Software Requirement 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 supervisors or co-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

[Following table will contain list of people whom the document will be distributed after every sign-off]

Name	Role
Zulfiqar Ali	Supervisor
N/A	Co- Supervisor

Document Sign-Off

[Following table will contain sign-off details of document. Once the document is prepared and revised, this should be signed-off by the sign-off authority.

Any subsequent changes in the document after the first sign-off should again get a formal sign-off by the authorities.]

Version	Sign-off Authority	Sign-off Date

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 Abbreviations

[Describe the abbreviations use this document.]

Term	**Description**
ASP	Active Server Pages
DD	Design Specification
BookNGO	The name of the travel booking platform developed as part of this project.
API	Application Programming Interface, a set of tools and protocols for building and interacting with software applications.
Admin	A user with elevated privileges who manages hotels, flights, and airlines on the BookNGO platform.
User	An individual who interacts with the platform to book flights, hotels, and packages.

1.4 Document Convention

Font: Arial
Font Size: 12

2. Overall System Description

2.1. Project Background

The travel and tourism industry has seen a significant shift towards digitalization, with travelers increasingly relying on online platforms for their booking needs. Existing platforms often fail to provide a unified and seamless experience, leaving users juggling between separate apps for flights, hotels, and packages.

BookNGO is designed to address these challenges by creating an all-in-one travel booking platform. It offers users the ability to reserve hotels, flights, and complete packages that include outward flights, hotel stays, and return flights. The platform also enables users to view their booking history, modify or cancel reservations, and rate the services they have used.

2.2. Project 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.

2.3. Not In Scope

The following functionalities are beyond the scope of the current project:

- Integration with external travel APIs for live flight tracking or real-time hotel availability.

- Advanced payment gateway functionalities, such as installment-based payments or loyalty reward points.
- Multilingual support for global users.
- Multiple currencies support.
- Recommendations based on travel history.

2.4. Project Objectives

The objectives of **BookNGO** include:

- Providing a unified platform for booking flights, hotels, and packages to enhance user convenience.
- Simplifying travel management for users through easy access to booking history and the ability to modify or cancel reservations.
- Enabling administrators to efficiently manage data for hotels, flights, and airlines.
- Offering a user-friendly and visually appealing interface.
- Demonstrating practical implementation of software development concepts for academic purposes.

2.5. Stakeholders

Primary Stakeholders:

- **End-Users (Travelers)**:
  Individuals using the platform to book flights, hotels, and packages. They also interact with features like viewing booking history, modifying or cancelling reservations, and rating services.
- **Administrators (Admins)**:
  System administrators responsible for managing flights, hotels, and airlines by adding new entries or editing existing ones.

Secondary Stakeholders:

- **Course Instructors and Evaluators**:
  Professors and academic mentors evaluating the project for its technical correctness, functionality, and adherence to course requirements.

2.6. Operating Environment

- **Hardware Platform**: Desktop, laptop, and mobile devices.
- **Operating System**: Windows, macOS, Linux, iOS, Android.
- **Network Environment**: Requires an internet connection for accessing the platform.

- **Software Components**:
  - Backend API hosted on localhost during development.
  - Frontend web application built using React using localhost.
  - Database hosted on XAMPP.

2.7. System Constraints

Software Constraints:

- Limited API integrations in the current phase.

Hardware Constraints:

- The system should be operable on standard devices with minimal specifications.

Cultural Constraints:

- English will be the only supported language.

User Constraints:

- The interface will focus on simplicity to cater to a broad audience, avoiding complex navigation.

2.8. Assumptions & Dependencies

Assumptions:

- Users will have access to an internet connection.
- The browser used by the user will support modern JavaScript and CSS.
- Admins are trained to use the admin portal.

Dependencies:

- Successful integration of frontend and backend services.
- Reliable hosting environment for the database and APIs.
- Timely feedback from stakeholders for iterative development.

3. External Interface Requirements

3.1. Hardware Interfaces

The system will interact with hardware devices in the following ways:

- **User Devices**: The system will support desktops, laptops, tablets, and mobile devices running web browsers.
  - Supported browsers include Chrome, Firefox, Edge, and Safari.
- **Server Infrastructure**: Hosted on local host (e.g., XAMPP setup).
  - Requires sufficient RAM and CPU for managing concurrent user sessions

        and database operations.
    - Compatible with Oracle MySQL as the database management system.
- **Network Hardware**: Ensures reliable and high-speed connectivity for seamless operation.

3.2. Software Interfaces

The system will connect with the following software components:

Operating Systems: Compatible with Windows, macOS, and Linux for client-side users.
Database: Oracle MySQL for managing booking, user, and admin data.
Data exchange includes user profiles, booking history, and availability records.
Backend Framework: Node.js serves as the backend for API development and communication with the database.
Frontend Framework: React.js, along with HTML, CSS, and JavaScript, is used to create the user interface.
Development Tools: React for project setup and build process.

3.3. Communications Interfaces

The system uses the following communication standards and protocols:

- **HTTP/HTTPS**: Ensures secure communication between the client and server.
- **REST APIs**: Used for communication between the frontend and backend components.
- **Encryption**: All sensitive data, such as user credentials and payment details, will be encrypted using protocols.
- **Data Transfer Rates**: Optimized for real-time interactions, ensuring minimal latency for user actions such as searches or bookings.

4. Functional Requirements

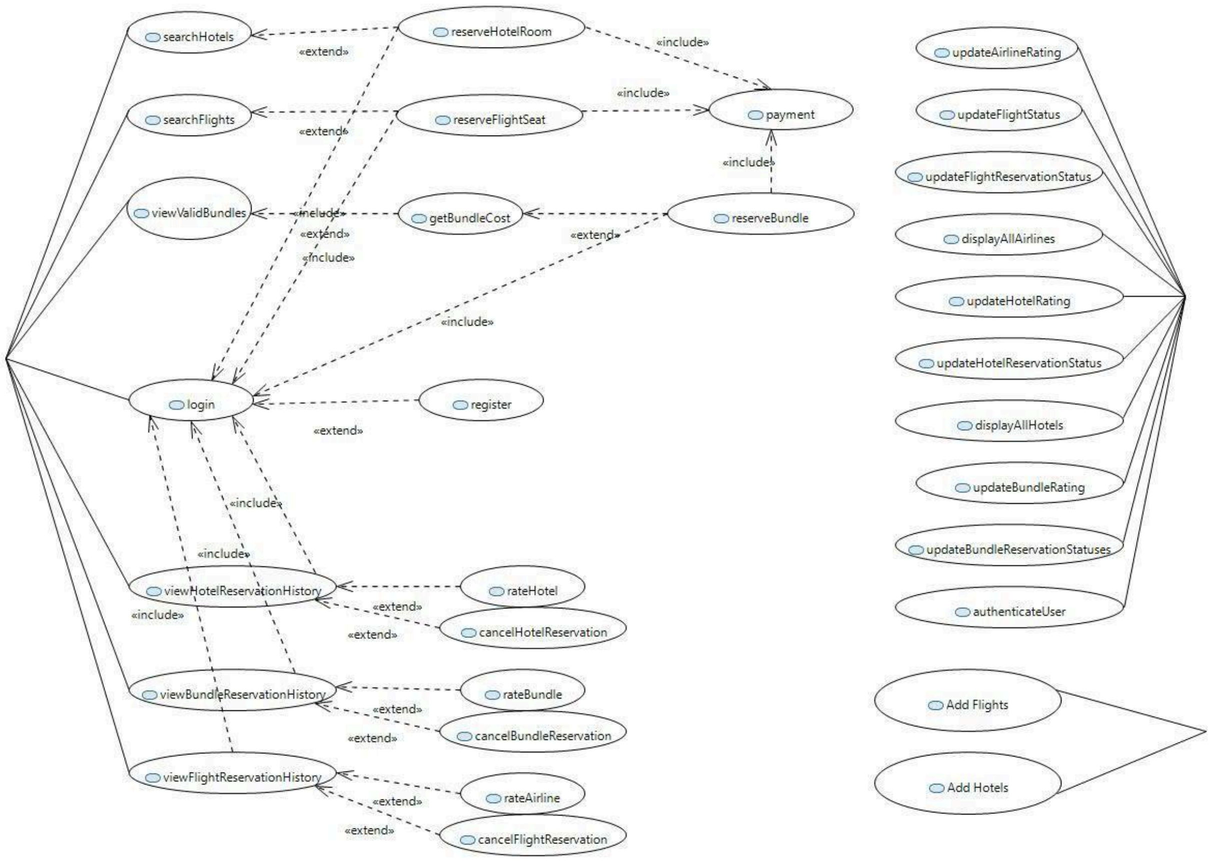
4.1. Functional Hierarchy

The system's functionality is structured into a hierarchy of modules and sub-functions:

1. **User Module**
   1.1. Registration
   1.2. Login/Logout
   1.3. Profile Management
   1.4. View Booking History
2. **Search Module**
   2.1. Search Accommodations
   2.2. Search Flights
   2.3. Search Vacation Packages
3. **Booking Module**
   3.1. Accommodation Booking
   3.2. Flight Booking
   3.3. Package Booking
   3.4. Modify Bookings
   3.5. Cancel Bookings
   3.6. Payment Processing
4. **Admin Module**
   4.1. Accommodation Management
   4.2. Travel Route Management
5. **Rating and Feedback Module**
   5.1. Leave Ratings for Services
   5.2. View Ratings

4.2. Use Cases

Universal Use Case Diagram:



4.2.1. Booking a flight

<Use case Id: Bookingaflight>		
Use case Id:	1	
Actors: Registered User, Payment Gateway		
Feature: Flight Reservation Management		
Pre-condition:	• The user must be logged into the system. • Flights for the specified route and date must be available.	
Scenarios		
Step#	**Action**	**Software Reaction**

1.	The actor searches for flights by entering departure and arrival locations, dates, and other filters.	The system receives search criteria and queries the database for matching flights.
2.	.	The system displays a list of available flights based on the search criteria.
3.	The actor selects a flight and clicks on "Book Now."	The system stores the selected flight temporarily and moves to the passenger information form.
4.	The actor enters passenger details (e.g., name, passport number).	The system validates the input and prepares for payment.
5.	The actor confirms the booking and proceeds to the payment page.	The system redirects the actor to the secure payment gateway.
6.	The payment gateway processes the payment securely.	The system awaits confirmation from the payment gateway.
7.		Upon successful payment, the system saves the booking to the database.
8.		The system generates and displays a confirmation message. A confirmation email is sent to the user.
Alternate Scenarios: Write additional, optional, branching or iterative steps. Refer to specific action number to ensure understandability.		
1a: If the user is not logged in, the system redirects them to the login/registration page and then back to the booking page after successful login **2a:** If the selected flight becomes unavailable during the process, the system informs the user.		
Post Conditions		
Step#	**Description**	
1	The flight booking is recorded in the system.	

2	The user receives a booking confirmation email.
Use Case Cross referenced	● Payment Processing ● Search Flights

4.2.2. User Books a hotel

<Use case Id: UserBooksahotel>	
Use case Id:	2
Actors:	Registered User, Payment Gateway
Feature:	Hotel Booking System
Pre-condition:	● The user must be logged into the system. ● Hotels must be available for the specified location and date.

Scenarios

Step#	**Action**	**Software Reaction**
1.	The actor searches for hotels by entering location, check-in/check-out dates, and filters such as price or rating.	System retrieves hotels based on criteria.
2.	. The system displays a list of hotels matching the criteria.	The system displays a list of available hotels based on the search criteria.
3.	The actor selects a hotel, chooses a room type, and clicks "Book Now."	The system stores the selected hotel temporarily and moves to the resident information form.
4.	The system prompts for guest details and any special requests.	The system validates the input and prepares for payment.
5.	The actor confirms the booking and proceeds to payment.	The system redirects the actor to the secure payment gateway.
6.		The system awaits confirmation from the payment gateway.
7.		The payment gateway processes the transaction.
8.		The system confirms the booking.

Alternate Scenarios: Write additional, optional, branching or iterative steps. Refer to specific action number to ensure understandability.	
1a: If the user is not logged in, redirect to login page. **2a:** If the selected room becomes unavailable, notify the user.	
Post Conditions	
Step#	**Description**
1	The hotel booking is recorded in the system.
2	The user receives a booking confirmation email.
Use Case Cross referenced	● Payment Processing

4.2.3. User Cancels a Booking

<Use case Id: UserCancelsaBooking>		
Use case Id:	3	
Actors:	Registered User, Payment Gateway	
Feature:	Booking Cancellation	
Pre-condition:	● The user must be logged into the system. ● Booking must be cancellable	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	Actor navigates to booking history.	System displays all bookings.

2.	Actor selects a booking and clicks "Cancel."	
3.		System asks for confirmation and shows refund amount.
4.	Actor confirms cancellation.	
5.	The actor confirms the booking and proceeds to payment.	
6.		System processes cancellation and updates database.
Alternate Scenarios: Write additional, optional, branching or iterative steps. Refer to specific action number to ensure understandability.		
1a: If booking is non-refundable, notify the user.		
Post Conditions		
Step#	**Description**	
1	Booking is marked canceled in the system.	
2	Availability is updated.	
Use Case Cross referenced	● View booking History ● Modify Booking	

4.2.4. Admin Adds Accommodation Details

<Use case Id: AdminAddsAccomodation>		
Use case Id:	4	
Actors:	Admin	
Feature:	Admin Panel - Accommodation Management	
Pre-condition:	Admin must be logged in.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	Admin logs in.	System verifies credentials.
2.	Admin navigates to accommodation module.	
3.	Admin clicks “Add accommodation”	System asks for confirmation and shows refund amount.
4.	Actor confirms cancellation.	System prompts for accommodation details.
5.	Admin submits details.	

6.		System validates and updates the database.
Alternate Scenarios: Write additional, optional, branching or iterative steps. Refer to specific action number to ensure understandability.		
1a: If required fields missing, system displays error.		
Post Conditions		
Step#	**Description**	
1	Accommodation is added and available for booking.	
Use Case Cross referenced	● Manage accommodation	

4.2.5. User Rates a Service

<Use case Id: UserRatesService>		
Use case Id:	5	
Actors: Registered User		
Feature: Service Rating		
Pre-condition:	● User must have completed a booking.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	Actor navigates to booking history.	System displays all completed bookings.
2.	Actor selects a booking and clicks "Rate."	

3.		System prompts for rating
4.	Actor submits rating.	
5.		System saves the rating
Alternate Scenarios: Write additional, optional, branching or iterative steps. Refer to specific action number to ensure understandability.		
1a: If user already rated, notify them **2a:** If submission fails, allow retry.		
Post Conditions		
Step#	**Description**	
1	Rating is recorded and shown to others.	
2	Feedback is stored for admin review.	

Use Case Cross referenced	● View booking History

4.2.6. User Modifies a booking

<Use case Id: UserModifiesbooking>	
Use case Id:	6
Actors:	Registered User
Feature:	Booking Management
Pre-condition:	● The user must be logged into the system. ● Booking must be modifiable ● Requested changes must be available.
Scenarios	

Step#	**Action**	**Software Reaction**
1.	Actor logs in and goes to booking history.	System Shows Booking List
2.	Actor selects booking to modify	
3.		System shows booking details and options.
4.	Actor selects desired changes	
5.		System checks feasibility
6.	The payment gateway processes the payment securely.	If available system updates booking and recalculates cost.
7.	Actor proceeds to payment/refund.	Upon successful payment, the system saves the booking to the database.

8.		System confirms update.
Alternate Scenarios: Write additional, optional, branching or iterative steps. Refer to specific action number to ensure understandability.		
1a: If update is not possible user is denied **2a:** Process restarts upon failure at any step		
Post Conditions		
Step#	**Description**	
1	Booking is updated in the system.	
Use Case Cross referenced	● View Booking History	

5. Non-functional Requirements

5.1. Performance Requirements

Response Time: The system should respond to user queries (e.g., flight or hotel search) within 2 seconds under normal load conditions.

Concurrent Users: The platform should support multiple users without degradation in performance.

Availability: The system must maintain **99.5% uptime**, ensuring it is accessible to users most of the time.

Capacity: The database should handle **1,000 records** each for flights, hotels, and bookings without performance issues.

Scalability: The system should support seamless scalability to accommodate increased user loads or data.

5.2. Safety Requirements

Error Handling: The system includes proper error-handling mechanisms to log and notify administrators of any critical issues.

Fail-Safe Mechanism: In the event of system crashes, user transactions remain consistent and recoverable without data corruption.

5.3. Security Requirements

User Authentication: All users must authenticate before accessing their account or making reservations.

Role-Based Access Control (RBAC): Admin functionalities (e.g., managing flights, hotels) should be restricted to authenticated admin accounts only.

Data Encryption: Sensitive user data (e.g., passwords) must be encrypted.

5.4. User Documentation

Documents:

- Software Requirements Specification.
- Software Design Specification

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