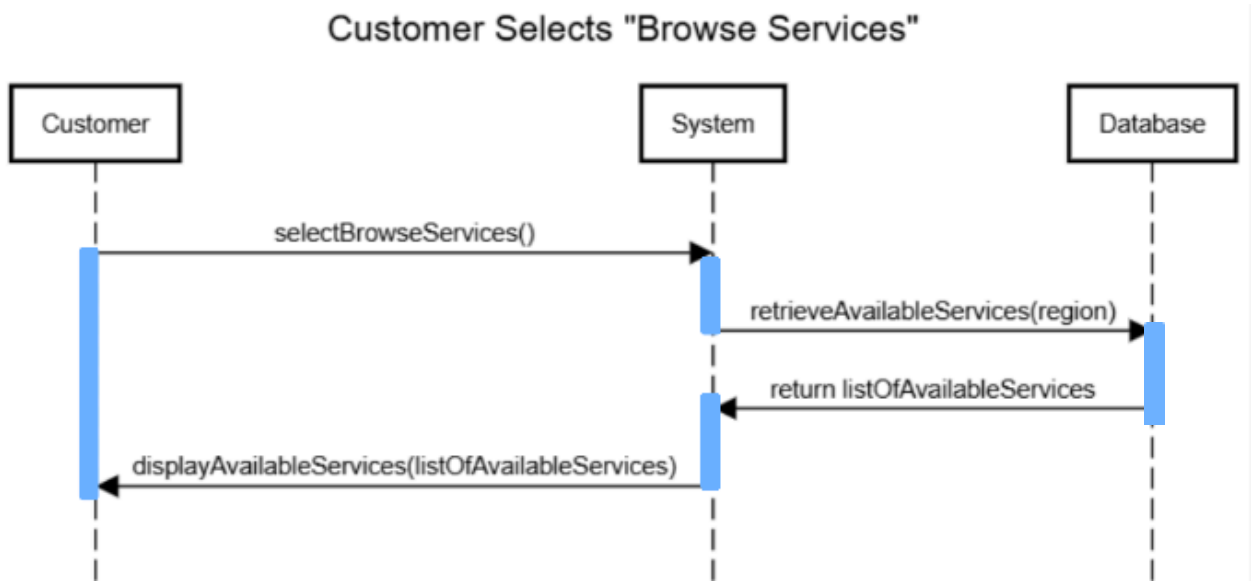
Sequence Diagram for Alternative Scenario `3a: Incorrect Credentials`



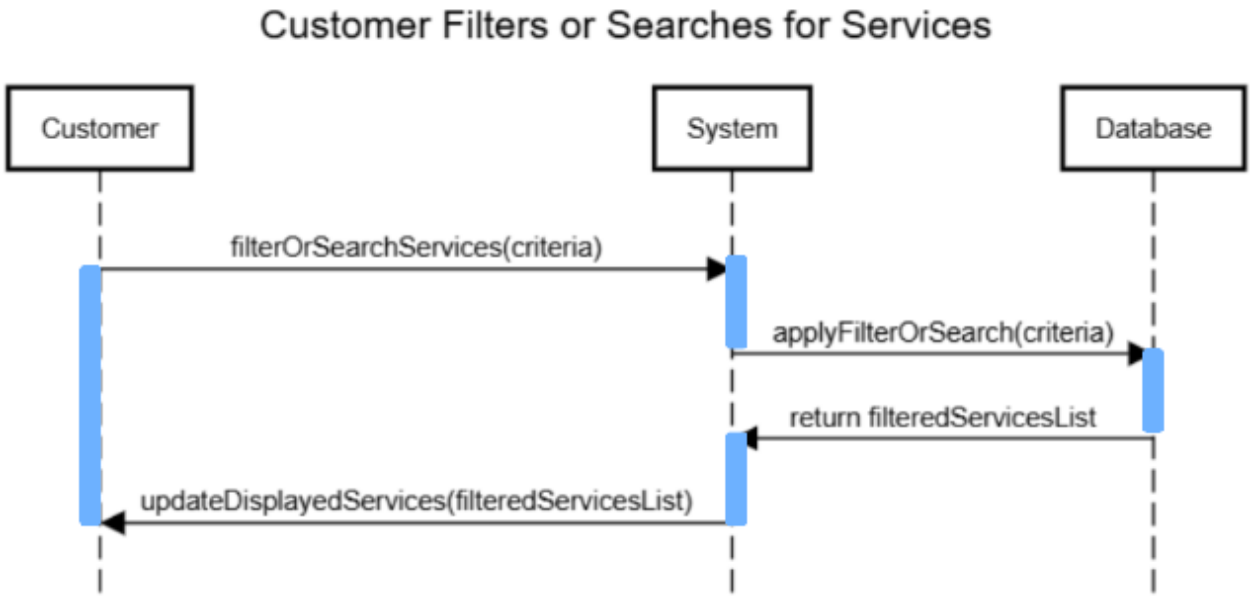
Use case 3: Browse Services

SEQUENCE DIAGRAMS

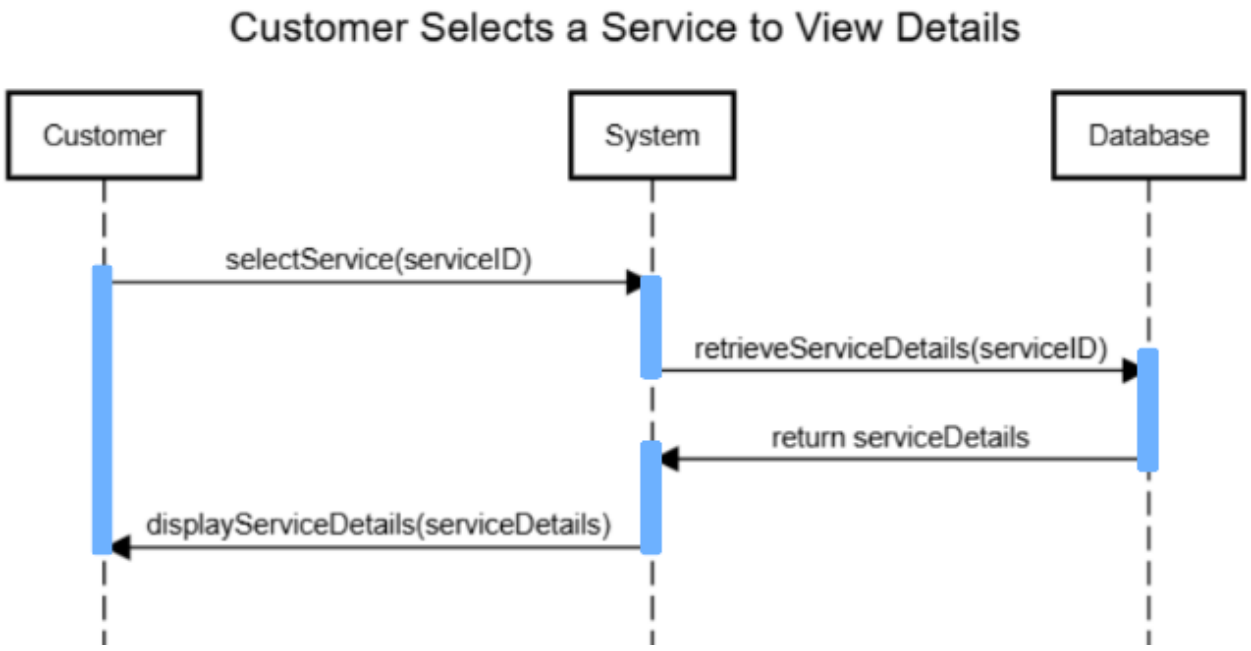
Event 1: Customer Selects "Browse Services"



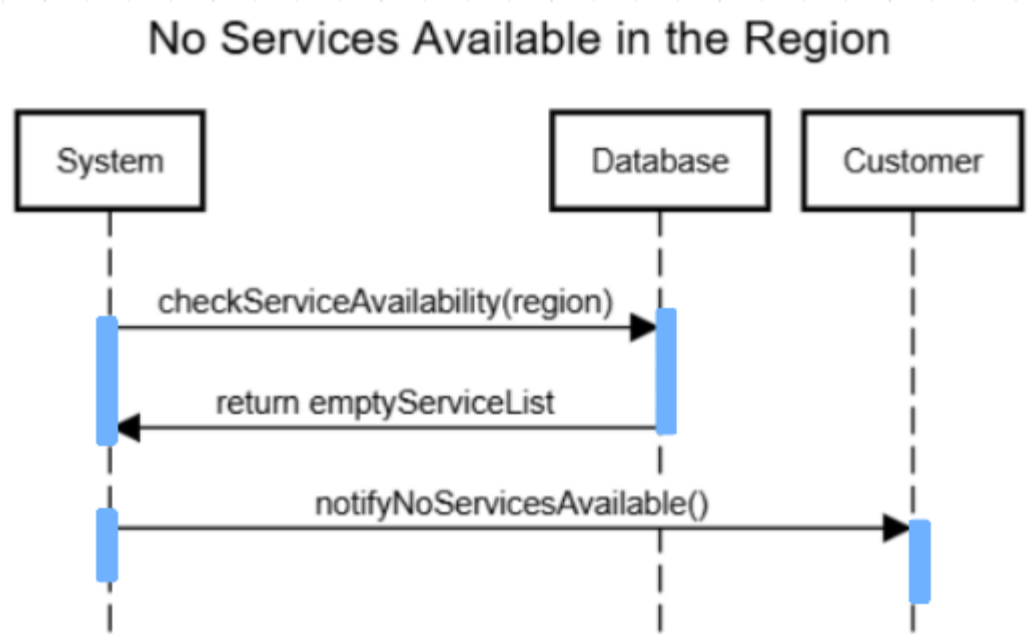
Event 2: Customer Filters or Searches for Services



Event 3: Customer Selects a Service to View Details



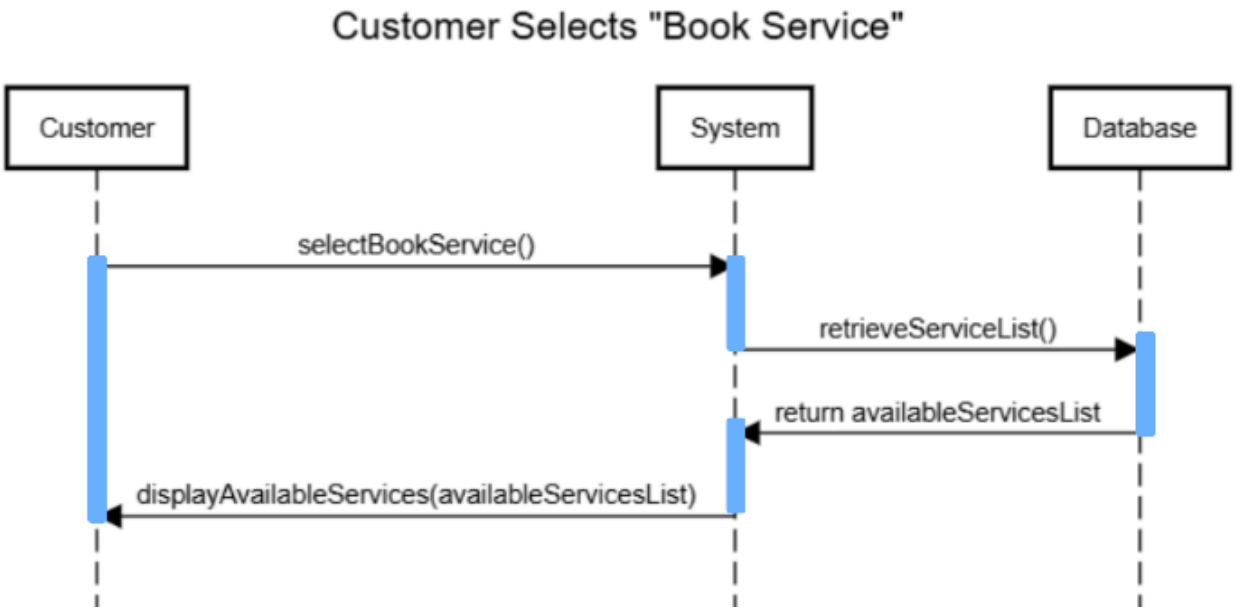
Alternative Scenario 2a: No Services Available in the Region



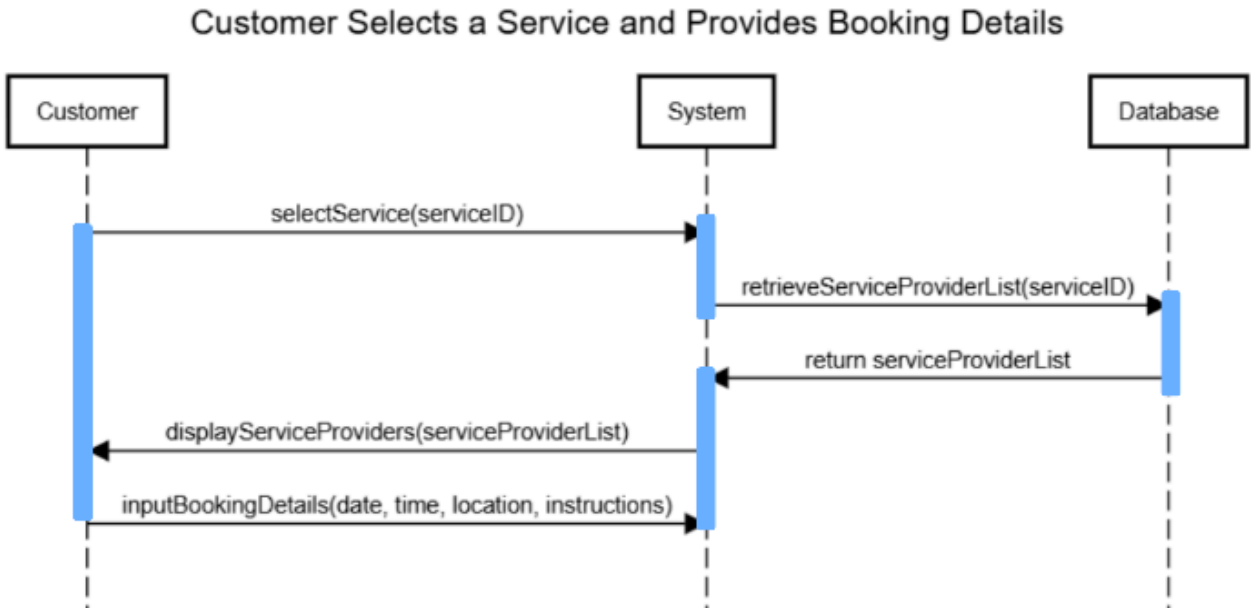
Use case 4: Book Service

SEQUENCE DIAGRAMS

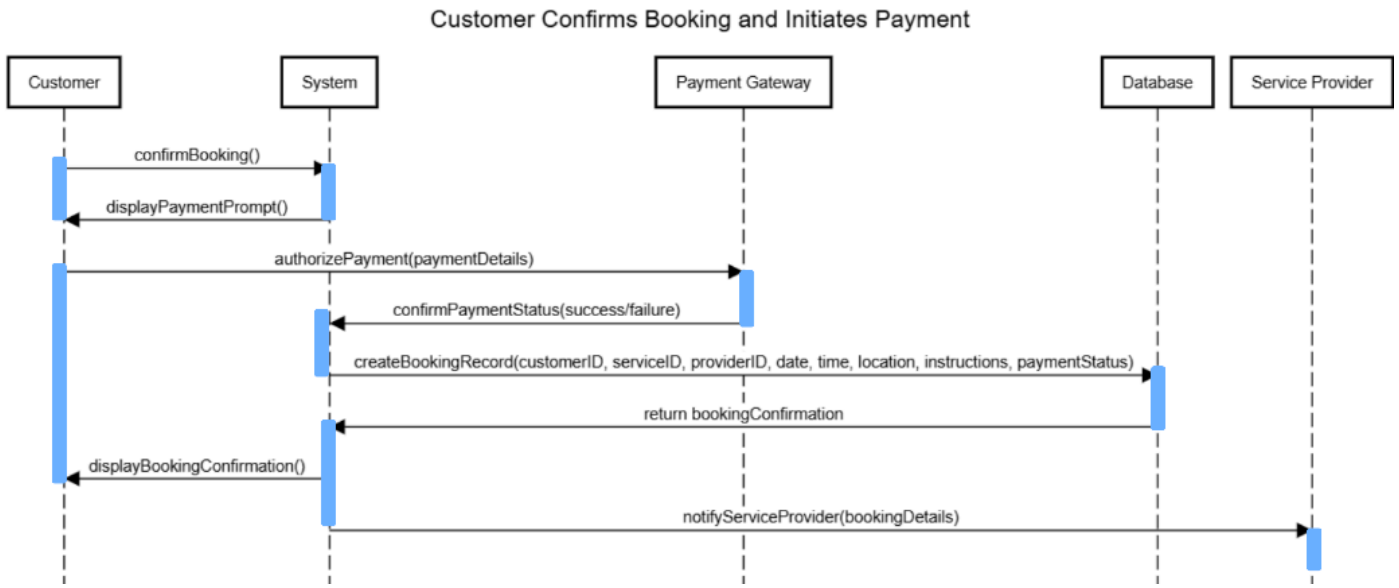
Event 1: Customer Selects "Book Service"



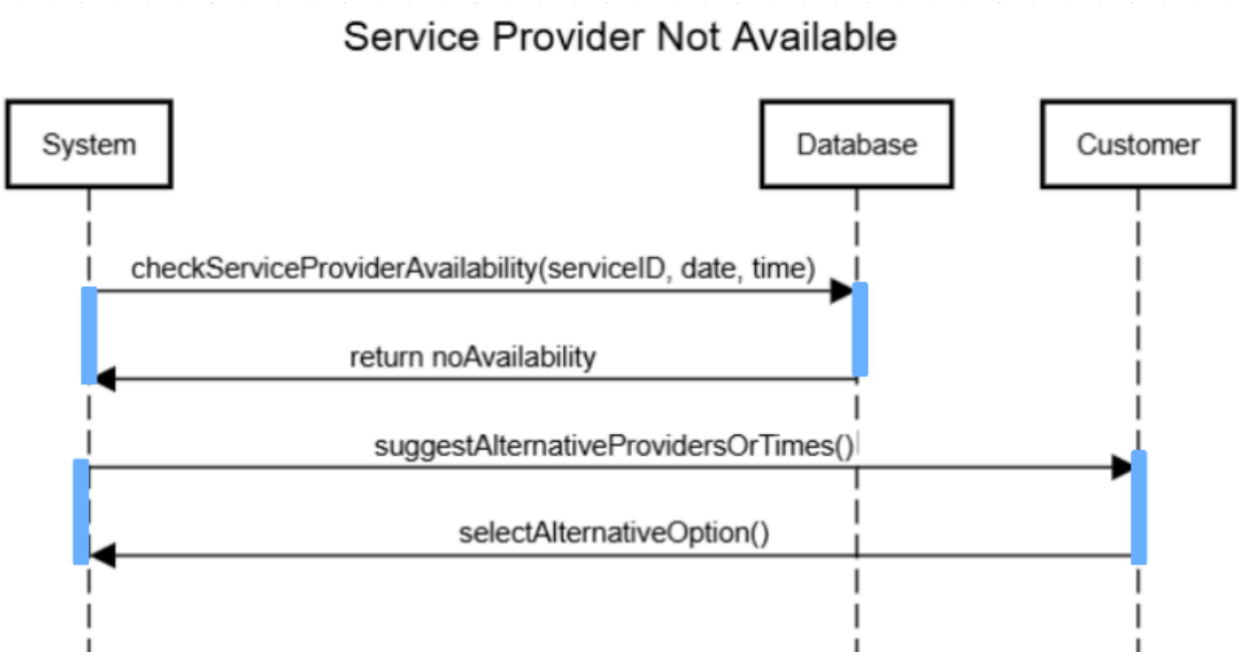
Event 2: Customer Selects a Service and Provides Booking Details



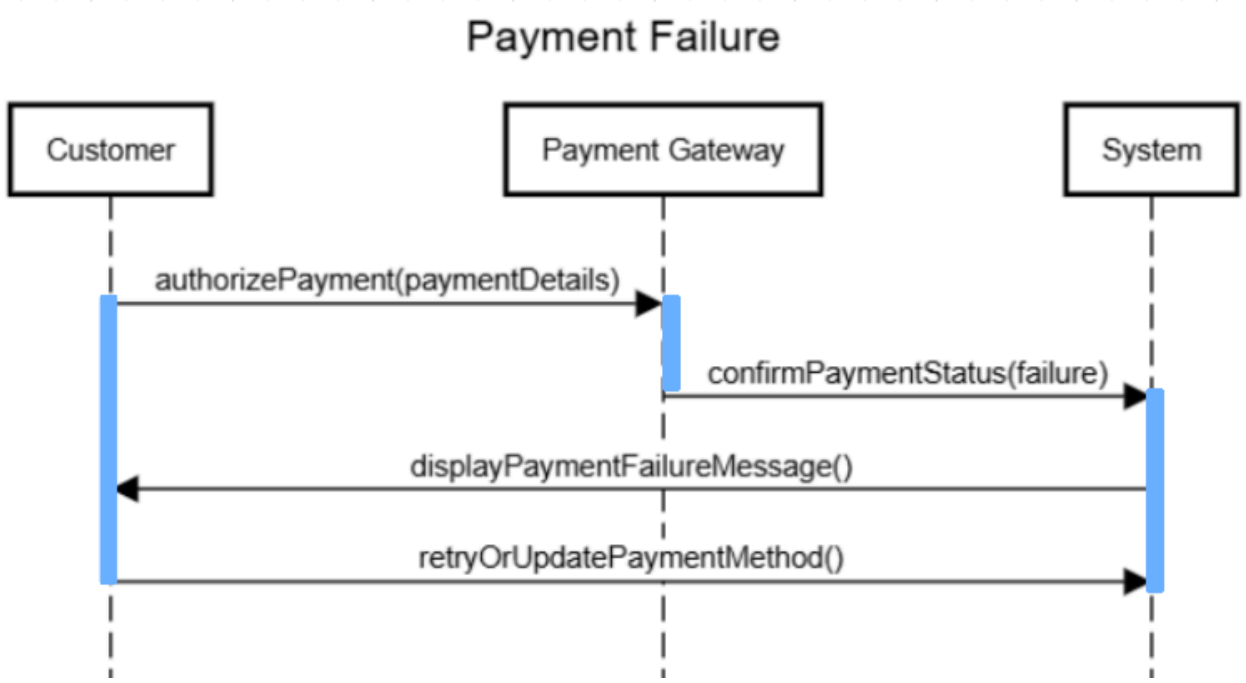
Event 3: Customer Confirms Booking and Initiates Payment



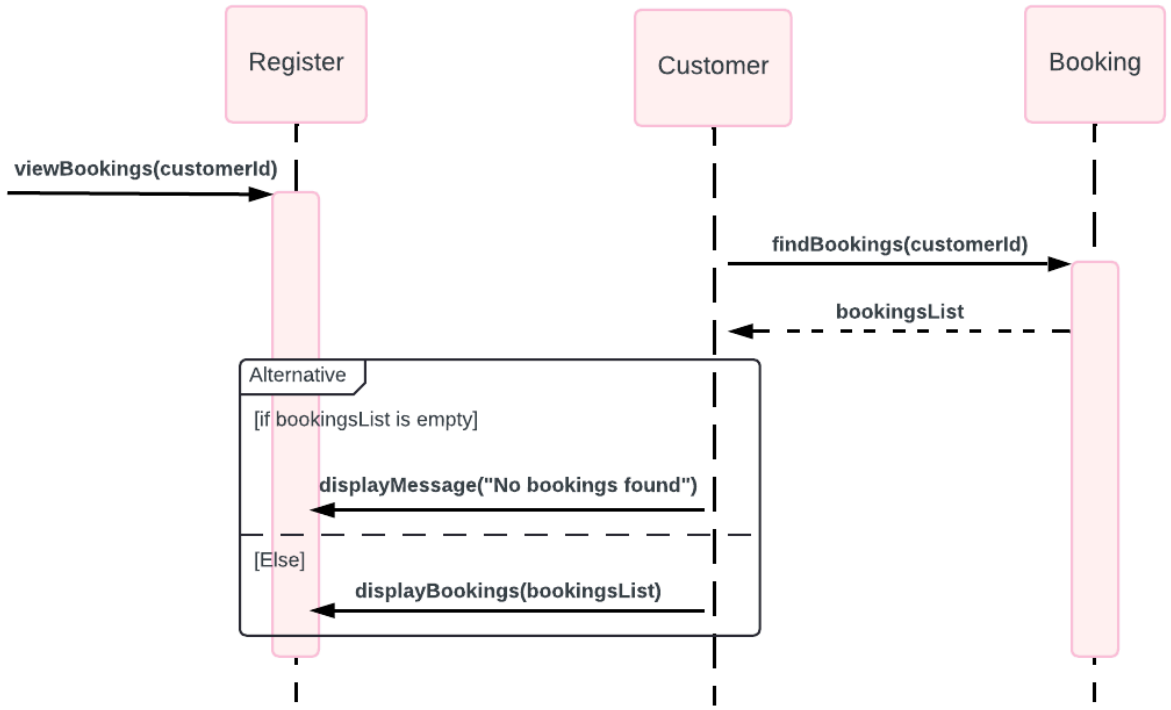
Alternative Scenario 1: Service Provider Not Available



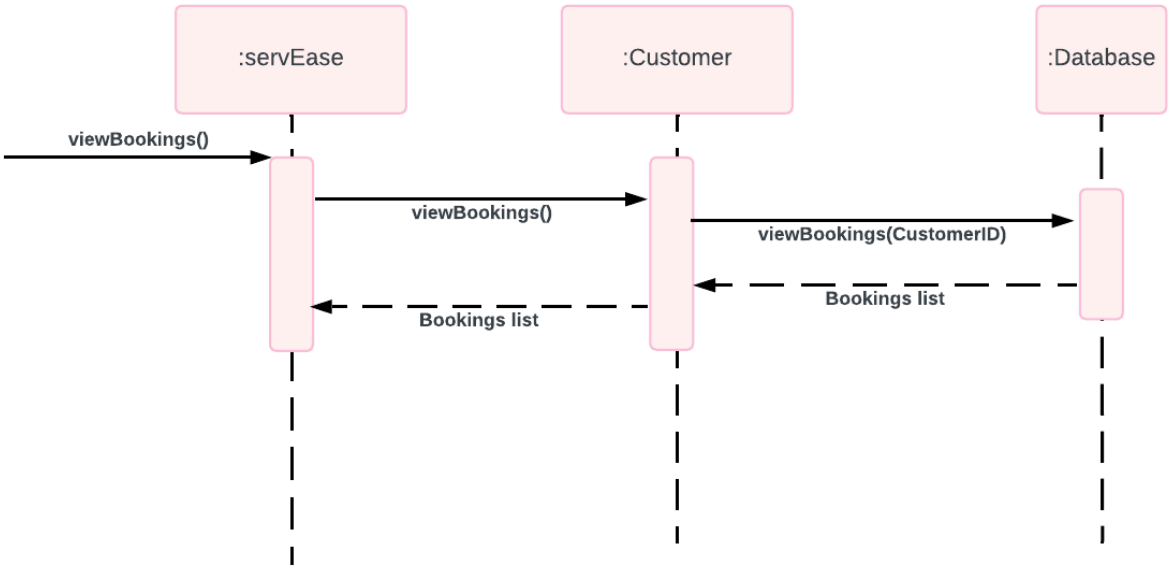
Alternative Scenario 2: Payment Failure



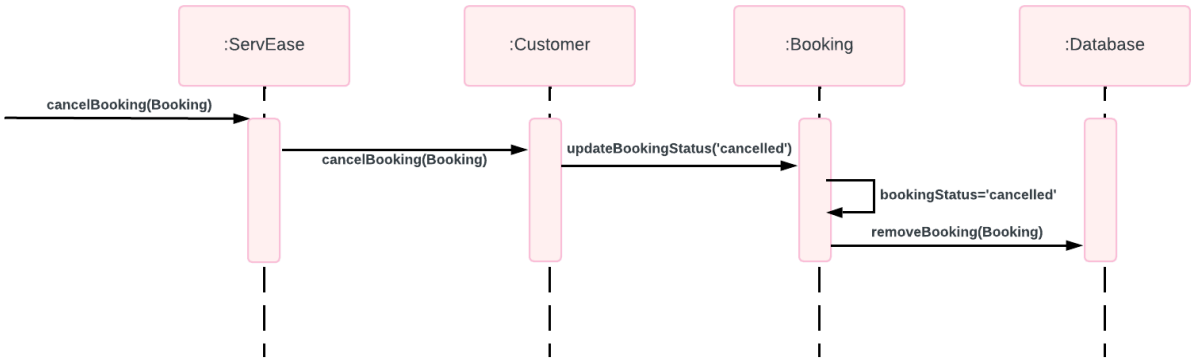
SD for viewBookings



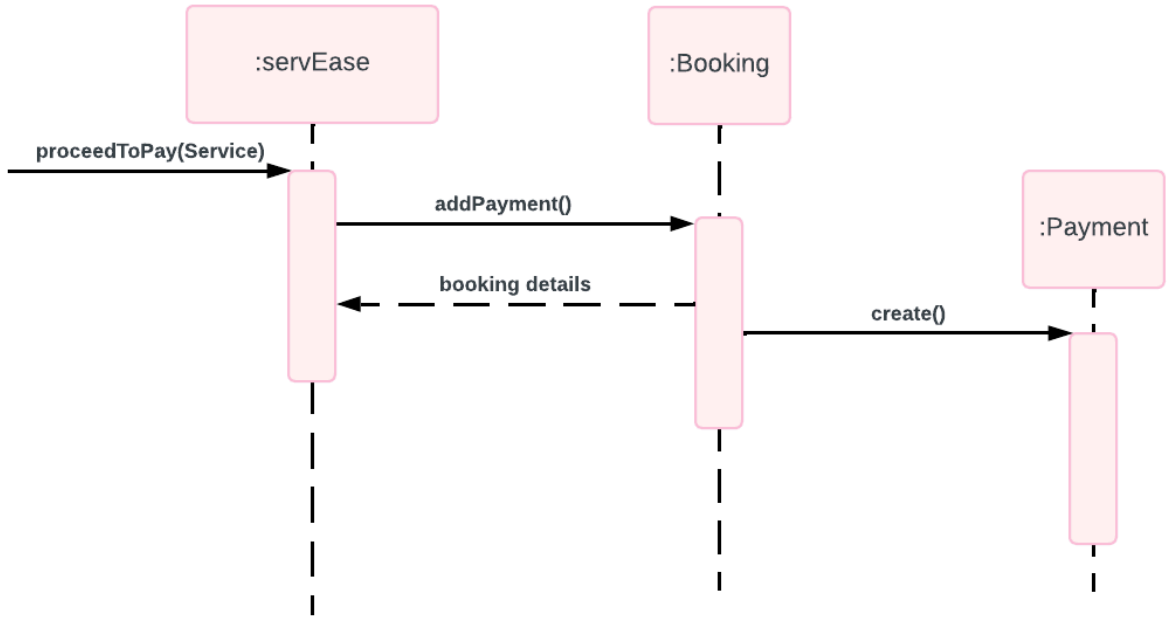
view bookigs



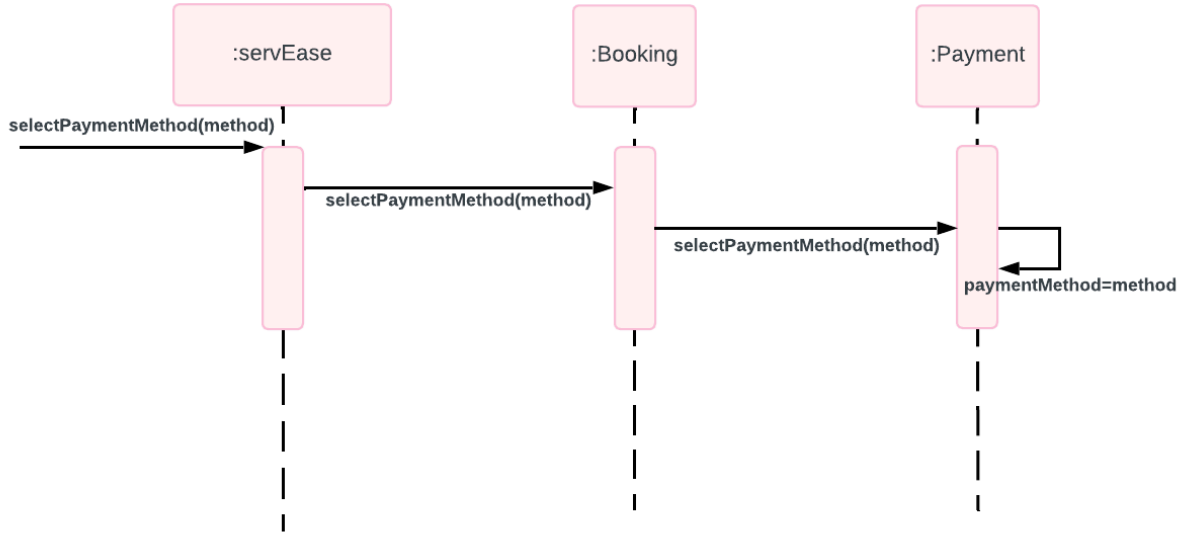
SD for cancelBooking



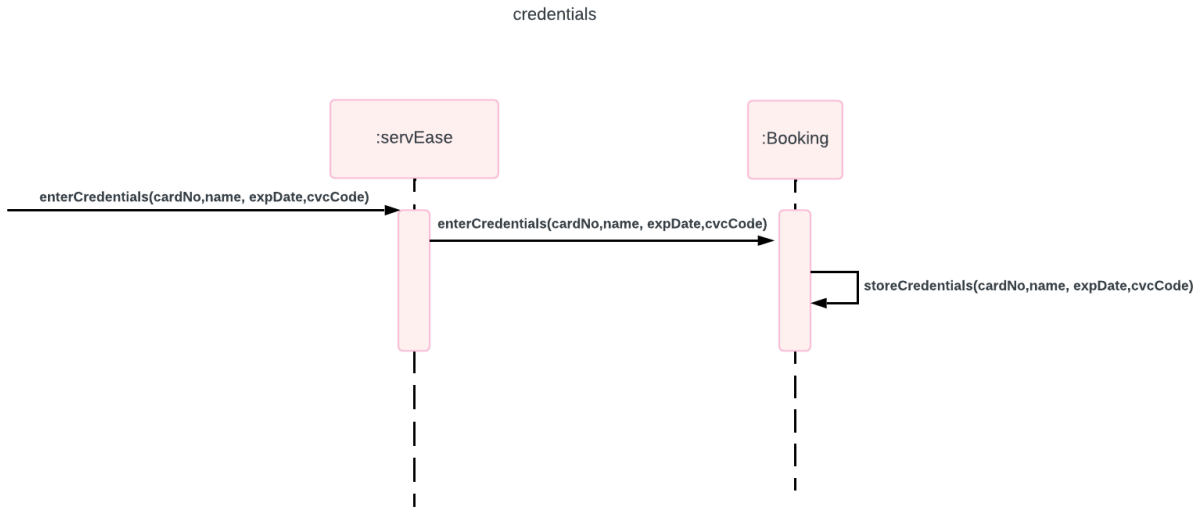
SD for proceedToPay



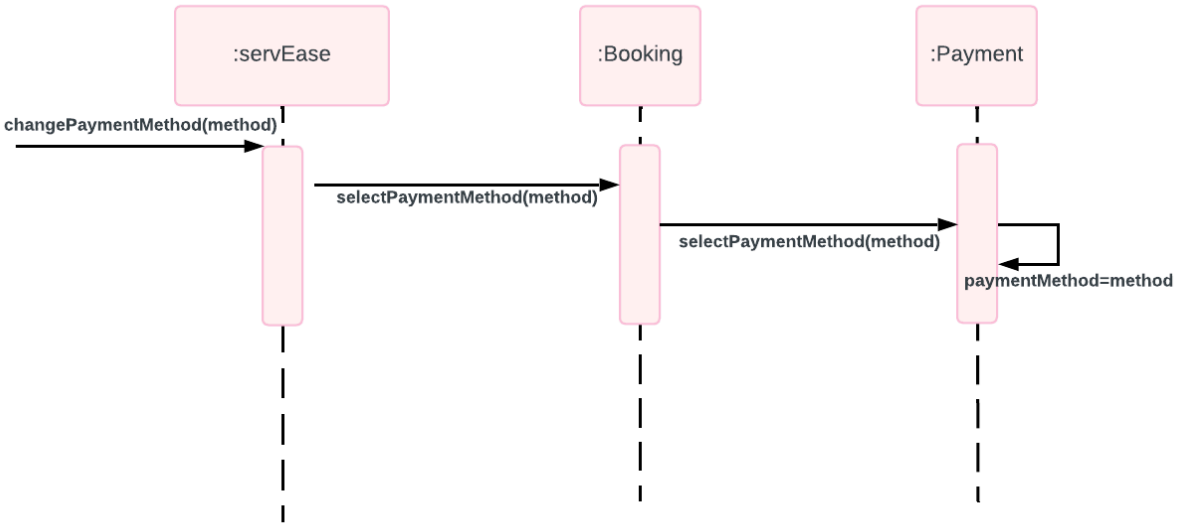
mehod



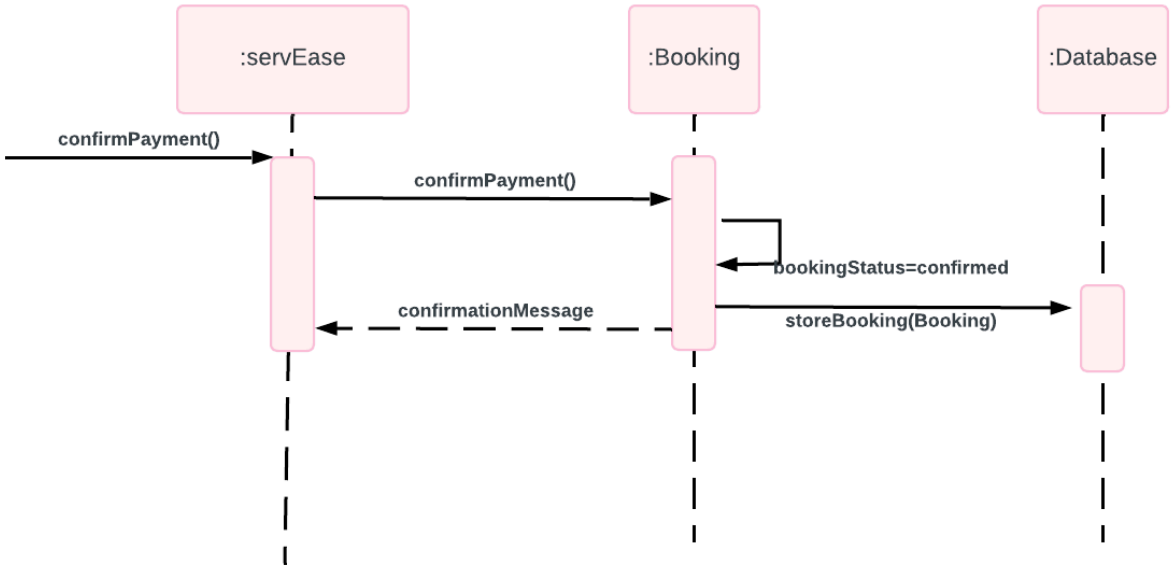
enterCredentials()



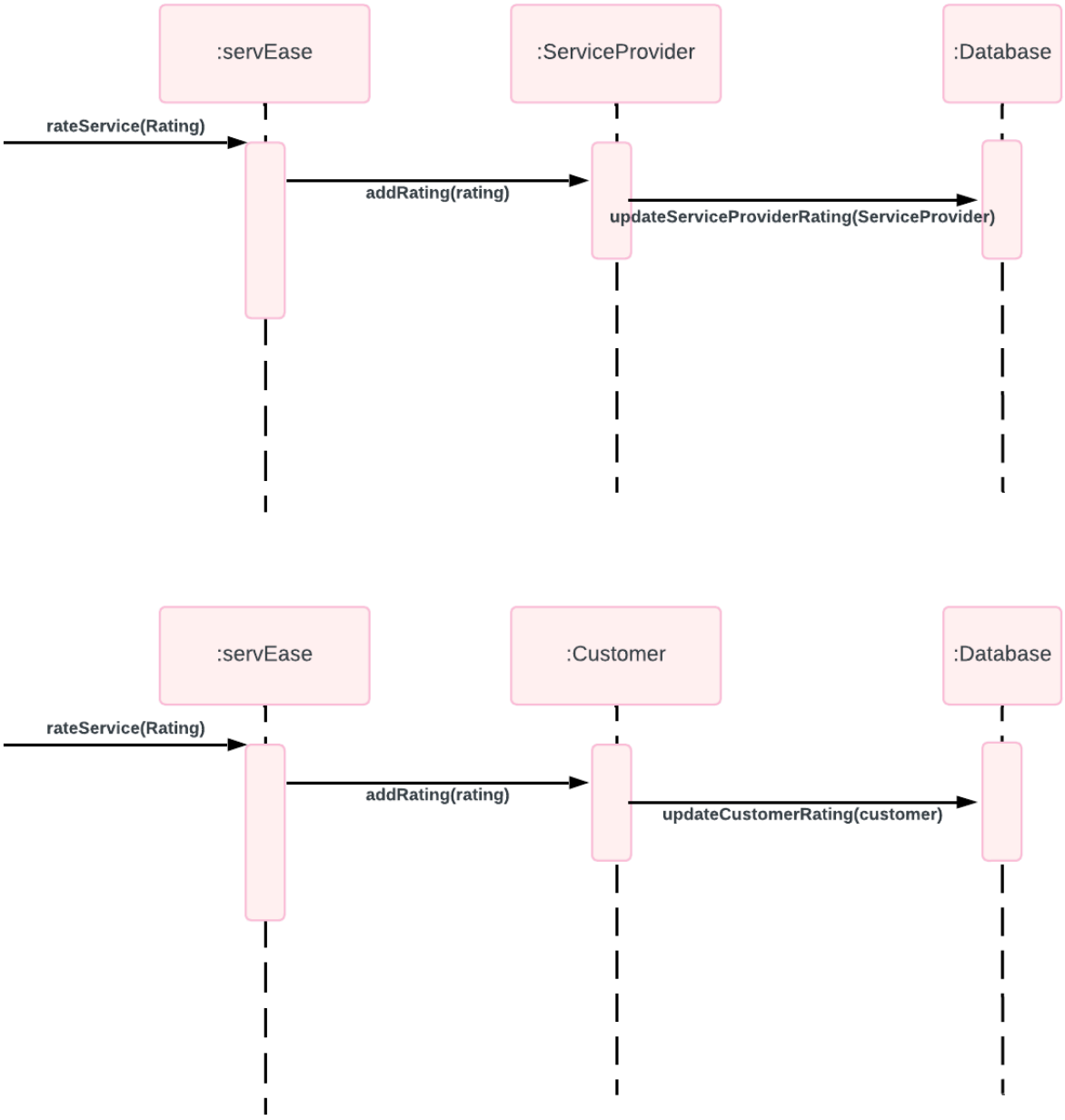
change payment method



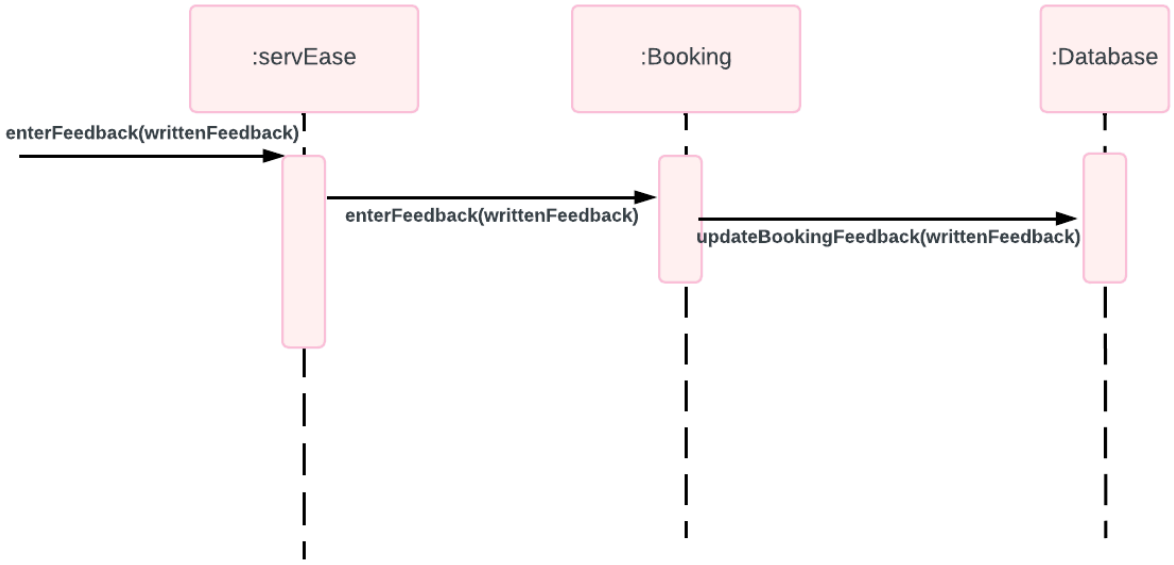
confirm



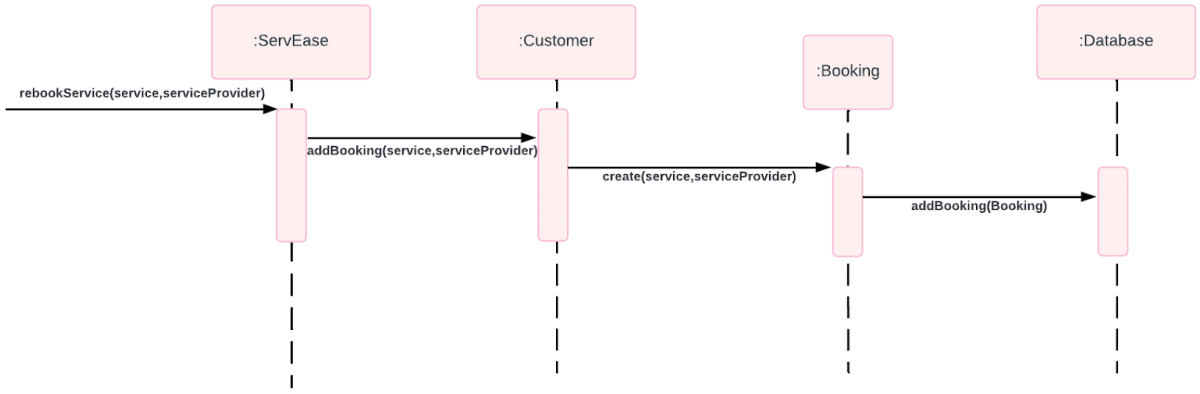
rateService()

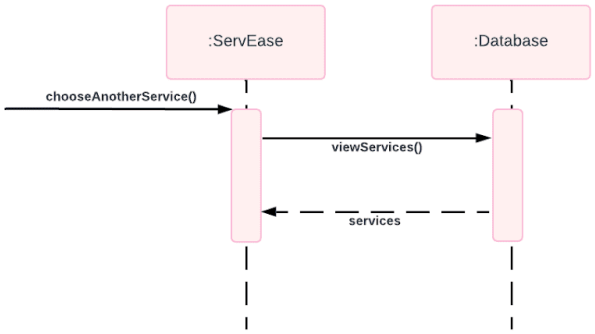


enterFeedback()

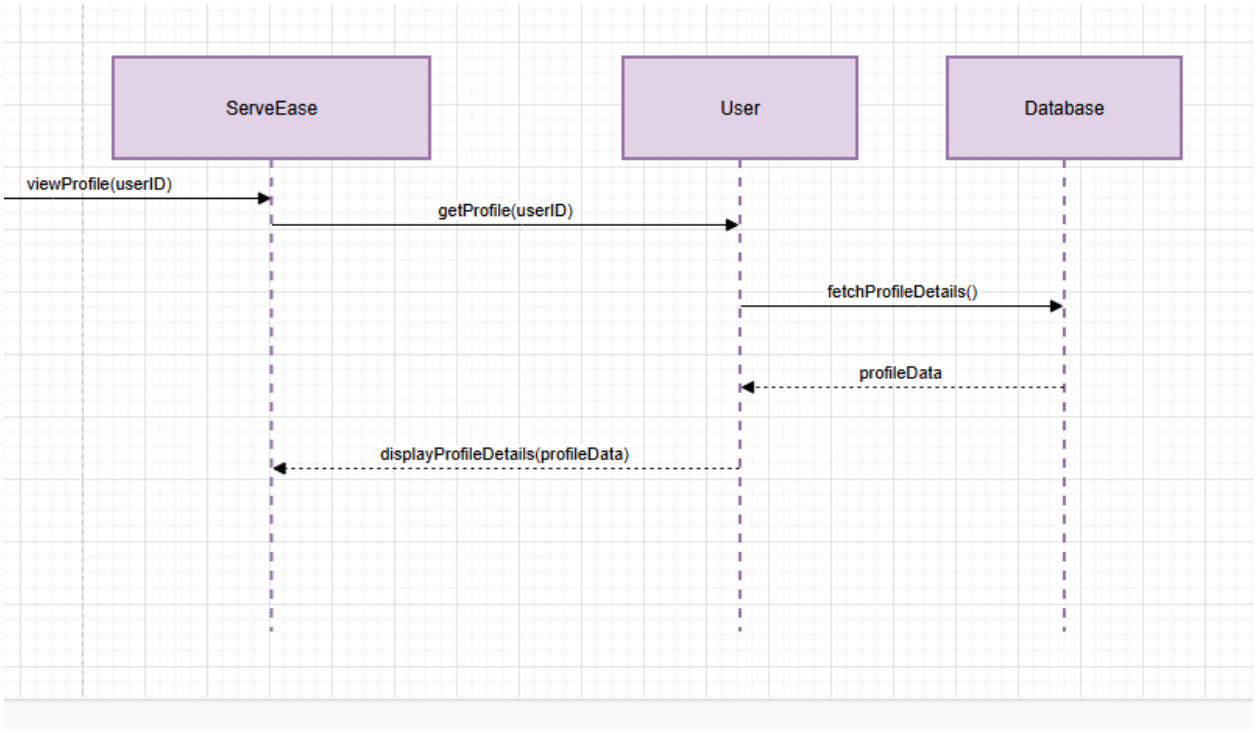


rebookService() and chooseAnotherService()

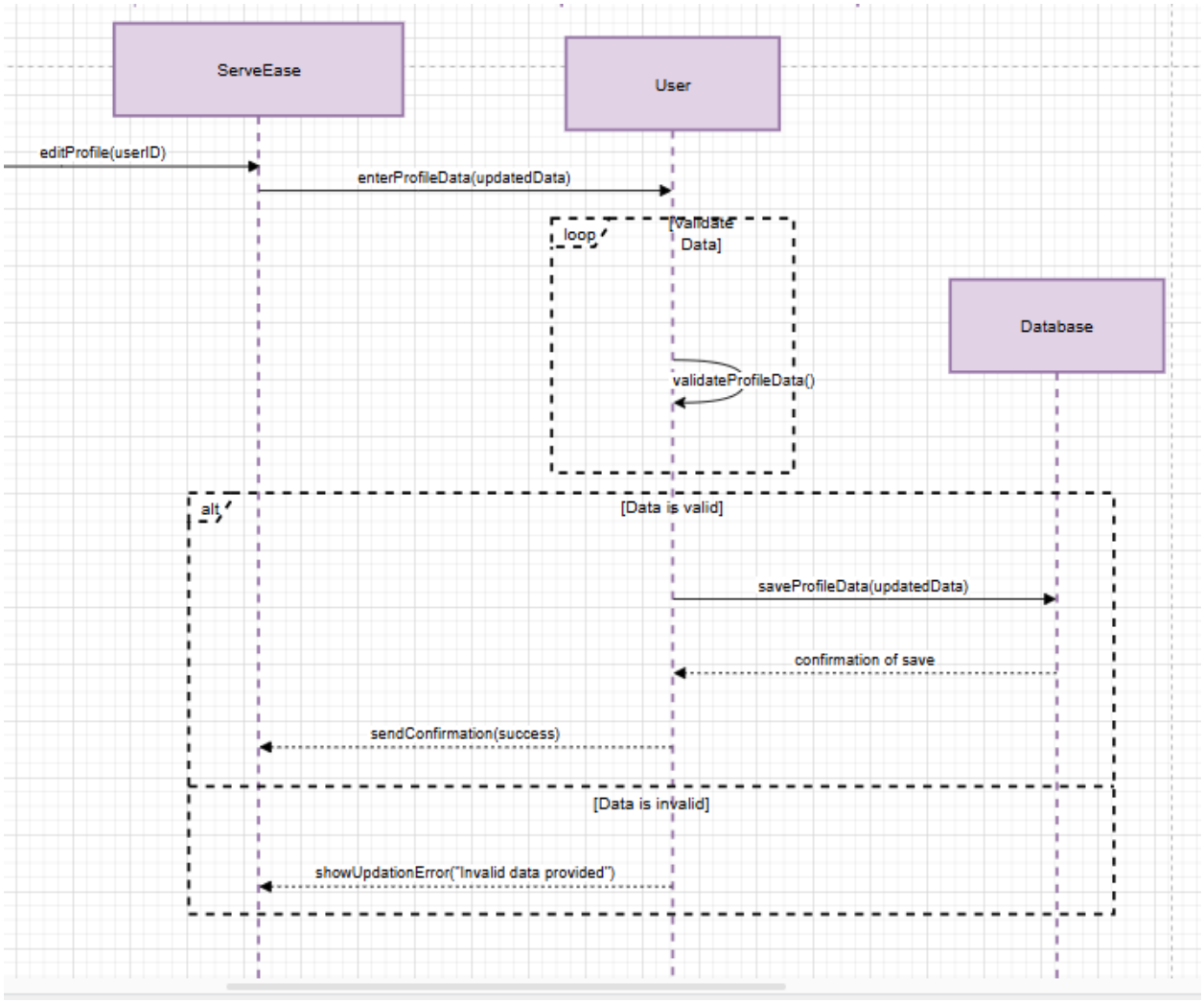




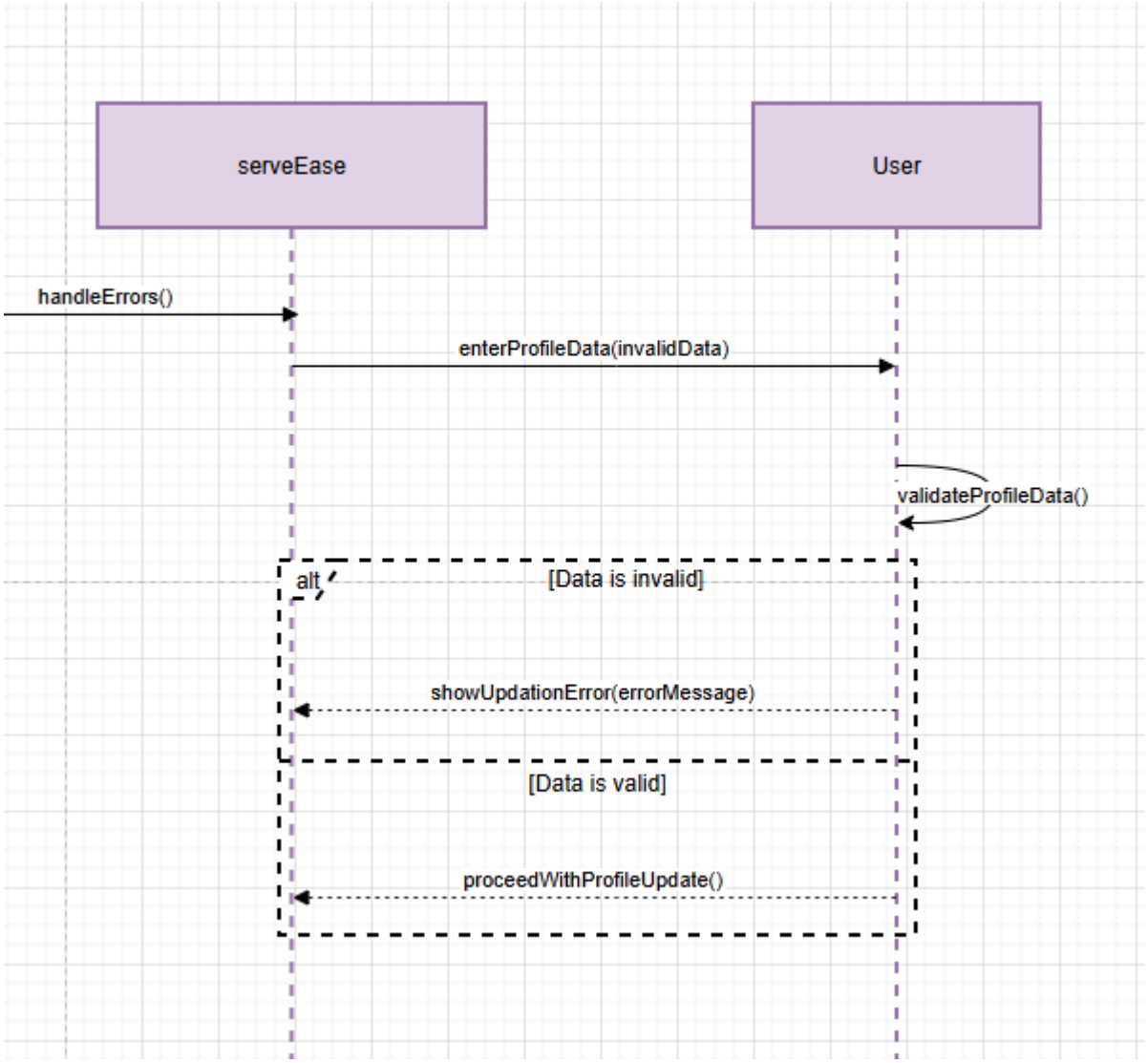
Update Profile
SEQUENCE DIAGRAMS:
EVENT 1: View and Display Profile


ServeEase
User
Database
viewProfile(userID)
getProfile(userID)
fetchProfileDetails()
profileData
displayProfileDetails(profileData)

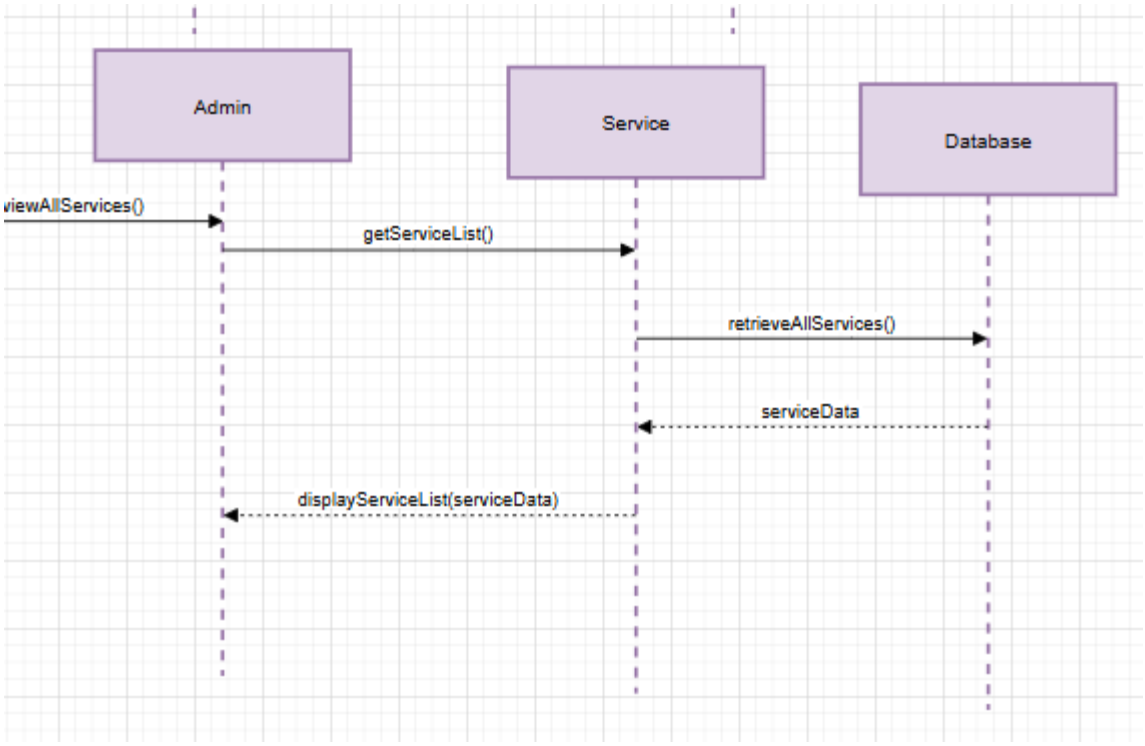
EVENT 2: Edit and Save Profile



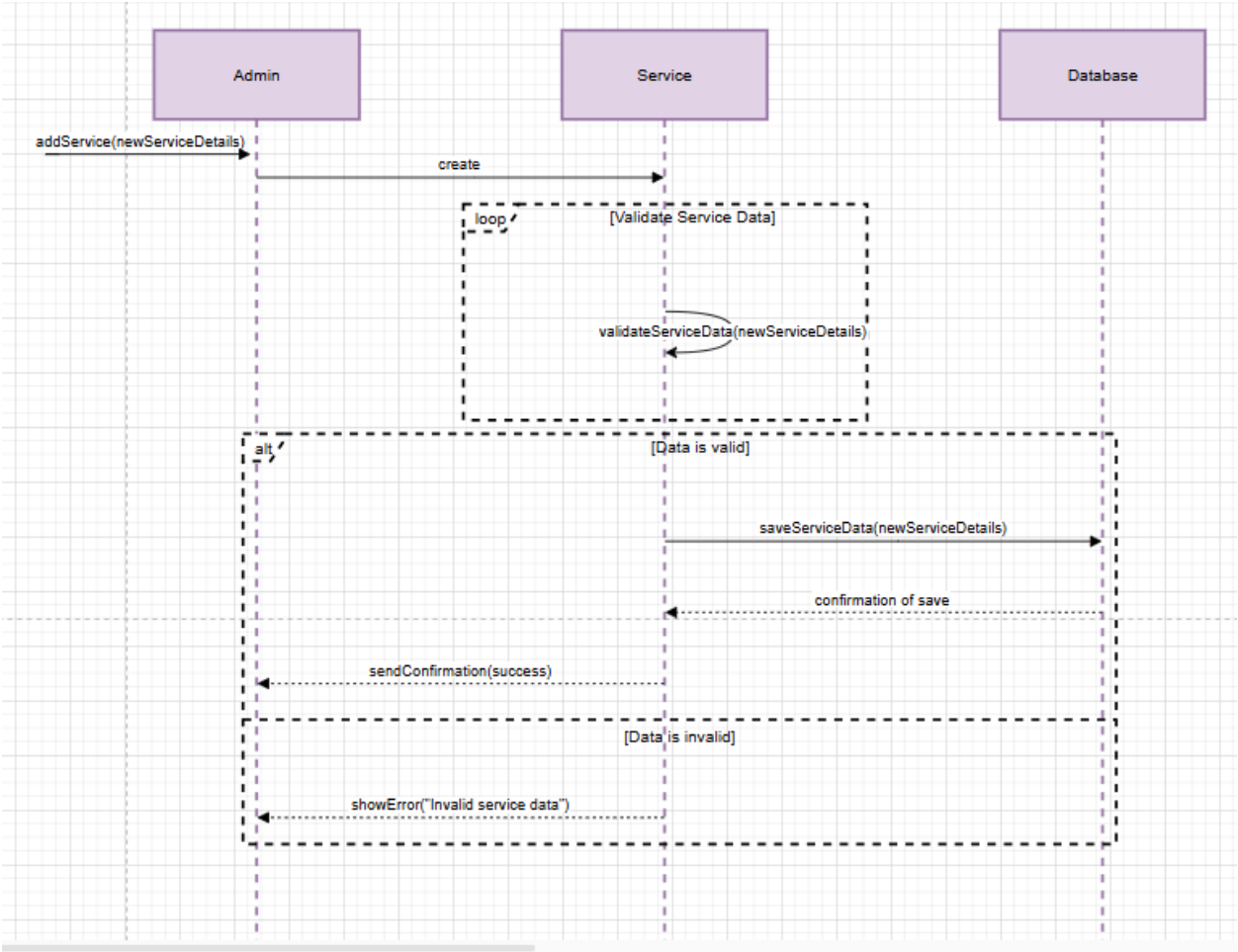
EVENT 3: Handle Errors



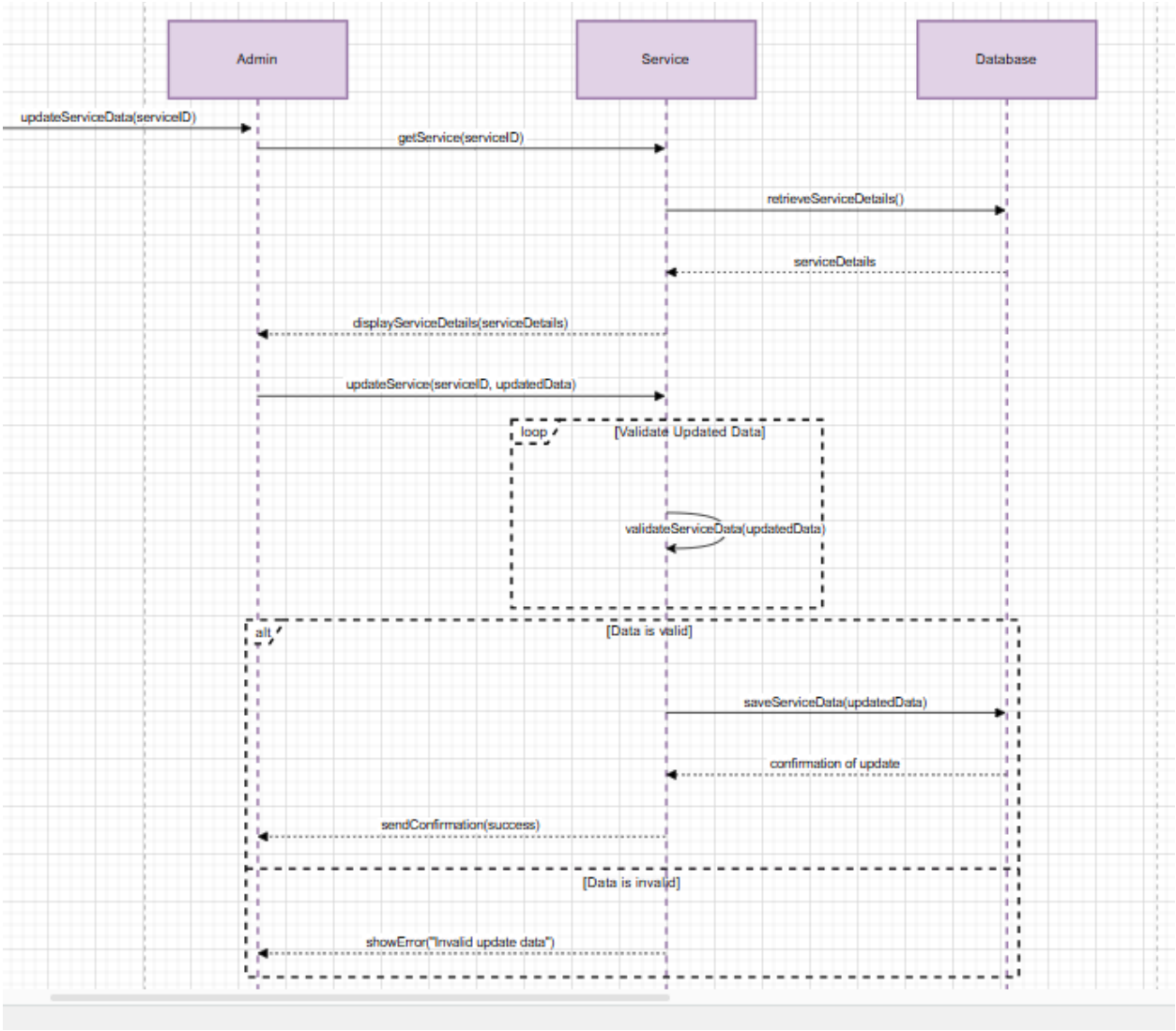
SEQUENCE DIAGRAMS:
EVENT 1: View All Services



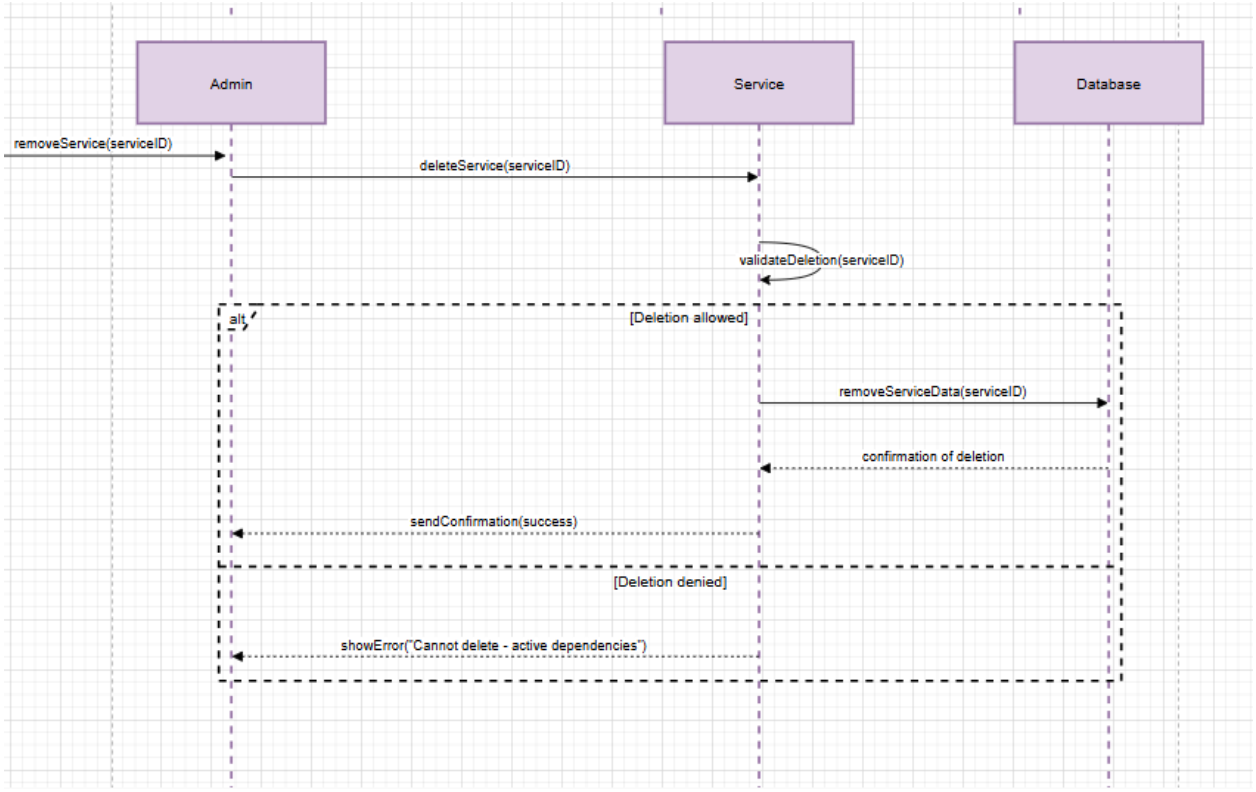
EVENT 2: Add Service



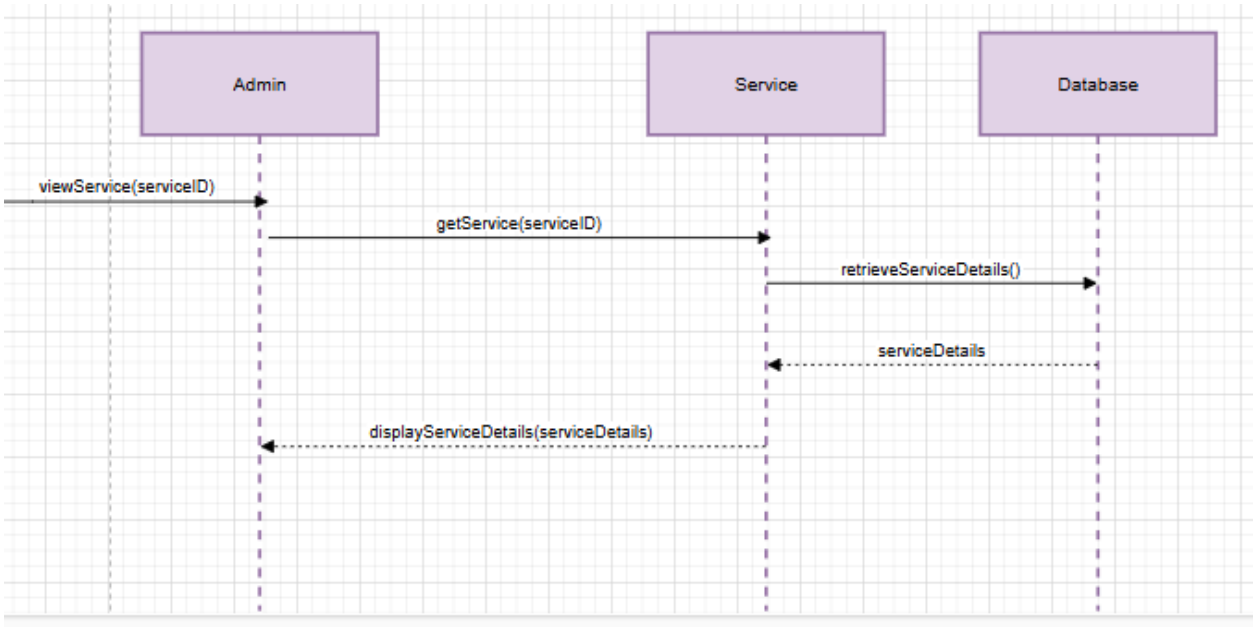
EVENT 3: Update Service


Admin
Service
Database
updateServiceData(serviceID)
getService(serviceID)
retrieveServiceDetails()
serviceDetails
displayServiceDetails(serviceDetails)
updateService(serviceID, updatedData)
loop
[Validate Updated Data]
validateServiceData(updatedData)
alt
[Data is valid]
saveServiceData(updatedData)
confirmation of update
sendConfirmation(success)
[Data is invalid]
showError("Invalid update data")

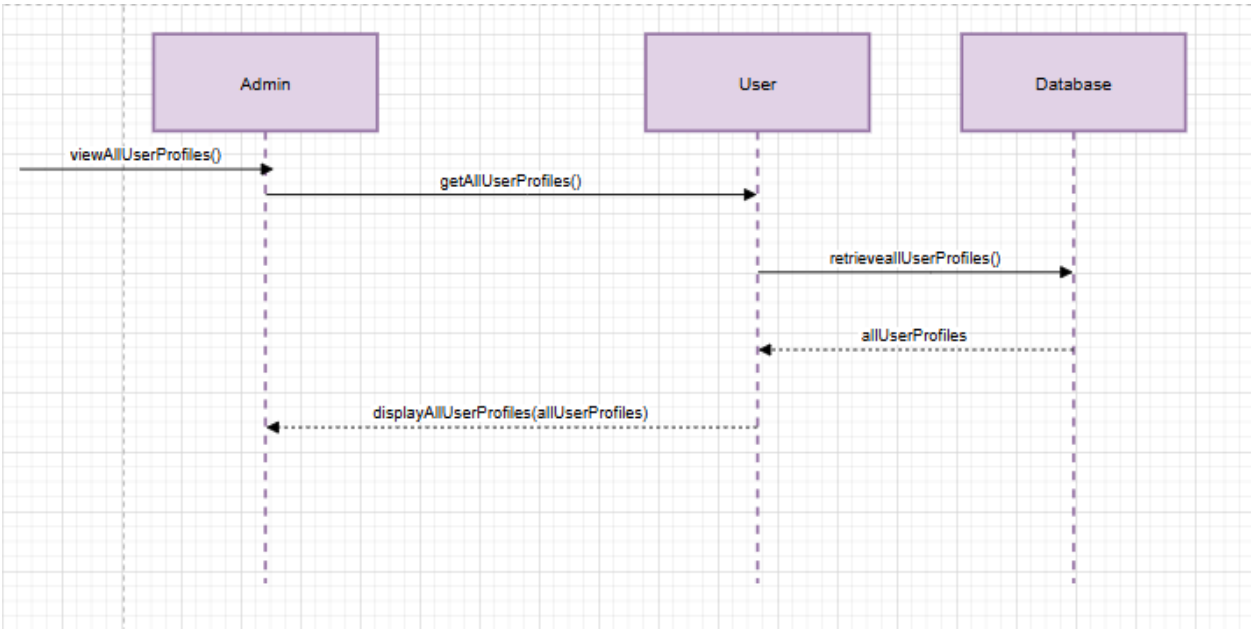
EVENT 4: Delete Service



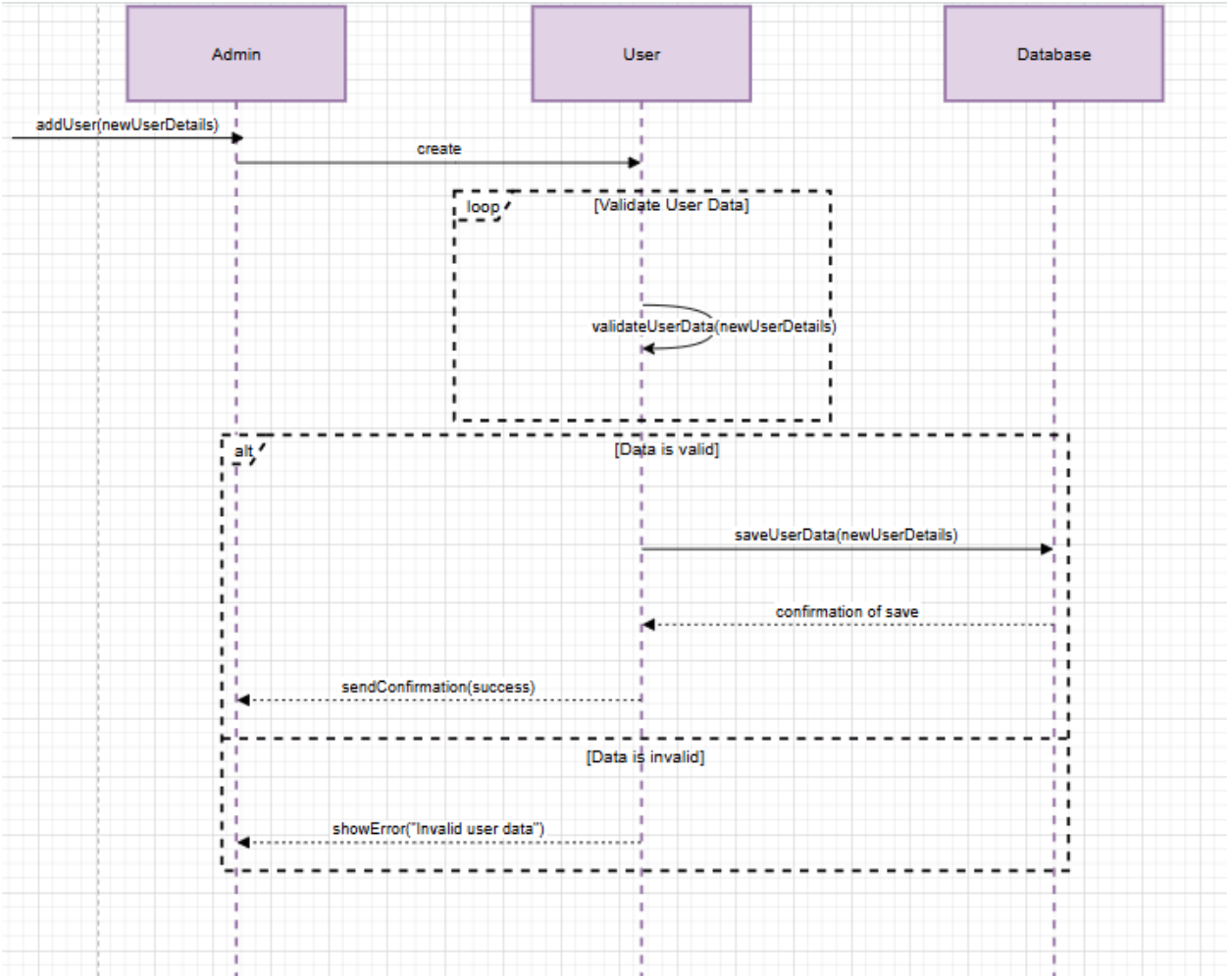
EVENT 5: View and Display Service Details


Admin
Service
Database
viewService(serviceID)
getService(serviceID)
retrieveServiceDetails()
serviceDetails
displayServiceDetails(serviceDetails)

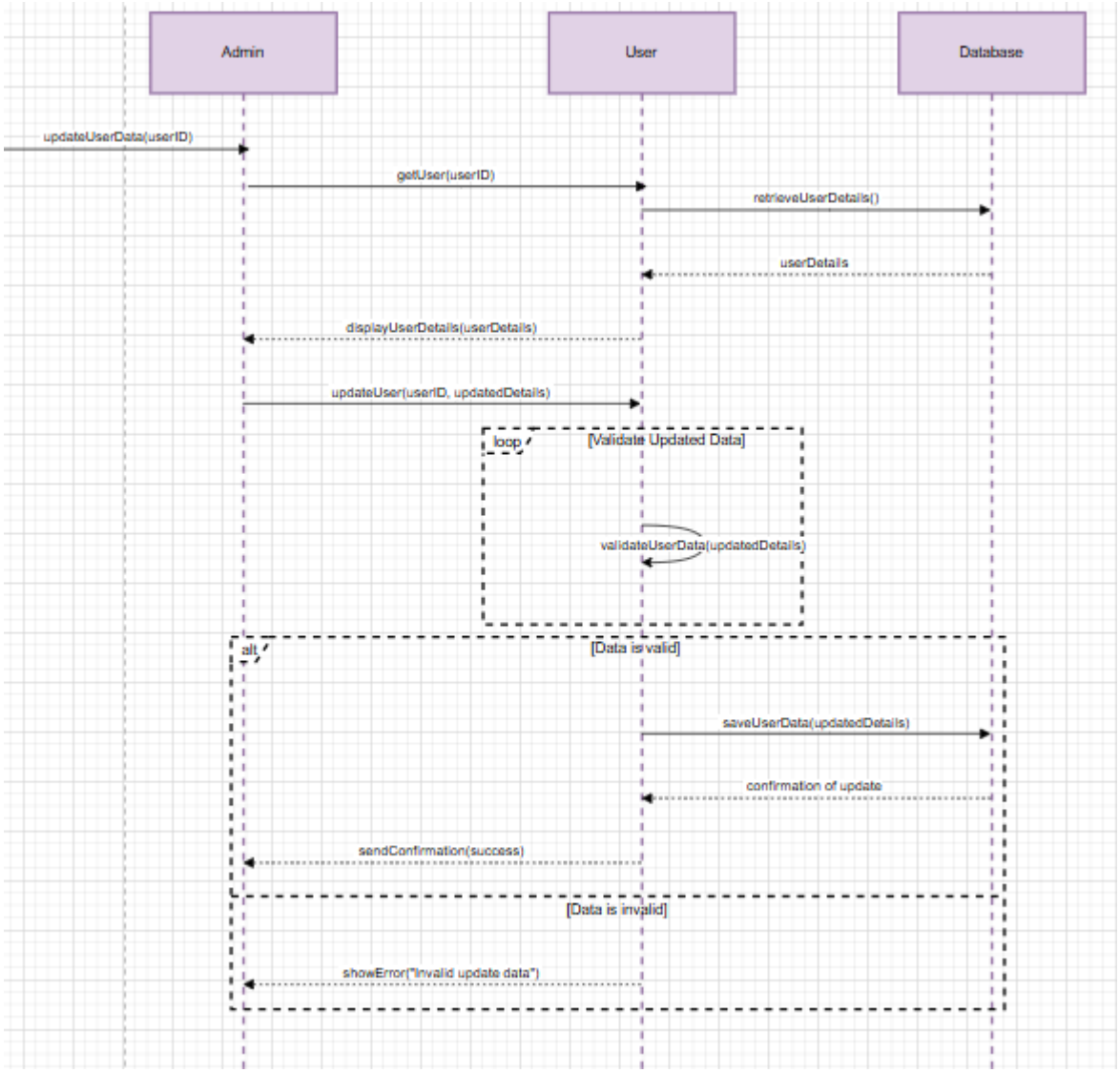
SEQUENCE DIAGRAMS:
EVENT 1: View All Users



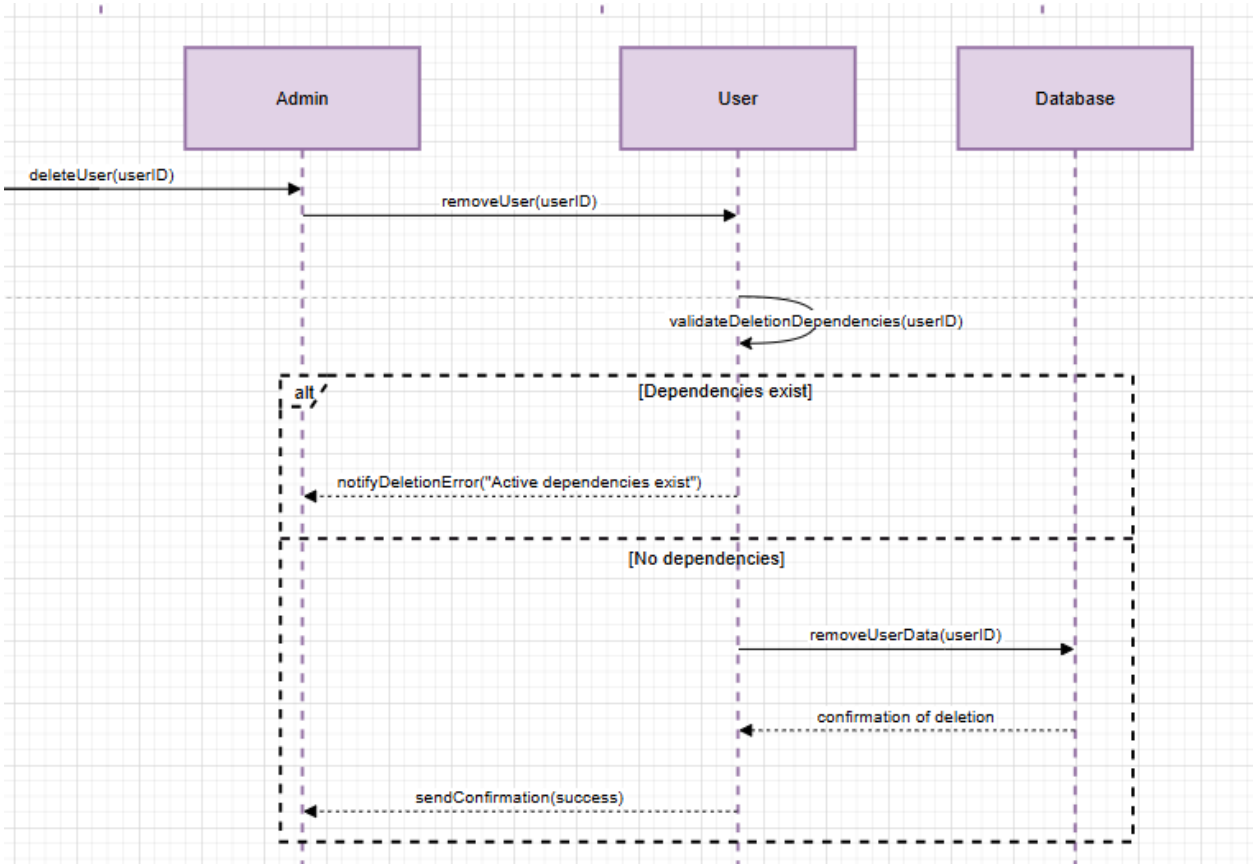
EVENT 2: Add User



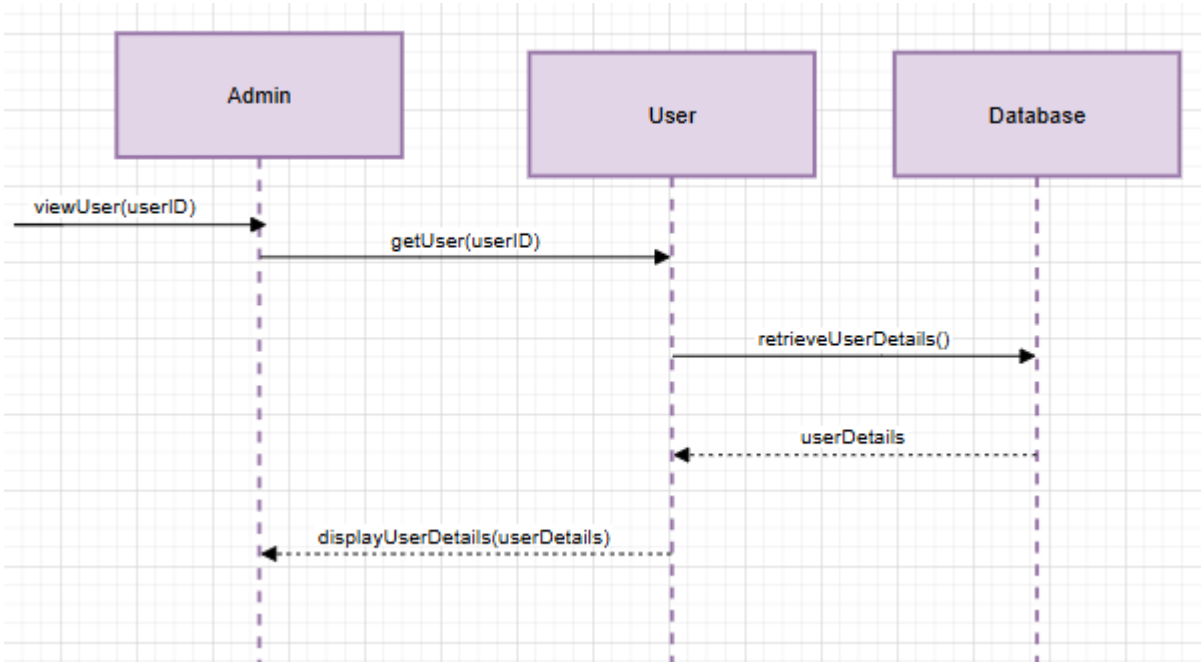
EVENT 3: Update User



EVENT 4: Delete User with Checks

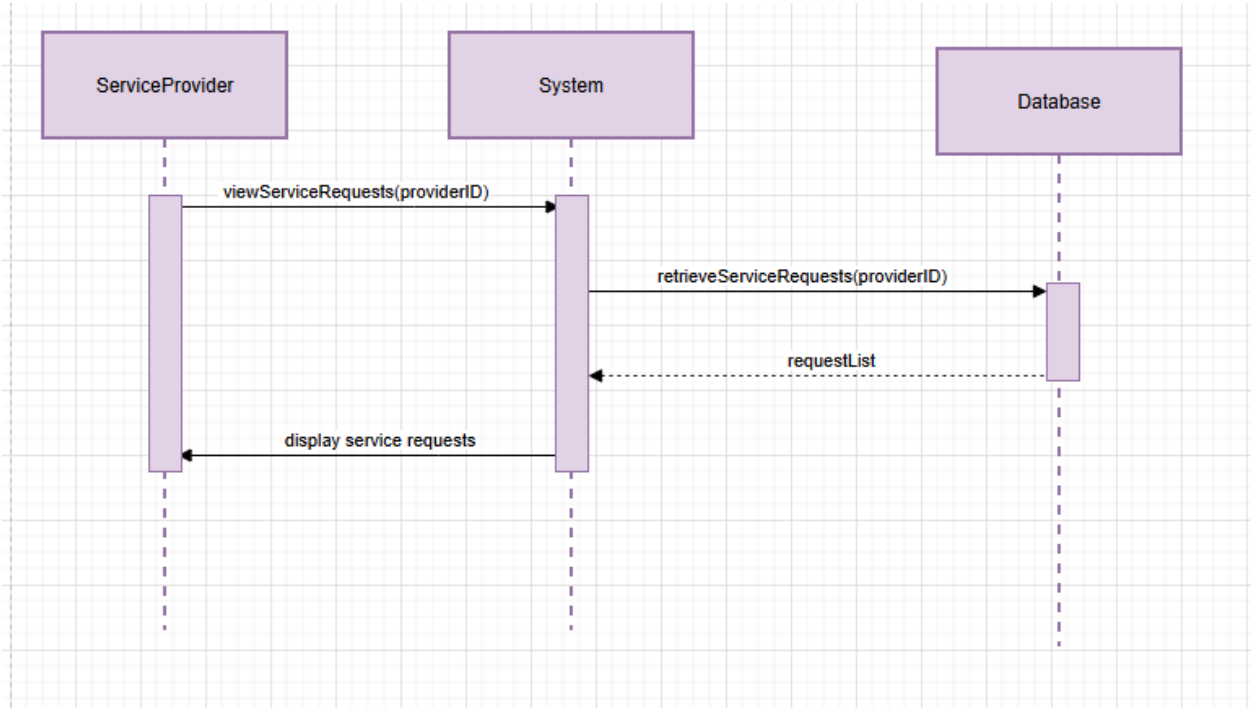


EVENT 5: Display User Details

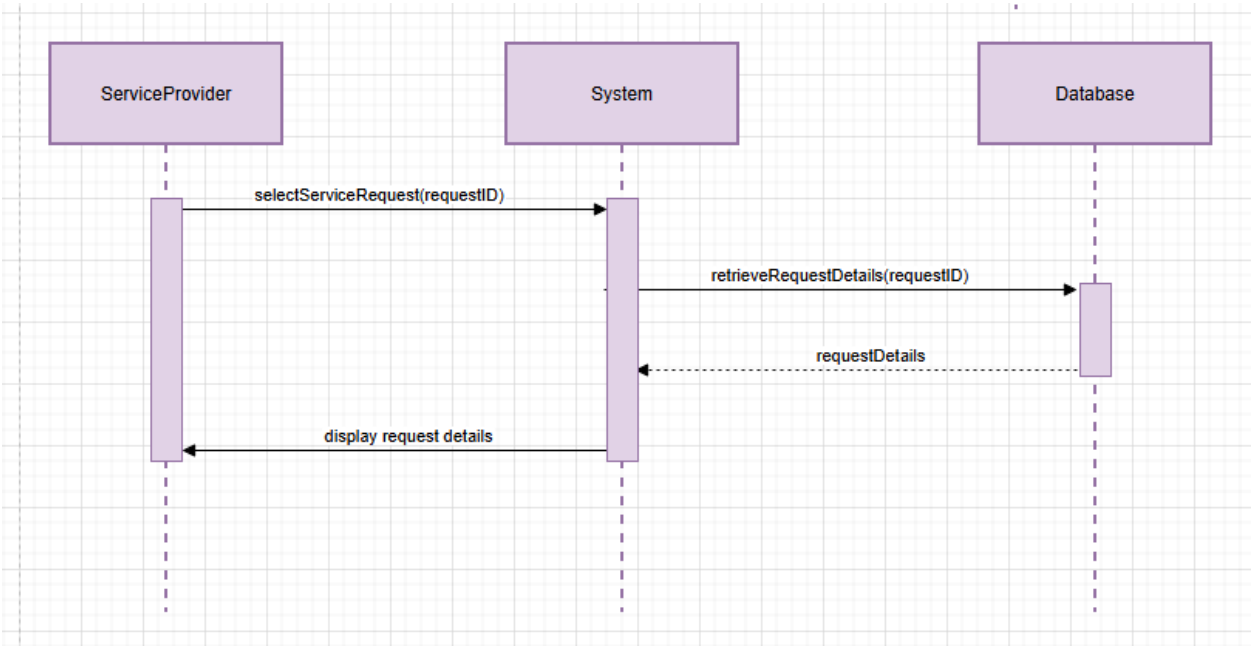


SEQUENCE DIAGRAMS:

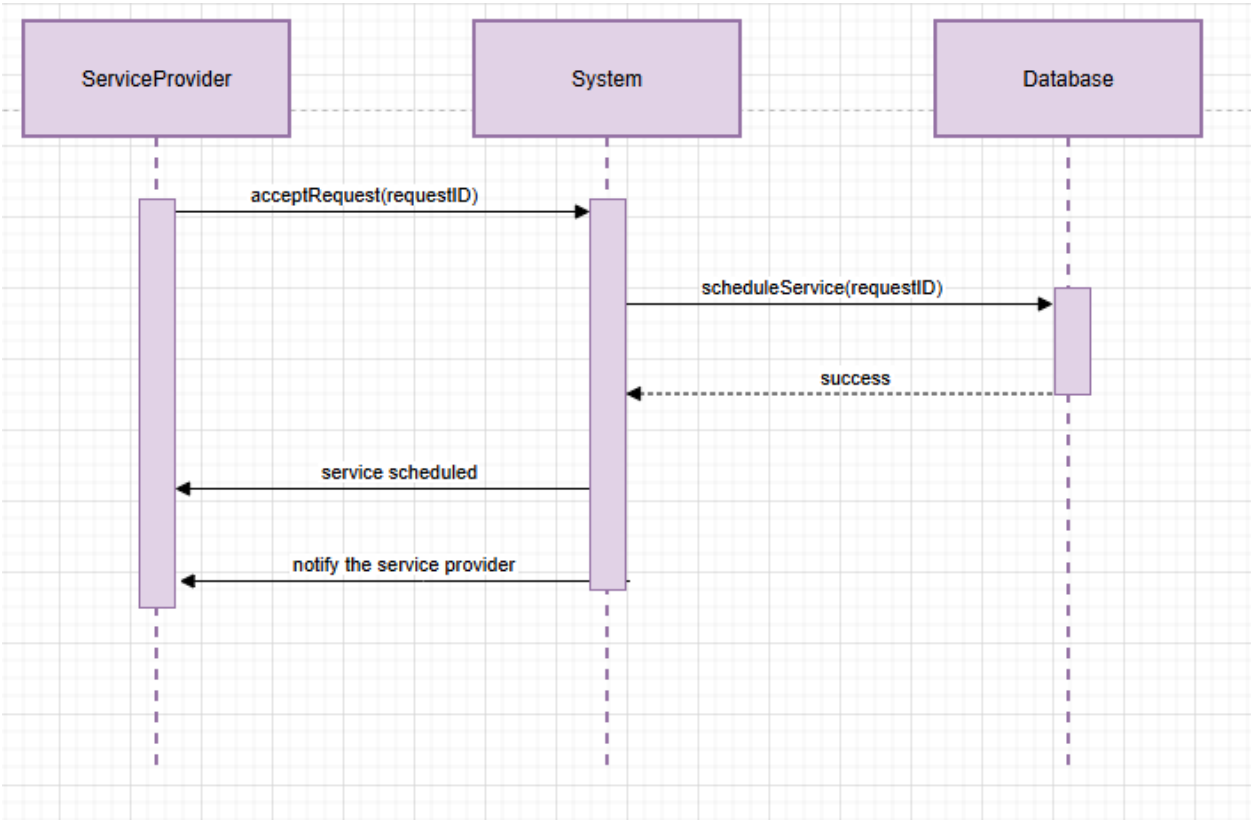
EVENT 1: View Service Requests



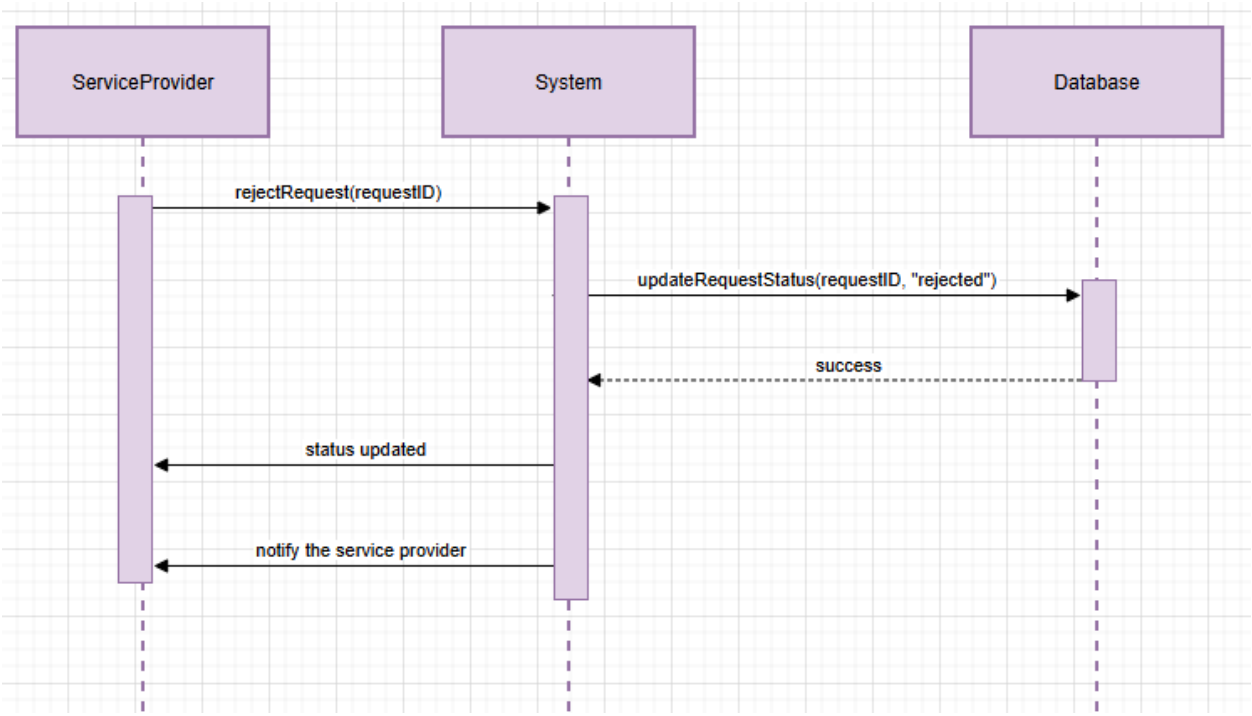
EVENT 2: Select Service Request



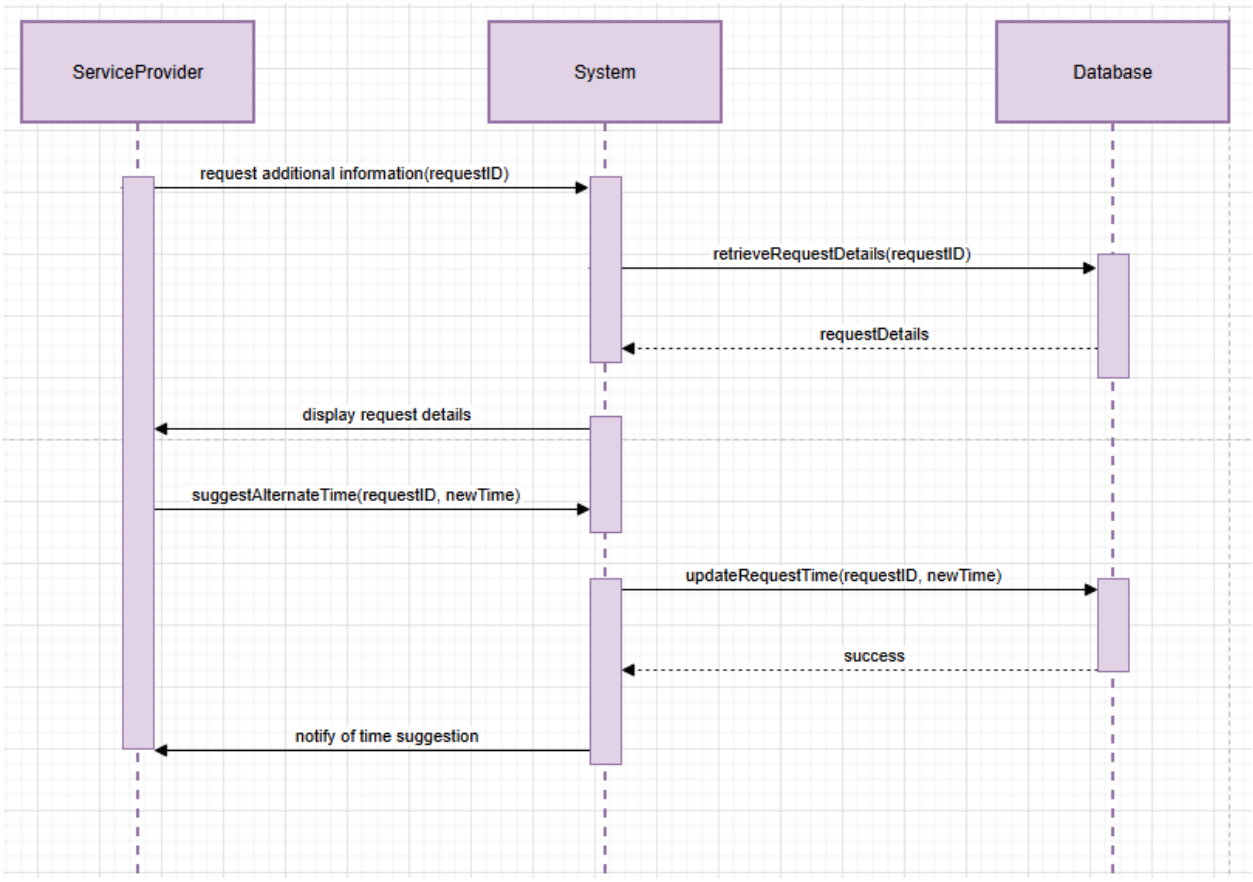
EVENT 3: Accept Service Request



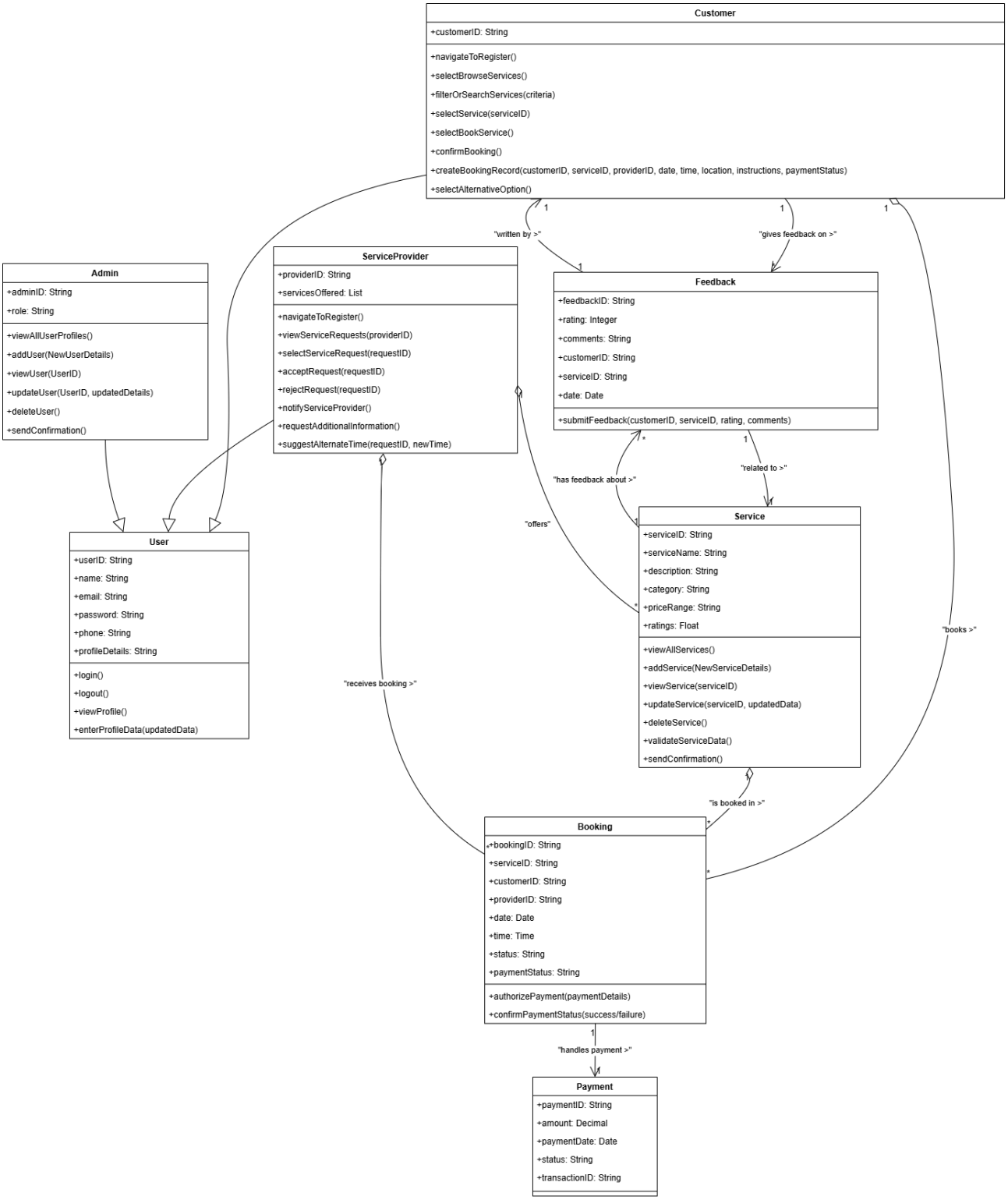
EVENT 4: Reject Service Request



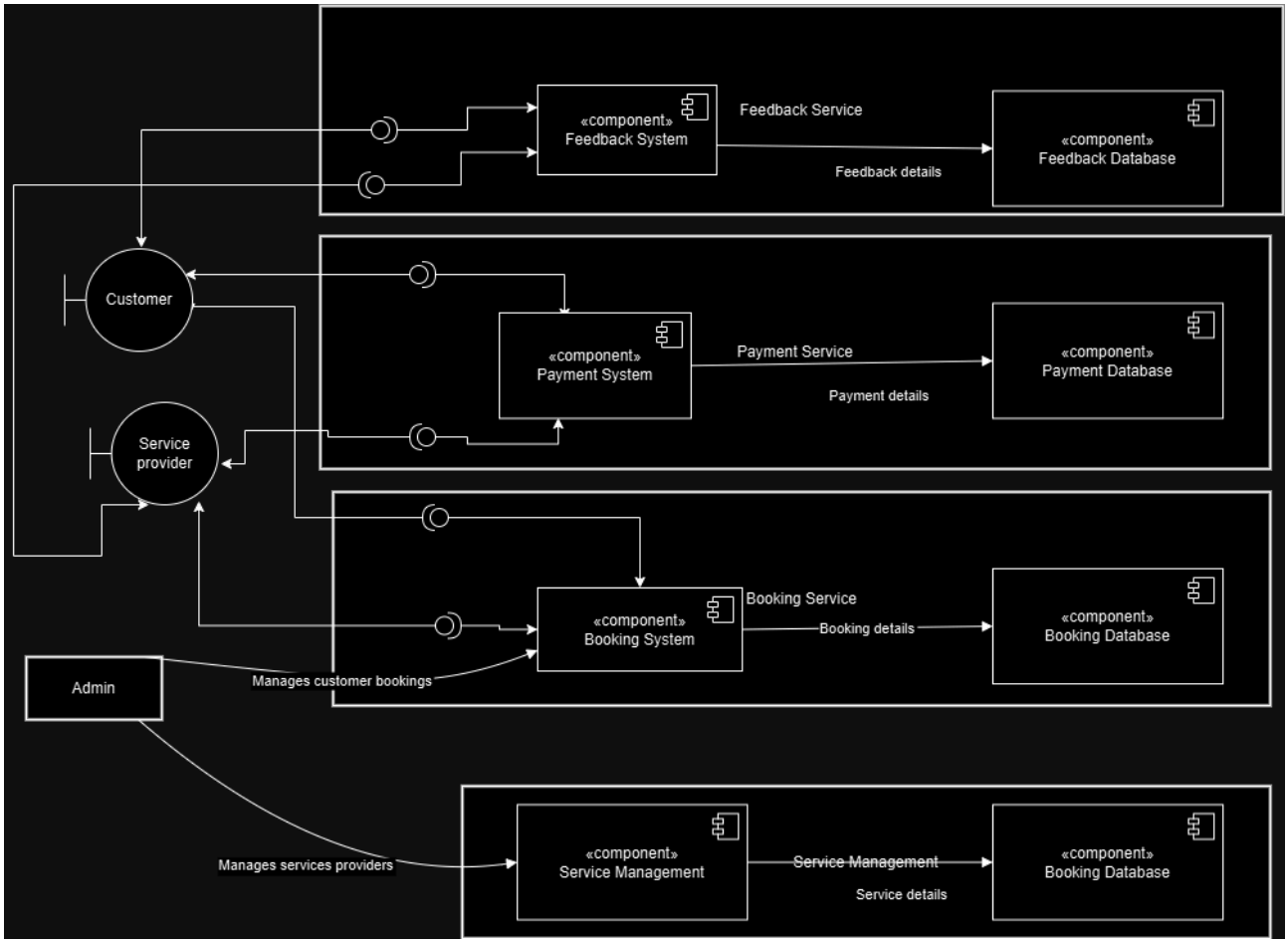
EVENT 5: Request Additional Information / Suggest Alternate Time



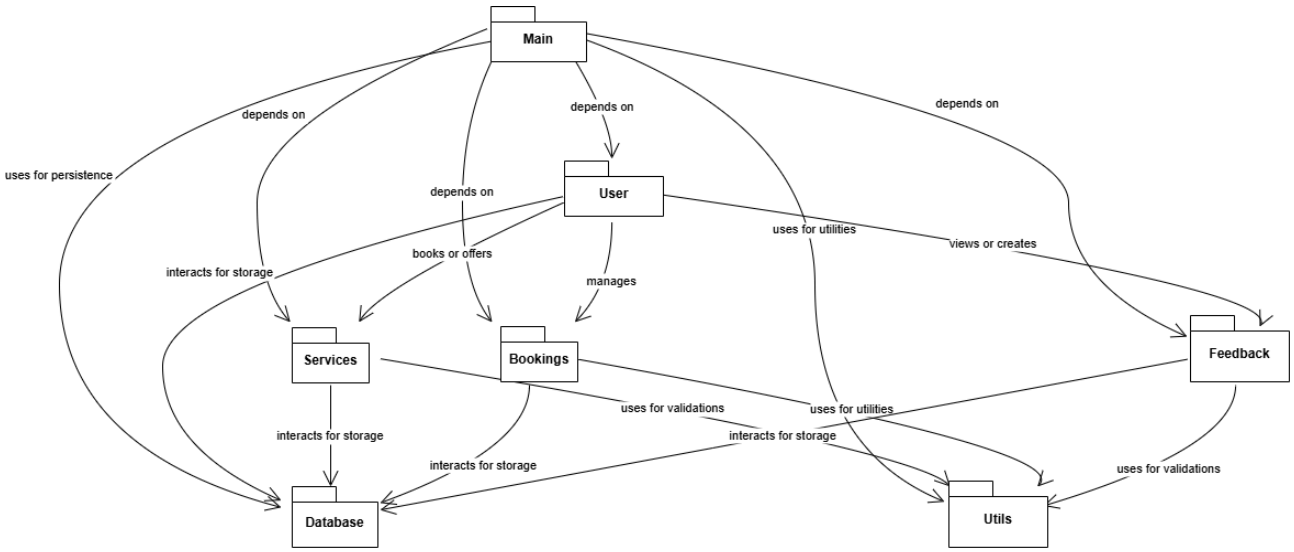
7. Class Diagram



8. Component Diagram



9. Package Diagram



10. Deployment Diagram