Software Design Specifications

BookNGo

Version: [1.01]

Project Code	CS3009
Supervisor	Zulfiqar Ali
Co Supervisor	
Project Team	Armughan Ather 22k-4416 Mohammad Shahmeer 22k-4643 Roohan Ahmed 22k-4611
Submission Date	30th April, 2025

[Instructions]

- No section of template should be deleted. You can write 'Not applicable' if a section is not applicable to your project. But all sections must exist in the final document.
- All comments/examples mentioned in square brackets ([]) are in the template for explanation purposes and must be replaced / removed in final document.
- This' Instruction' section should also be removed in final document.
- MS-Word Reviewing feature must be used to get the document reviewed by PMs or supervisors.

Document History

[Revision history will be maintained to keep a track of changes done by anyone in the document.]

Version	Name of Person	Date	Description of change
1.01	Armughan Ather – Mohammad Shahmeer – Roohan Ahmed	30th April, 2025	Document Created

Distribution List

Name	Role
Zulfiqar Ali	Supervisor
N/A	Co Supervisor

Document Sign-Off

Version	Sign-off Authority	Project Role	Signature	Sign-off Date

Document Information

Category	Information
Customer	FAST-NU
Project	BookNGo
Document	Software Design Specification
Document Version	1.0
Status	Draft
Author(s)	Armughan Ather, Mohammad Shahmeer, Roohan Ahmed
Approver(s)	
Issue Date	30th April, 2025
Document Location	GCR
Distribution	Advisor Project Coordinator's Office (through Advisor)

Definition of Terms, Acronyms and Abbreviations

[This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly.]

Term	Description
ASP	Active Server Pages
DD	Design Specification

Term	Description

Table of Contents

1 Introduction

1.1 Purpose of Document

The purpose of this document is to outline the Software Requirements Specification (SRS) for the BookNGO project. It provides a detailed description of the functional and non-functional requirements of the system, including the user interactions and system behavior.

1.2 Intended Audience

This document is intended for the following audience:

1. **Group Members**: To ensure all team members have a unified understanding of the project requirements and objectives.
2. **Course Instructor/Evaluator**: To evaluate the project against the course objectives and requirements.
3. **Peers/Reviewers**: To provide feedback and suggestions during project presentations and discussions.
4. **Future Group Projects**: As a reference for students or teams working on similar projects.

1.3 Document Convention

Font: Arial
Font Size: 12

1.4 Project Overview

The system is a web-based travel booking platform designed to allow users to seamlessly search, book, and manage accommodations and flights. It supports both customer-facing features and administrative functionalities, aiming to replicate the core user experience of platforms like Booking.com while maintaining a more academic and functional scope suitable for educational purposes.

The platform provides registered users with the ability to search for flights and hotels based on criteria such as location, date, price, and rating. Once search results are displayed, users can proceed to book flights or hotel rooms by providing passenger or guest details and completing payments through a simulated payment gateway. The system ensures booking confirmations are recorded and sent to the users via email.

In addition to bookings, the system also allows users to cancel or modify existing reservations, view their booking history, and rate the services they have availed. These features ensure a complete and user-friendly booking lifecycle.

For administrative users, the system includes capabilities to log in securely and manage accommodation listings. Admins can add new hotels, update existing ones, and manage other related data. This ensures that the system's offerings remain dynamic and up to date.

The system is developed using **React.js** for the frontend and **Node.js with Express** for the backend. It uses **MySQL** as the primary database, managed through **XAMPP**. APIs are used to handle client-server communication, and tools like **Papyrus UML** and **Postman** were utilized for design and testing support.

This project serves as a complete information system solution that integrates user interface design, database interaction, backend logic, and administrative control—all essential components of a modern Management Information System (MIS).

1.5 Scope

BookNGO will provide the following functionalities:

- **User Features**:
  - Search flights, hotels and packages.
  - View services ratings.
  - Book flights, hotels, and travel packages.
  - View booking history.
  - Modify or cancel existing reservations.
  - Rate availed services.
- **Admin Features**:
  - Manage hotels, flights, and airlines (view, create, edit).

Project Boundaries:

- The platform will focus on travel booking and management.
- It will not include additional features such as loyalty programs, advanced analytics, or real-time updates on flight delays in this phase.

2 Design Considerations

This section outlines the foundational design principles, considerations, and constraints that influence the architecture and structure of the travel booking system. These factors have been identified to ensure that the resulting system is maintainable, scalable, and adaptable to future requirements and technological developments.

2.1 Assumptions and Dependencies

- It is assumed that all users will access the system via modern web browsers that support HTML5, CSS3, and JavaScript.
- The design assumes a stable internet connection for real-time data retrieval, booking, and user interactions.
- The system design depends on the availability and correct configuration of backend services such as the Node.js server and MySQL database.
- It is assumed that the Payment Gateway, although simulated, functions as intended for testing purposes and will not require integration with a real banking API.
- User authentication and session handling are assumed to be implemented securely and correctly at the backend level.
- Admin users are assumed to have exclusive rights to manage accommodation and flight data, and appropriate access control mechanisms are expected to be enforced.
- The system is designed for small to medium scale usage (e.g., student project use) and may require re-architecture for enterprise-level deployment.

2.2 Risks and Volatile Areas

- **Frontend–Backend Compatibility**: As the frontend (React.js) and backend (Node.js/Express) communicate over APIs, any change in API structure or data contracts could break parts of the user interface. This risk is mitigated by maintaining clear API documentation and versioning if necessary.
- **Simulated Payment Gateway**: The use of a simulated payment gateway abstracts away many of the challenges of secure payment processing. However, in a production system, integrating real-world payment services (e.g., Stripe, PayPal) introduces significant complexity and security considerations.
- **Database Schema Changes**: As new features such as promotions, reviews, or advanced filtering are added, the database schema may require updates. These changes may cause data migration issues or break existing queries. To mitigate this, migrations should be version-controlled.
- **User Requirements Change**: Future requests for mobile responsiveness, dynamic pricing, or internationalization could lead to large-scale design changes. The system is therefore designed using modular components and scalable backend logic to allow for easier adaptability.

- **Technology Stack Risks**: While React and Node.js are widely adopted and stable, dependencies on third-party libraries or outdated packages could create security vulnerabilities or compatibility issues over time. Regular updates and use of well-maintained libraries help reduce this risk.
- **Session Management and Authentication**: Improper session handling could expose user data. As such, authentication tokens or cookies must be securely managed and protected, especially for admin sessions.

To prepare for these risks, the system emphasizes clear module separation, documentation, and version control. Changes in core functionalities are expected to be localized to specific components or modules to allow targeted updates rather than broad redesigns.

3 System Architecture

Frontend Subsystem:

- Built using React for creating user-friendly interfaces.
- Handles user interactions, input validation, and rendering of dynamic content for both users and admins.
- Also uses MD Bootstrap and .css.

Backend Subsystem:

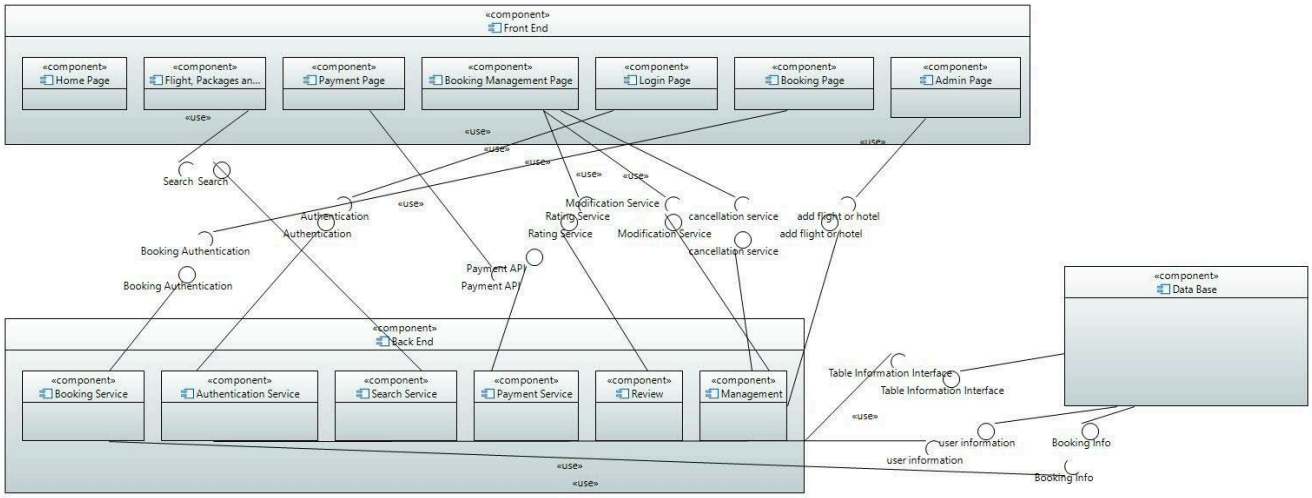
- Developed with Node.js and Express to serve as the API layer.
- Manages business logic, data validation, and communication between the frontend and database.

Database Subsystem:

- Uses Xampp for storing and retrieving application data such as user accounts, bookings, hotels, flights, and packages.

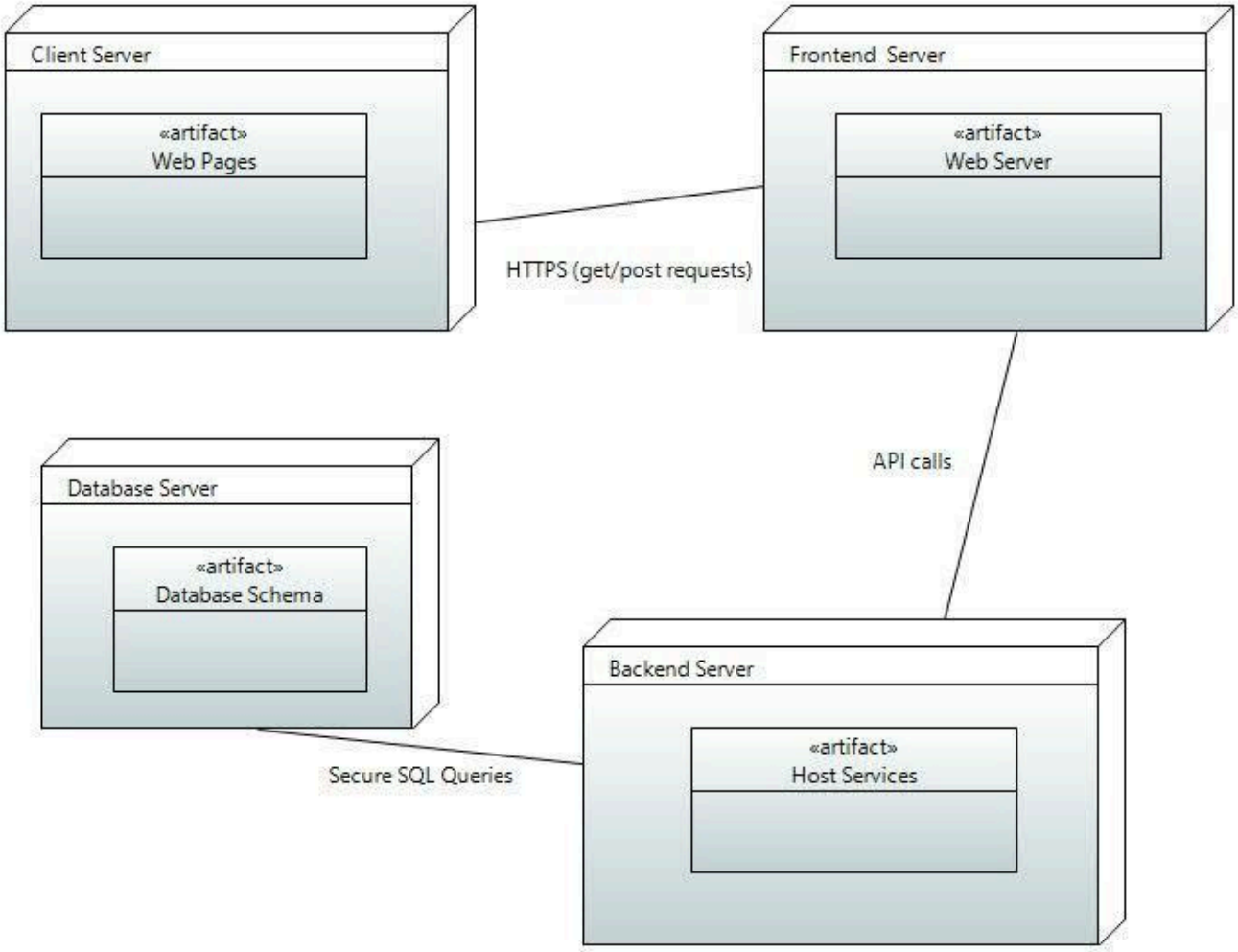
3.1 System Level Architecture

Component Diagram:



3.2 Software Architecture

Deployment Diagram:


Client Server
«artifact»
Web Pages
Frontend Server
«artifact»
Web Server
HTTPS (get/post requests)
API calls
Database Server
«artifact»
Database Schema
Backend Server
«artifact»
Host Services
Secure SQL Queries

3.2.1 Architectural Style

- **Frontend (Client-Side):** Built using **React.js**, the frontend is responsible for all user interactions. It communicates with the backend using RESTful API calls. The UI is designed using responsive components and styled with **MD Bootstrap**, providing a smooth and interactive experience.
- **Backend (Server-Side):** Implemented in **Node.js with Express.js**, the backend handles business logic, processes API requests from the frontend, and communicates with the database. It also manages user authentication, booking operations, admin controls, and data validation.
- **Database Layer:** The **MySQL database** stores persistent data such as user information, hotel/flight listings, and booking records. The database schema is normalized and accessed via backend APIs using SQL queries.

3.2.2 Component Responsibilities

- **User Interface Component:** Allows users to search, book, modify, and cancel accommodations or flights. Admin users can manage listings through a separate admin panel.
- **Authentication Component:** Manages login, registration, and session handling for both users and admins.
- **Booking Engine Component:** Handles hotel and flight bookings, including availability checks, price calculations, and payment simulation.
- **Feedback and Rating Component:** Allows users to leave reviews and ratings for services they've used.
- **Admin Management Component:** Enables admins to add, modify, or delete accommodation and travel options in the system.

3.2.3 Communication Flow

- The **frontend** sends API requests to the **Node.js backend**, which validates input, performs operations (such as booking or retrieving listings), and interacts with the **MySQL database**.

🎬 Once a transaction is successful (e.g., booking), the backend sends a response back to the frontend for user confirmation.

🎬 Email confirmations (simulated or real depending on implementation) may be handled asynchronously.

4 Design Strategy

4.1 Future System Extension or Enhancement \

The system is designed using a modular component-based approach (React on frontend, REST APIs on backend) that supports future enhancements. For instance:

- New modules like vacation packages or loyalty programs can be added without impacting core booking functionality.
- Features like real-time availability or third-party API integration (e.g., Google Maps or actual flight APIs) can be incorporated with minimal architectural changes.

4.2 System Reuse

Reusable components are employed both on the frontend and backend:

- React components (like HotelCard, BookingForm) are designed to be generic and reusable across multiple pages.
- Backend API routes are modular, and helper functions (e.g., validation, authentication middleware) promote reuse of logic.

4.3 User Interface Paradigms

The user interface is built with **React.js** and styled using **MD Bootstrap**, providing a responsive and modern experience. Key principles used include:

- Single Page Application (SPA) behavior with routing handled via React Router.
- Component-based design for consistency and scalability.
- Separate views and dashboards for users and admins.

4.4 Data Management

- The system uses a **MySQL relational database** to ensure data consistency and integrity.
- All interactions with the database are routed through the backend API, ensuring a secure and controlled access point.
- Data persistence is ensured through proper normalization and indexing.
- Though current deployment is on a local or small-scale server, future versions can use cloud-hosted MySQL or NoSQL solutions depending on scalability needs.

4.5 Concurrency and Synchronization

Although the current system is single-threaded in nature (due to limited scale and usage), basic concurrency control is handled as follows:

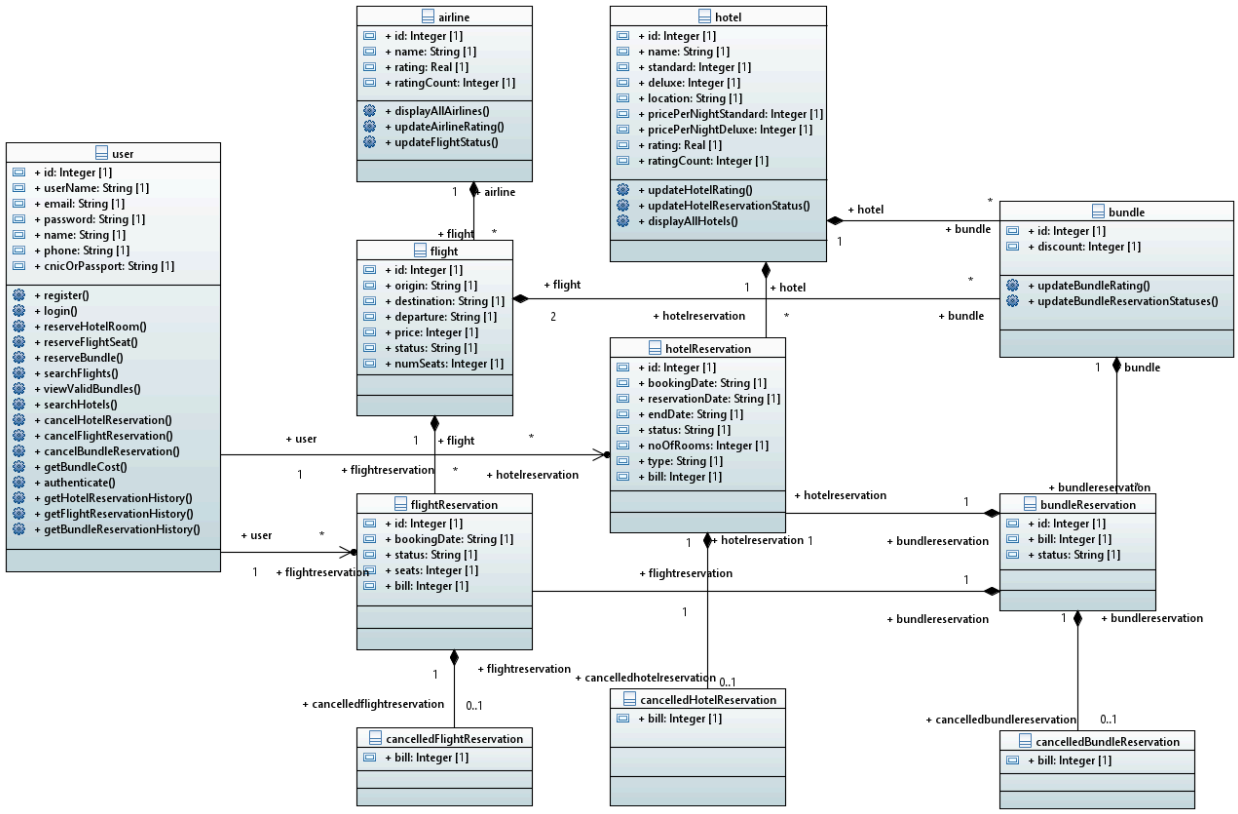
- The backend ensures no overbooking by checking availability before confirming a booking.

- In multi-user scenarios, database-level transactions or locks would be needed to prevent race conditions.
- Future scalability plans can include message queues or load balancers for managing concurrent user traffic.

5 Detailed System Design

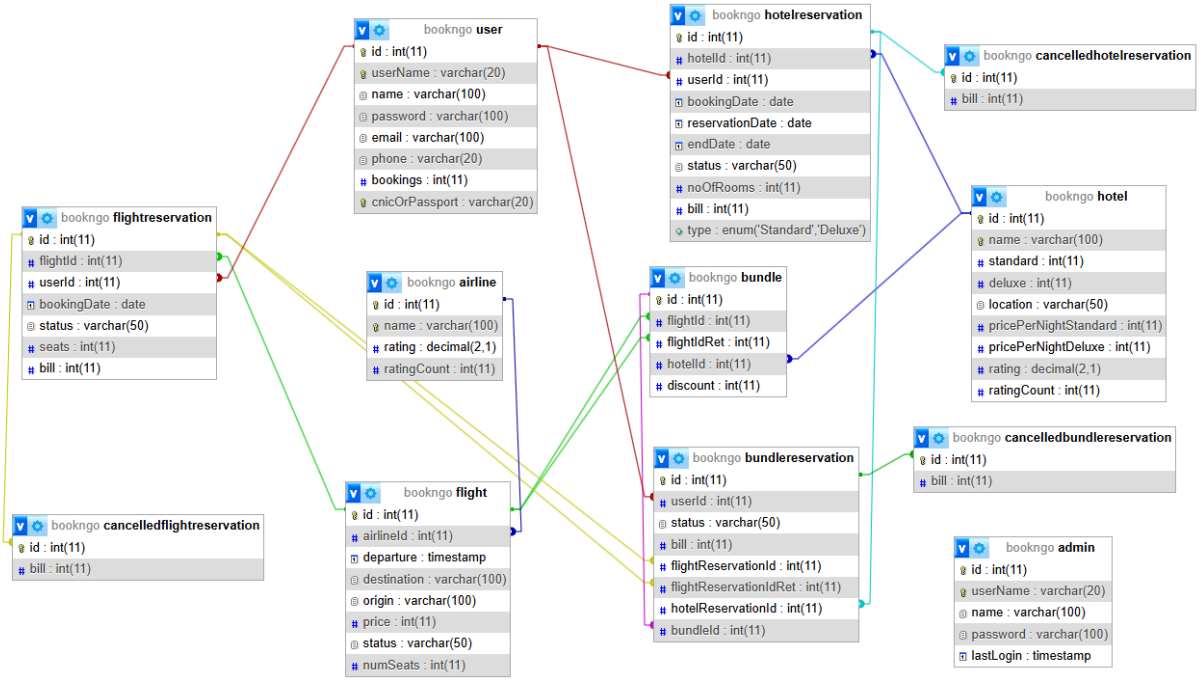
5.1 Database Design

Class Diagram:



5.1.1 ER Diagram

ER Diagram:



5.1.2 Data Dictionary

5.1.2.1 Data 1

< Data 1>						
Name	User					
Alias	Customer					
Where-used/how-used	🎬 Used as an external entity for authentication and profile management. 🎬 Referenced in flightreservation, hotelreservation, and bundlereservation as a foreign key (userId).					
Content description	user = userId + userName + name + password + email + phone + cnicOrPassport + bookings					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**
id	Unique identifier for the user	int	11	No	-	PK

userName	Login username	varchar	20	No	-	
name	Full name	varchar	100	No	-	
password	User password	varchar	100	No	-	
email	Email address	varchar	100	No	-	
phone	Contact number	varchar	20	No	-	
cnicOrPassport	ID for verification	varchar	20	No	-	
bookings	Total bookings made	int	11	Yes	0	

5.1.2.2 Data 2

< Data 2>	
Name	Flightreservation
Alias	Flight booking
Where-used/how-used	Input to: cancelledflightreservation Output from: flight
Content description	flightreservation = id + flightId + userId + bookingDate + status + seats + bill

Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**
id	Unique identifier	int	11	No	None	PK
flightId	Reference to flight	int	11	No	None	FK
userId	Reference to user	int	11	No	None	FK
bookingDate	Date of booking	date	—	No	None	
status	Booking status	varchar	50	No	'Pending'	

seats	Number of seats booked	int	11	No	None	
bill	Total bill amount	int	11	No	None	

5.1.2.3 Data 3

< Data 3>	
Name	Flight
Alias	N/A
Where-used/how-used	Referenced by: flightreservation, bundle Output to: Admin UI
Content description	flight = id + airlineId + departure + destination + origin + price + status + numSeats

Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**
id	Unique flight ID	int	11	No	None	PK
airlineId	Airline reference	int	11	No	None	FK
departure	Departure date/time	timestamp	—	No	None	
destination	Destination city	varchar	100	No	None	
origin	Origin city	varchar	100	No	None	
price	Ticket price	int	11	No	None	
status	Flight status	varchar	50	No	'Available'	
numSeats	Total number of seats	int	11	No	None	

5.1.2.4

< Data 4>	
Name	Hotel
Alias	

Where-used/how-used	• Used in hotelreservation and bundle as foreign key reference • Provides details for available hotels
Content description	Hotel = name + standard + deluxe + location + pricePerNightStandard + pricePerNightDeluxe + rating + ratingCount

Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**
id	Unique hotel ID	int	11	No		PK
name	Hotel name	varchar	100	No		
standard	Availability of standard rooms	int	11	Yes		
deluxe	Availability of deluxe rooms	int	11	Yes		
location	Hotel location	varchar	50	Yes		
pricePerNightStandard	Price per standard room per night	int	11	Yes		
pricePerNightDeluxe	Price per deluxe room per night	int	11	Yes		
rating	Average customer rating	decimal	2,1	Yes		
ratingCount	Number of ratings received	int	11	Yes		

5.1.2.5

< Data 5>	
Name	Airline
Alias	
Where-used/how-used	Referenced in flight for flight-airline relationship

Content description	Airline = name + rating + ratingCount					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**
id	Unique airline ID	int	11	No		PK
name	Airline name	varchar	100	No		
rating	Customer rating average	decimal	2,1	Yes		
ratingCount	Number of ratings	int	11	Yes		

5.1.2.6

< Data 6>						
Name	hotelreservation					
Alias						
Where-used/how-used	● User hotel booking records ● Used in bundle and cancellation tables					
Content description	HotelReservation = hotelId + userId + bookingDate + reservationDate + endDate + status + noOfRooms + bill + type					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**
id	Unique hotel reservation ID	int	11	No		PK
hotelId	Related hotel	int	11	No		FK
userId	Booking user	int	11	No		FK
bookingDate	Date the booking was made	date	-	Yes		
reservationDate	Check-in date	date	-	Yes		
endDate	Check-out date	date	-	Yes		

status	Current status (e.g., confirmed)	varchar	50	Yes		
noOfRooms	Number of rooms booked	int	11	Yes		
bill	Total cost of reservation	int	11	Yes		
type	Room type	enum	-	Yes	Standard	

5.1.2.7

< Data 7>						
Name	cancelledhotelreservation					
Alias						
Where-used/how-used	Tracks cancelled hotel bookings					
Content description	CancelledHotelReservation = id + bill					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**
id	Associated hotel reservation ID	int	11	No		PK, FK
bill	Cancelled booking bill	int	11	Yes		

5.1.2.8

< Data 8>						
Name	bundle					
Alias						
Where-used/how-used	● Combined package of hotel and flights ● Referenced by bundlereservation					
Content description	Bundle = flightId + flightIdRet + hotelId + discount					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**

id	Unique bundle ID	int	11	No		PK
flightId	Outbound flight ID	int	11	No		FK
flightIdRet	Return flight ID	int	11	No		FK
hotelId	Hotel included in the bundle	int	11	No		FK
discount	Discount offered on bundle	int	11	Yes		

5.1.2.9

< Data 9>						
Name	bundlereservation					
Alias						
Where-used/how-used	Stores reservations involving bundles					
Content description	BundleReservation = userId + flightReservationId + flightReservationIdRet + hotelReservationId + bundleId + bill + status					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Defaul t Value**	**Key Type**
id	Unique bundle reservation ID	int	11	No		PK
userId	User who booked the bundle	int	11	No		FK
flightReservationId	Related outbound flight booking	int	11	Yes		FK
flightReservationIdRet	Related return flight booking	int	11	Yes		FK
hotelReservationId	Related hotel reservation	int	11	Yes		FK

bundleId	ID of the booked bundle	int	11	No		FK
status	Current booking status	varchar	50	Yes		
bill	Total cost of the bundle reservation	int	11	Yes		

5.1.2.10

< Data 10>						
Name	cancelledbundlereservation					
Alias						
Where-used/how-used	Track cancelled bundles					
Content description	CancelledBundleReservation = id + bill					
Column Name	***Description***	***Type***	***Length***	***Nullable***	***Default Value***	***Key Type***
id	*Associated bundle reservation ID*	*int*	*11*	*No*		*PK, FK*
bill	*Cost of cancelled bundle booking*	*int*	*11*	*Yes*		

5.1.2.11

< Data 11>						
Name	cancelledflightreservation					
Alias						
Where-used/how-used	Tracks cancelled flight reservations					
Content description	CancelledFlightReservation = id + bill					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**

id	Associated bundle reservation ID	int	11	No		PK, FK
bill	Cost of cancelled bundle booking	int	11	Yes		

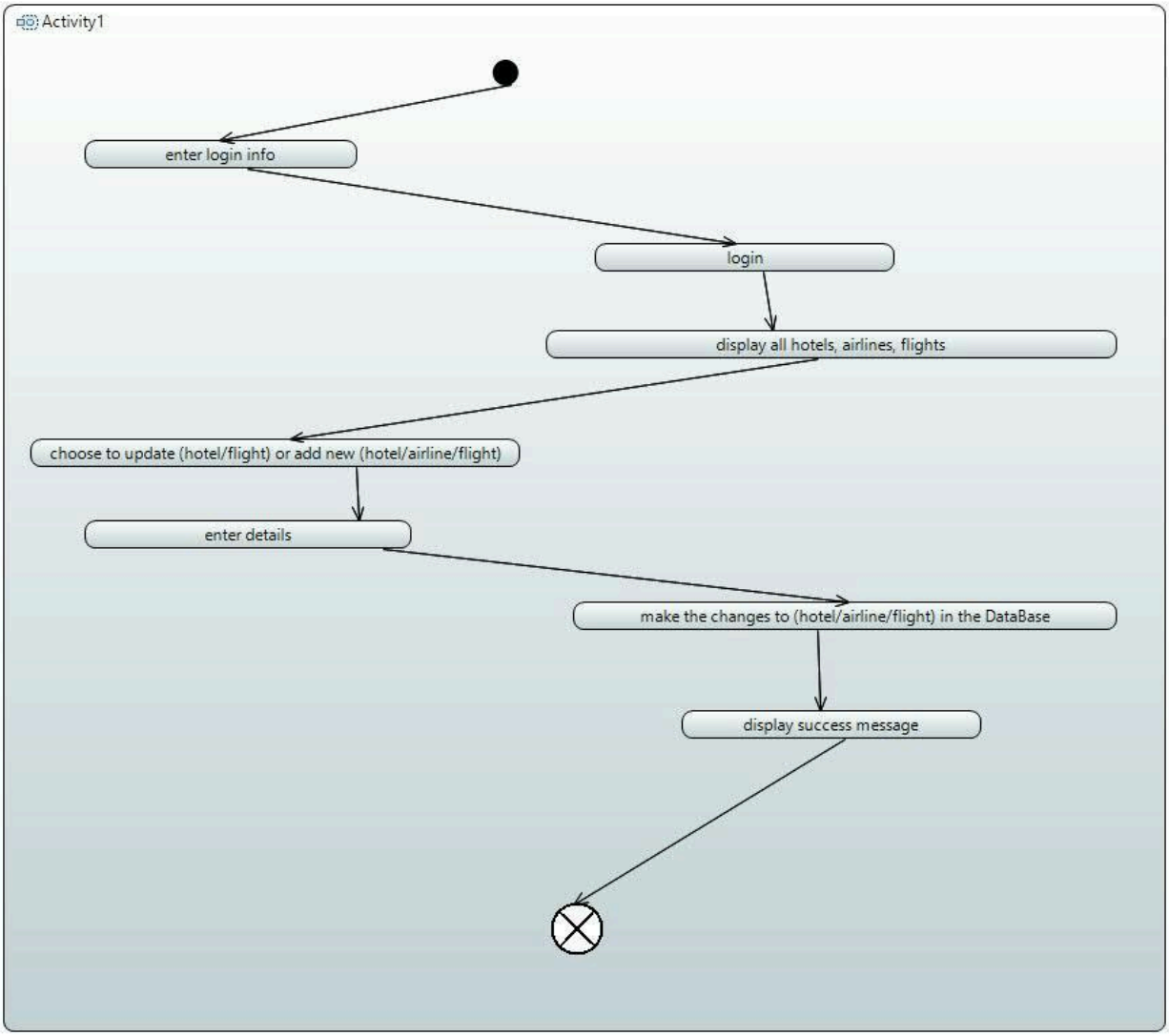
5.1.2.12

< Data 12>						
Name	Admin					
Alias						
Where-used/how-used	Used to manage flights and hotels					
Content description	Admin = userName + name + password + lastLogin					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**
id	Admin ID	int	11	No		PK
userName	Username	varchar	20	No		
name	Full name	varchar	100	No		
password	Login password	varchar	100	No		
lastLogin	Timestamp of login	timestamp	-	Yes		

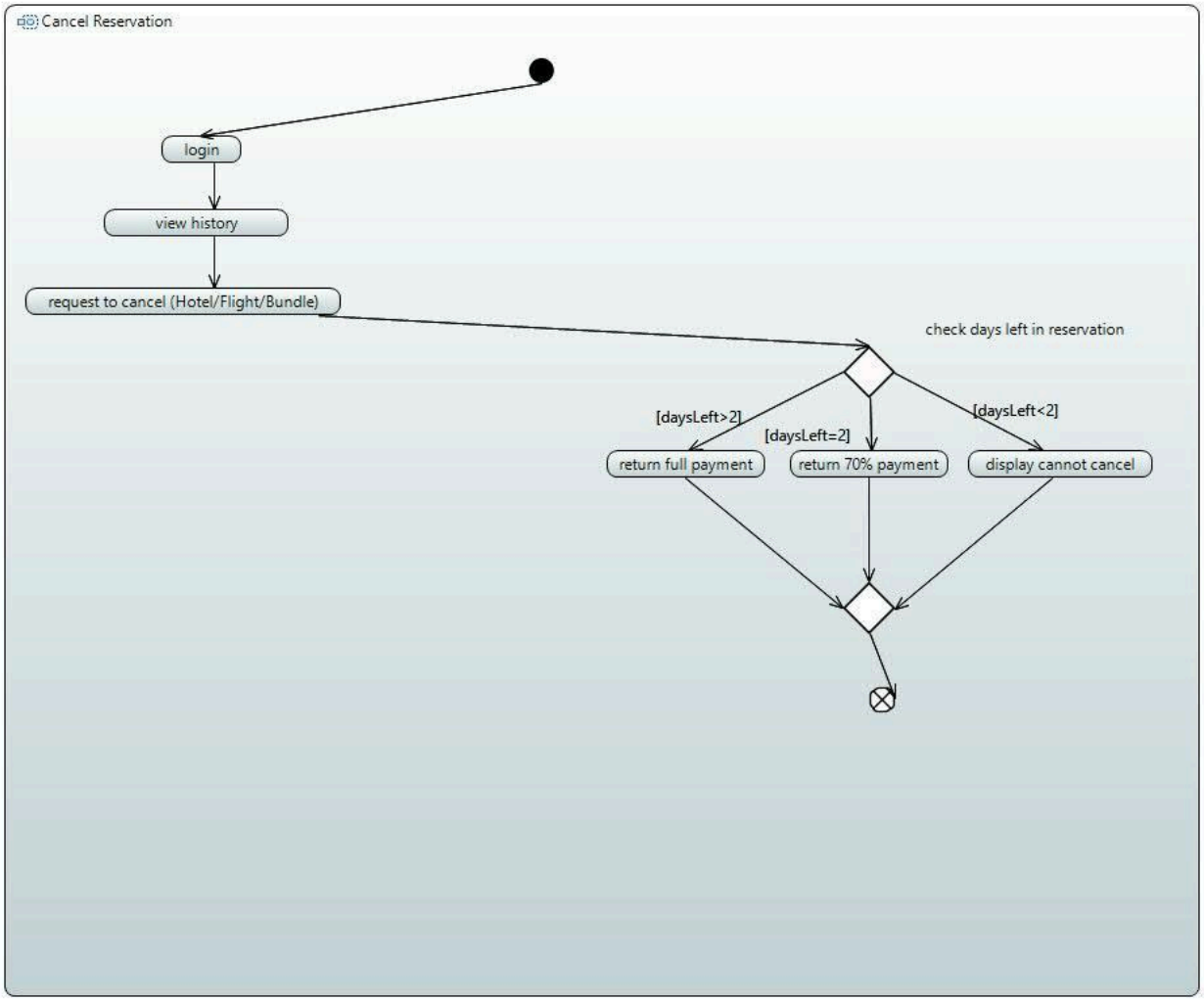
5.2 Application Design

Activity Diagrams:

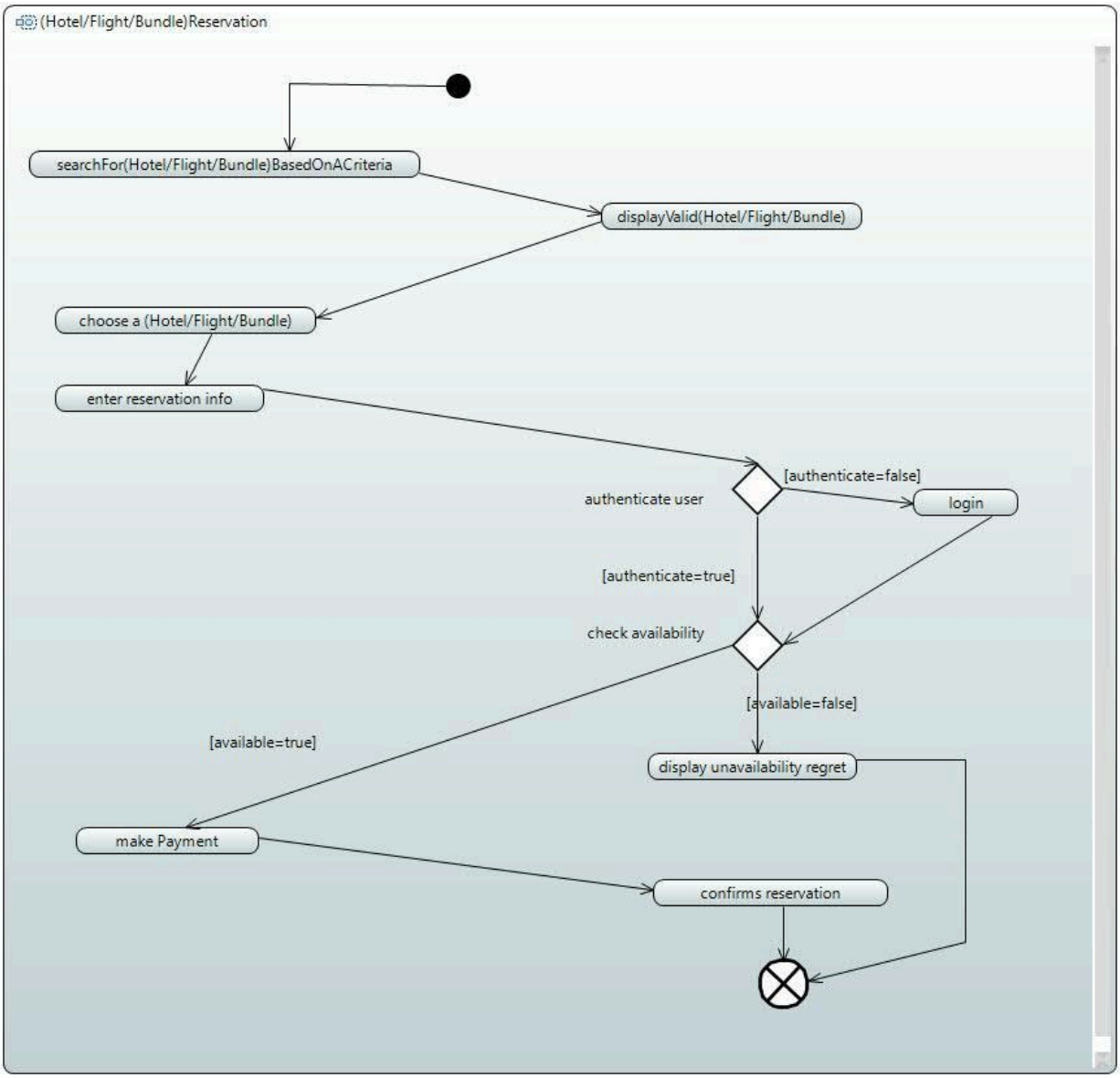
Admin Activity Diagram:



Cancellation Activity Diagram:

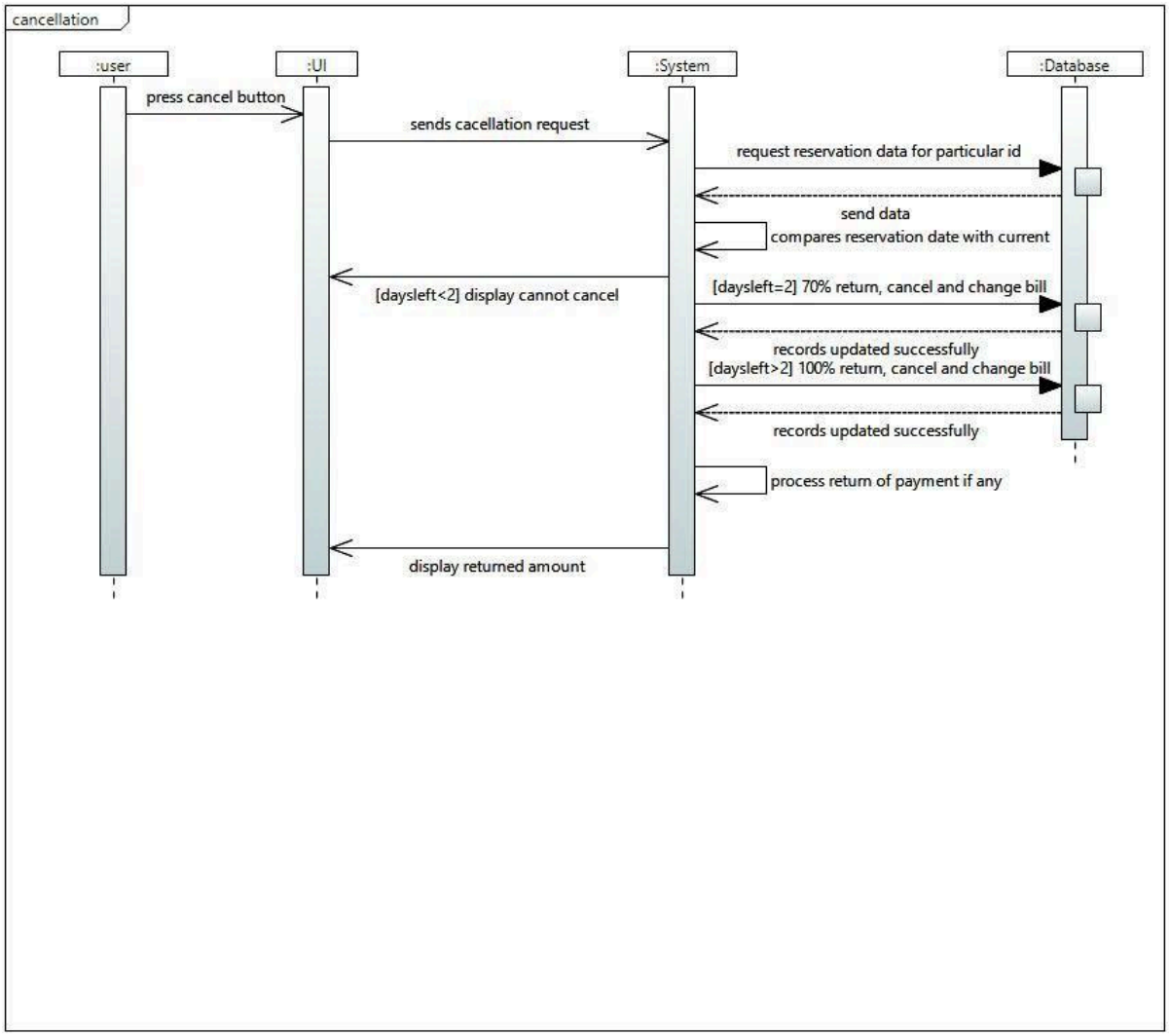


Reservation Activity Diagram:

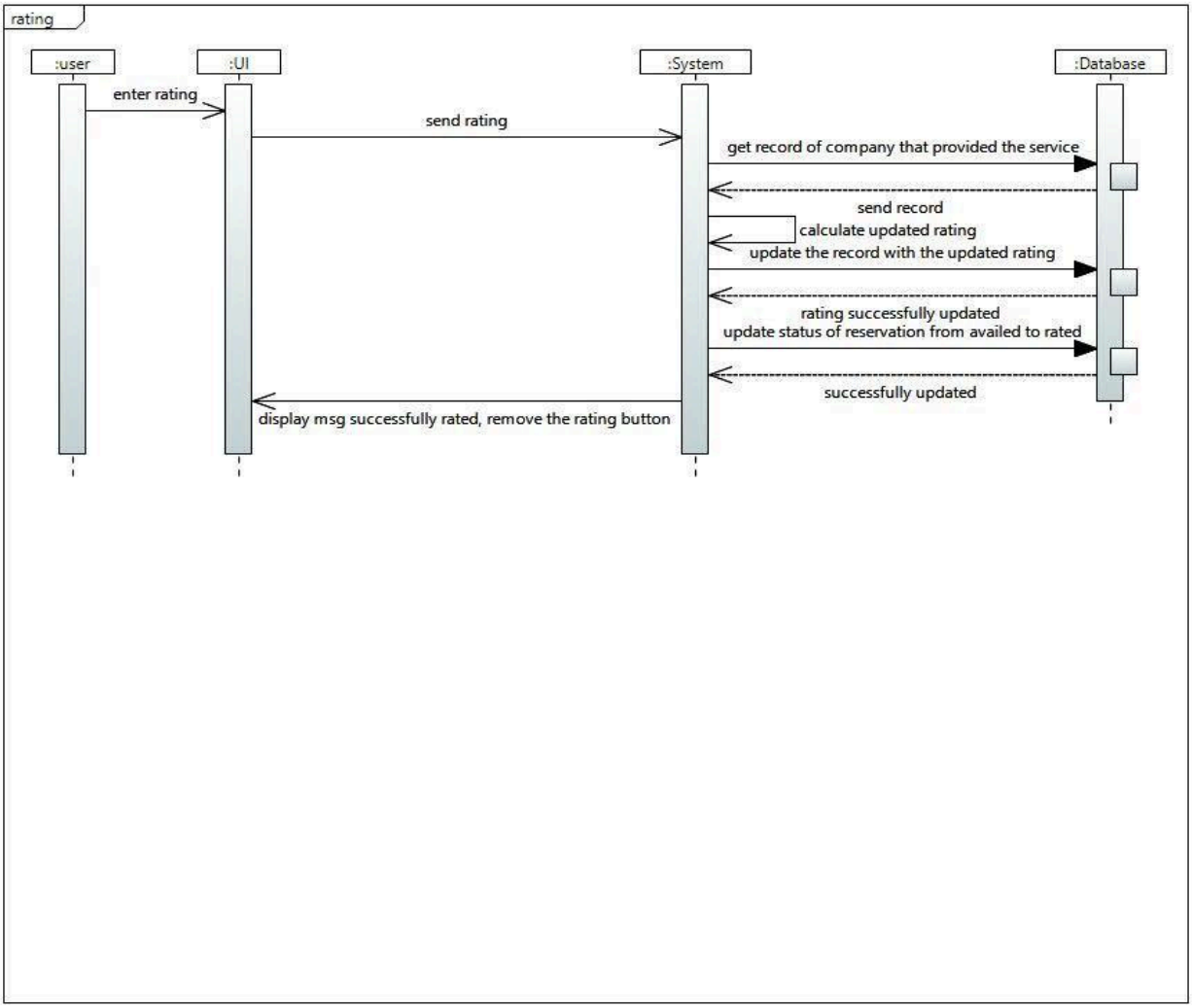


5.2.1 Sequence Diagram

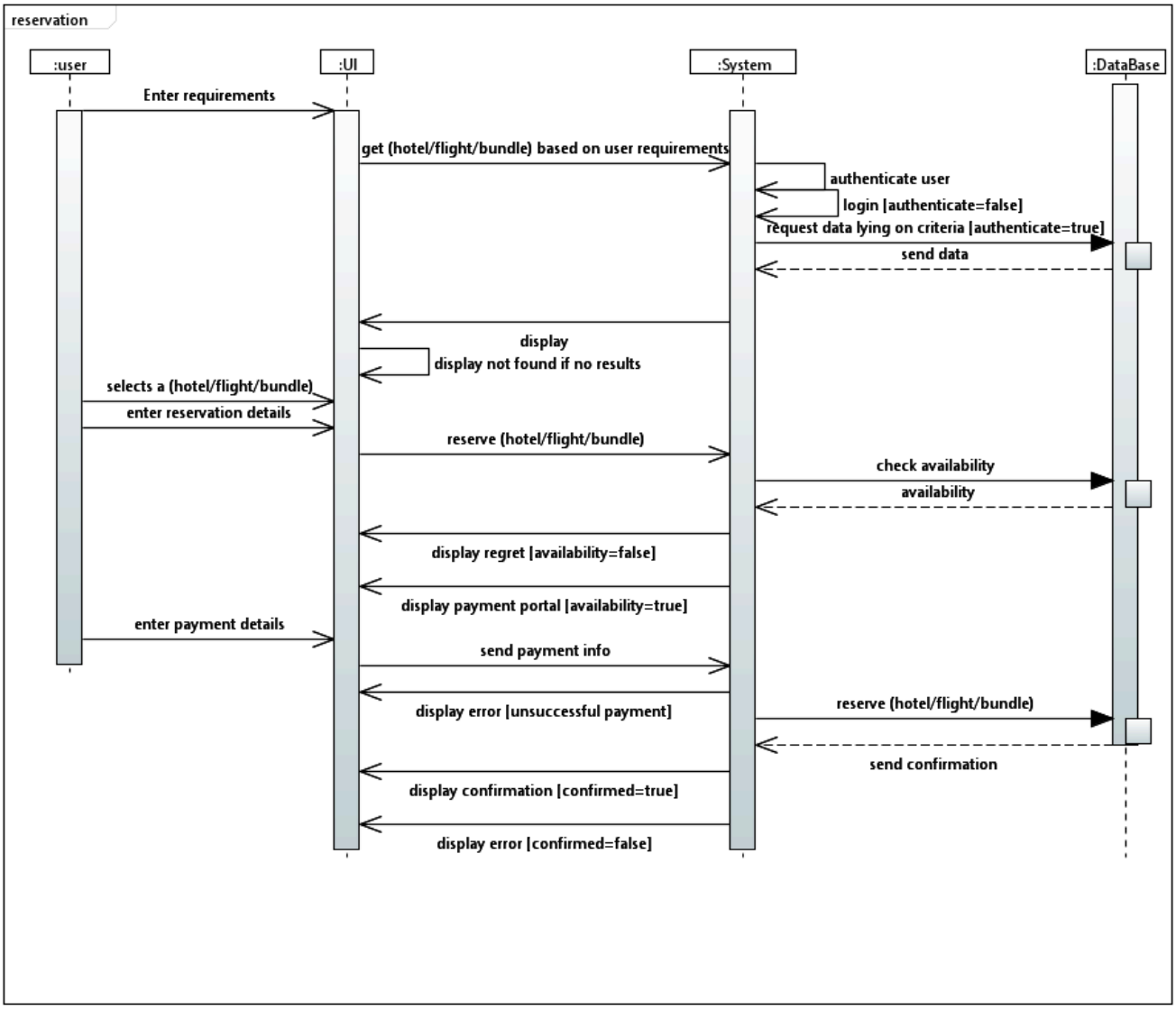
5.2.1.1 <Sequence Diagram 1>



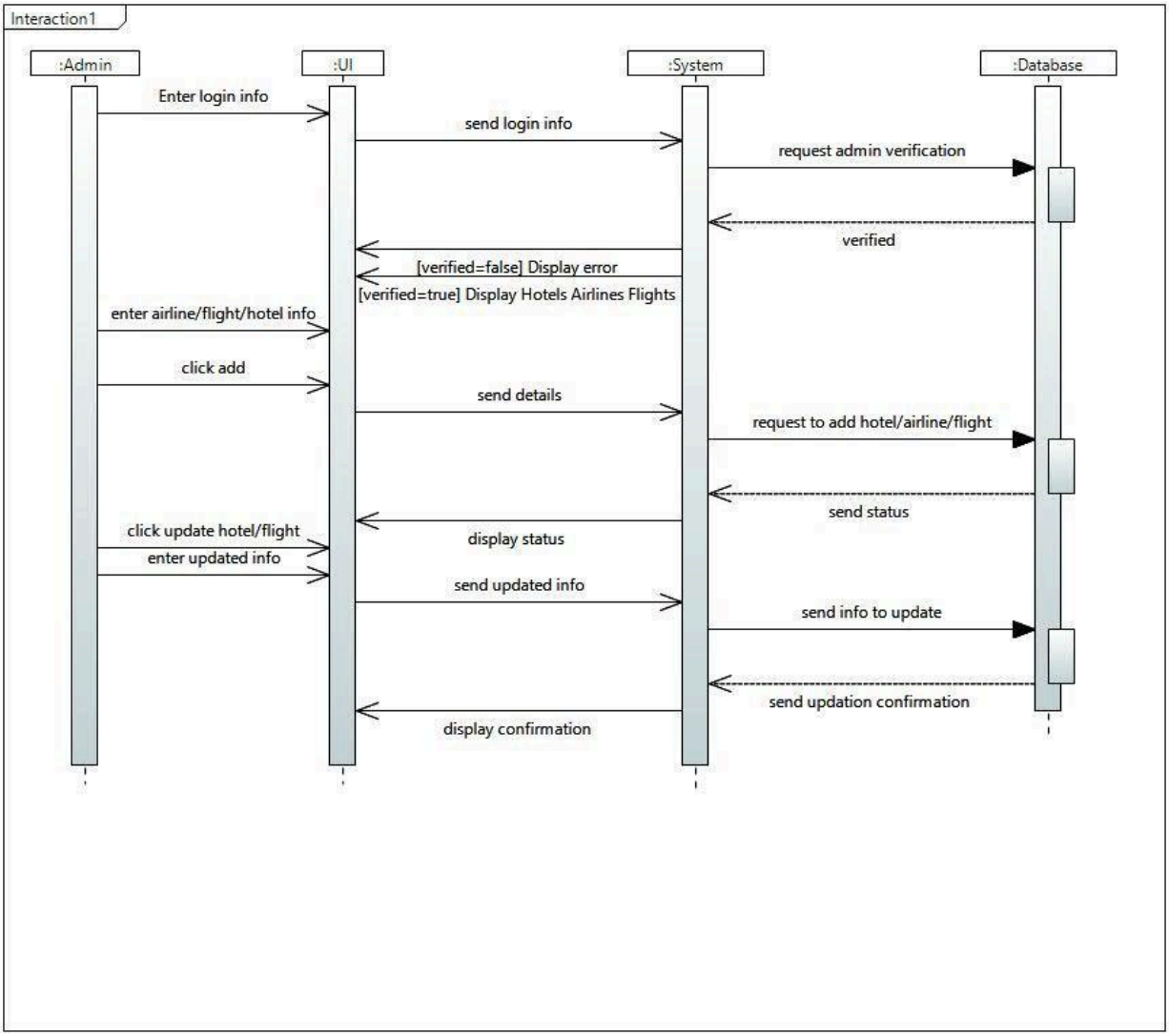
5.2.1.2 <Sequence Diagram 2>



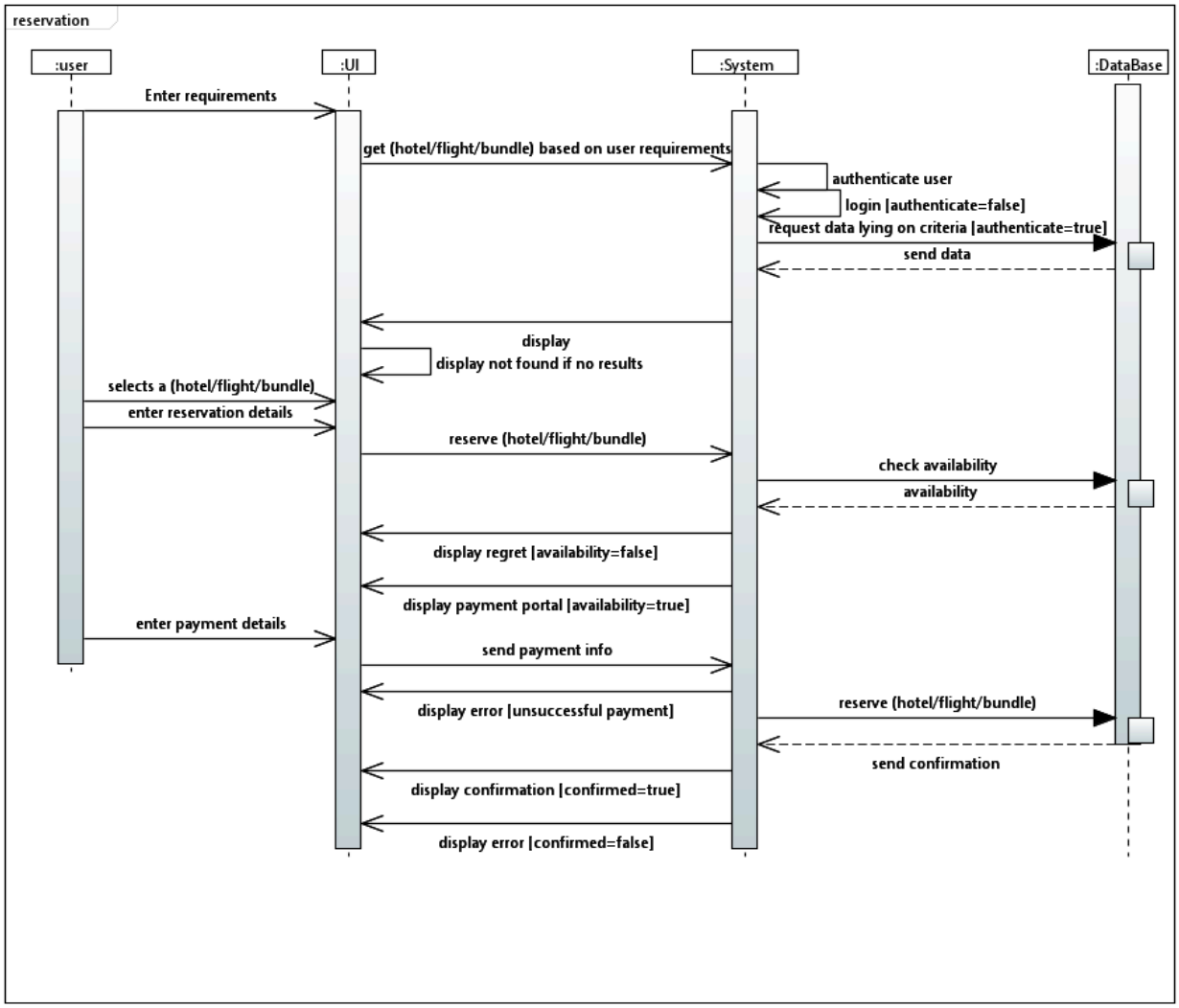
5.2.1.3 <Sequence Diagram 3>



5.2.1.4 <Sequence Diagram 4>

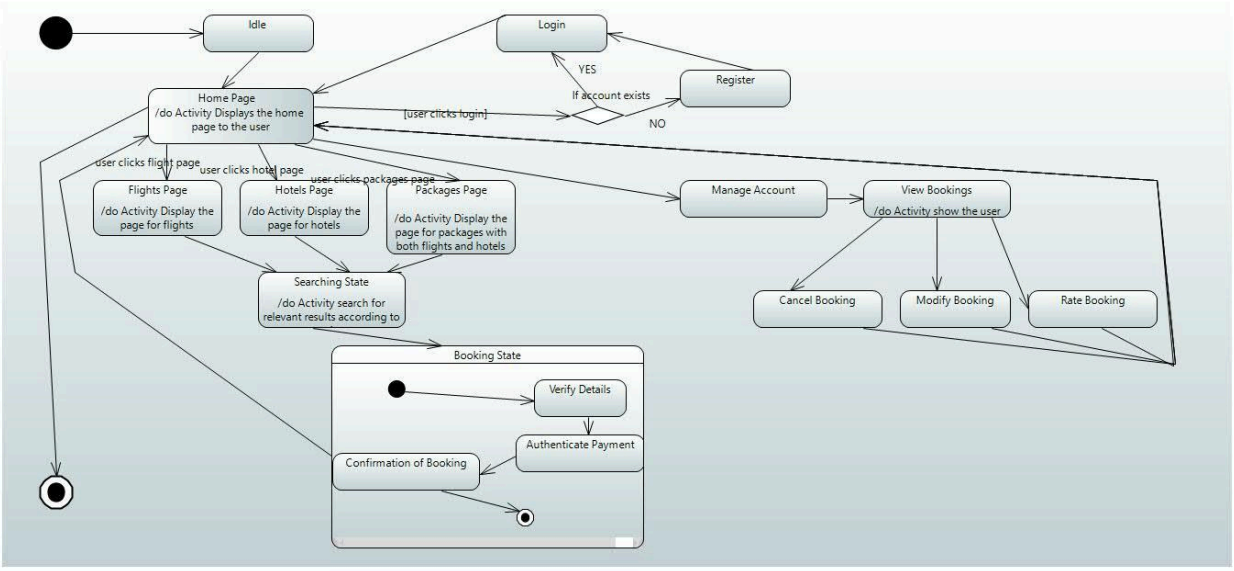


5.2.1.5 <Sequence Diagram 5>



5.2.2 State Diagram

5.2.2.1 <State Diagram 1>



6 References

- **React.js** – Frontend JavaScript library for building user interfaces
- **MDBootstrap (Material Design for Bootstrap)** – UI framework for styling React components
- **Node.js** – Backend JavaScript runtime for server-side logic
- **Express.js** – Web framework for Node.js
- **MySQL / XAMPP** – Relational database and local server stack
- **Postman** – Tool for testing and developing APIs
- **Papyrus UML** – For designing Use Case and Sequence diagrams
- **VS Code** – Source code editor used for development
- **Draw.io** – (If used) For creating diagrams like ERDs
- **Google Fonts / FontAwesome** – For UI styling (fonts and icons)
- **OpenAI / ChatGPT** – For support with documentation and guidance

7 Appendices

N/A