SRS-BACK-UP

MINISTRY OF EDUCATION AND TRAINING

POD BOOKING SYSTEM

An online office booking support system. Providing diverse services and comfortable spaces for users to choose and book an effective working space.

Project Code	SE1872G8
Document Code	SE1872G8 - v1.0
Group Member	Ha Xuan Khang - SE180105 Le Hai Dang - SE183661 Nguyen Pham Cong Hau - SE183665 Nguyen Ha Linh - SS170495
Supervisor	Le Thi Quynh Chi

Table of Content

1. INTRODUCTION

1.1 Purpose

The purpose of this System Requirements Specification (SRS) document is to provide detailed descriptions of the requirements and necessary functionalities of the **POD Booking System**. This document serves as the foundation for the development, deployment, and maintenance of the system, ensuring that all stakeholders (developers, users, and project managers) have a clear and unified understanding of the system's intended functionalities.

Defining the requirements in this SRS clarifies the concepts related to the system's functionality, features, and conditions that must be met, thereby ensuring consistency, minimizing the risk of unnecessary requirements, and improving the efficiency of system implementation. This document plays a crucial role in guiding the system's development and serves as a primary reference throughout the project lifecycle.

1.2 Scope

The **POD Booking System** is a web-based platform designed to streamline the process of searching, reserving, and managing workspace bookings. This software aims to facilitate a seamless experience for users—whether individuals or businesses—by allowing them to find and book available rooms according to their specific needs, such as location, amenities, and available services.

The system aligns with organizational goals of optimizing resource utilization and providing a high-value, flexible workspace solution. It supports business objectives by enhancing operational efficiency, meeting the demand for hybrid and remote work environments, and delivering a user-friendly booking experience that encourages repeated use.

This release of the POD Booking System will focus on key functionalities, including room search and filtering, availability display, booking management, and integrated payment processing. As part of a long-term vision, this version establishes a foundation for additional features, such as personalized recommendations, user profiles, and expanded service offerings.

1.3 Document Convention

Acronym	Definition
CRUD	acronym for Create-Read-Update-Delete, meaning "create, read, update, delete" are the four basic functions of persistent storage

1.4 References:

1.4.1 Refer SRS Template

Software_Requirements_3rd_Edition_compressed

https://www.booksfree.org/wp-content/uploads/2022/03/Software_Requirements_3rd_Edition_compressed.pdf

1.4.2 Refer Website

Regus

Không Gian Văn Phòng Tại Việt Nam | Regus

Cinestar

Cinestar - Hệ thống rạp chiếu phim giá rẻ, hiện đại bậc nhất

1.5 Overview:

This Software Requirements Specification (SRS) document outlines the functional and non-functional requirements of the POD Booking System, a web-based platform designed to facilitate the booking of flexible workspaces. The document is structured to provide a comprehensive understanding of the system's objectives, design, and implementation details, ensuring alignment among stakeholders and guiding the development process.

The SRS is organized as follows:

- Section 2 - Overall Description:
  This section provides a detailed overview of the system's intended use, target audience, and primary functionalities. It also covers the product perspective within the larger organizational context, specifies the operating environment (e.g., supported devices and browsers), and outlines design constraints. Additionally, it addresses assumptions, dependencies, and requirements that may vary based on user roles, such as Admin, Manager, Staff, and Guest.
- Section 3 - Functional Requirements:
  Lists and details the functional requirements of the POD Booking System, including user registration, login, booking creation, schedule management, room availability check, and profile management functionalities. This section breaks down requirements by user roles and specifies interactions within the system for each primary use case.
- Section 4 - Non-Functional Requirements:
  Defines the performance, security, usability, and reliability expectations for the system. This section addresses scalability, data privacy, response time, and system availability, ensuring that the platform meets business and user standards for a seamless experience.
- Section 5 - Supporting Information:
  Provides additional information, such as glossary definitions, references, and document control details. This section serves as a resource for any supporting materials that facilitate better understanding and ensure clarity among project contributors and stakeholders.

2. OVERALL DESCRIPTION

2.1 Product Perspective

The POD Booking System is an entirely new solution designed to facilitate flexible workspace reservations and management. Developed from scratch, this system aims to meet the growing demand for adaptable work environments by allowing users to search for, book, and manage workspace reservations through a user-friendly online platform.

As a standalone product, the system integrates with third-party services such as VNPay for secure online payment processing and Google Maps for location-based workspace searches, enhancing user convenience and ensuring secure transactions. These integrations provide a seamless booking experience by simplifying payment and navigation.

The POD Booking System operates independently but is scalable, with potential for future expansion or integration into a larger ecosystem of smart workplace management tools. This foundation enables rapid response to user needs in a growing market for flexible, on-demand workspaces.

2.2 User Classes and Characteristics

The **POD Booking System** anticipates several user classes, each with unique roles and access levels within the system. Each user class interacts with specific functions aligned to their needs and responsibilities. Below are the primary user classes and their characteristics:

1. **Customer**

- **Description**: Customers are the primary users of the POD Booking System. They can browse available buildings, rooms, and services to book suitable workspaces. Customers may include individuals or business representatives seeking flexible workspace solutions.
- **Characteristics**:
  - Requires an intuitive interface for searching, viewing, and filtering available spaces
  - Can perform functions such as booking rooms, adding services, selecting time slots, making payments, and managing their bookings.
- **Use Cases**:
  - Search Building, View Building List, View Building Detail, View Room List, View Room Detail, Booking, Add Service, Book Slot, Make Payment, Check-in, Check-out.

2. **Staff**
   - **Description**: Staff members facilitate the booking experience by supporting customer check-in, check-out.
   - **Characteristics**:
     - Needs access to the room and booking management features to efficiently assist with customer bookings.
   - **Use Cases**:
     - Check-in, Check-out, View Room List, View Room Detail.
3. **Manager**
   - **Description**: Managers oversee the workspace facilities, including managing building, room information and staff assignments.
   - **Characteristics**:
     - Requires access to features that support CRUD (Create, Read, Update, Delete) operations on building and room data to maintain accurate and up-to-date information for customers.
     - Responsible for ensuring the quality and availability of workspaces.
     - Has the capability to assign work locations and schedules to staff, ensuring optimal staffing levels and service coverage for different workspace facilities.
   - **Use Cases**:
     - Manage Building (CRUD), Manage Room (CRUD), Manage Staff.
4. **Admin**
   - **Description**: Admins handle system-wide operations, including managing user accounts and overseeing payment processes.
   - **Characteristics**:
     - Has comprehensive access to the platform's administrative tools for managing accounts, processing payments, and monitoring system analytics.
     - Responsible for the security and efficiency of user account and payment management.
   - **Use Cases**:
     - Manage Account (CRUD), View Dashboard, Manage Payment.
5. **Guest**
   - **Description**: Guests are unregistered users who can explore basic features but need to register or log in to perform booking-related actions.
   - **Characteristics**:
     - Limited access to features, primarily focusing on login and registration.
     - Requires a straightforward onboarding experience to become a registered user.
   - **Use Cases**:
     - Login and register.

2.3 Operating Environment

The POD Booking System will operate in a web-based environment accessible across multiple platforms, including desktop and mobile devices. The key components of its operating environment are as follows:

- **Hardware Platform:** The system will run on cloud-based servers hosted by a reliable cloud service provider, ensuring scalability and high availability for handling user traffic. User devices can access the system via modern web browsers on desktops, laptops, tablets, and smartphones.
- **Operating Systems:**
  - **Server:** The application servers will run on VPS.
  - **Client:** The system is compatible with all major operating systems, including Windows, macOS, iOS, and Android, as long as they support modern web browsers.
- **Database:** A MySQL database hosted on a cloud platform will store all relevant data, including user profiles, bookings, payments, and workspace details.
- **Geographical Locations:**
  - **Users:** The system is designed for users in Vietnam and potentially other Southeast Asian regions, where the demand for flexible workspaces is growing.
  - **Servers and Databases:** The servers and databases are hosted in a data center within Vietnam to comply with local data residency regulations and optimize performance for local users.
- **Third-Party Integrations:** The system will integrate with:
  - **VNPay** for secure payment processing, allowing users to make payments directly within the platform.
  - **Google Maps API** for providing accurate location services, enabling users to find nearby workspaces and obtain directions.
- **Supported Browsers:** The system will support all modern web browsers, including Google Chrome, Firefox, Safari, and Microsoft Edge, ensuring broad compatibility across different devices.

2.4 Design and implementation constraints

The development of the POD Booking System is subject to several design and implementation constraints that will guide the choice of technologies, frameworks, and methodologies. These constraints include:

1. **Programming Language and Framework:**
   - The system will be developed using **Java** with the **Spring Boot** framework. This choice is based on the team's familiarity with Java, the framework's support for rapid development, and its robust features for building scalable web applications.
   - **Rationale:** Spring Boot simplifies the development process by providing built-in functionalities such as dependency management, security features, and RESTful API support, which are crucial for the system's needs.
2. **Database Management System:**
   - The project will utilize **MySQL** as the relational database management system (RDBMS).
   - **Rationale:** MySQL is a widely used, reliable, and well-supported database, offering robust performance for transactional applications. Its familiarity among the development team also supports efficient implementation and maintenance.
3. **Payment Processing:**
   - Integration with **VNPay** is mandatory for payment processing within the system.
   - **Rationale:** VNPay is a popular payment gateway in Vietnam, providing a secure and convenient option for users. This integration is essential to ensure compliance with local payment regulations and to facilitate smooth financial transactions.
4. **User Interface Libraries:**
   - The development will leverage **React.js** for the frontend to create a dynamic and responsive user interface.
   - **Rationale:** React.js allows for efficient rendering and provides a component-based architecture that aligns with the project's requirements for user interaction and real-time updates.
5. **Compliance and Data Security:**
   - The system must adhere to local data protection regulations in Vietnam, including ensuring user data privacy and secure handling of payment information.
   - **Rationale:** Compliance with legal requirements is critical to maintain user trust and avoid potential legal issues related to data breaches or mismanagement of personal information.

2.5 Assumptions and Dependencies

2.5.1 Assumptions

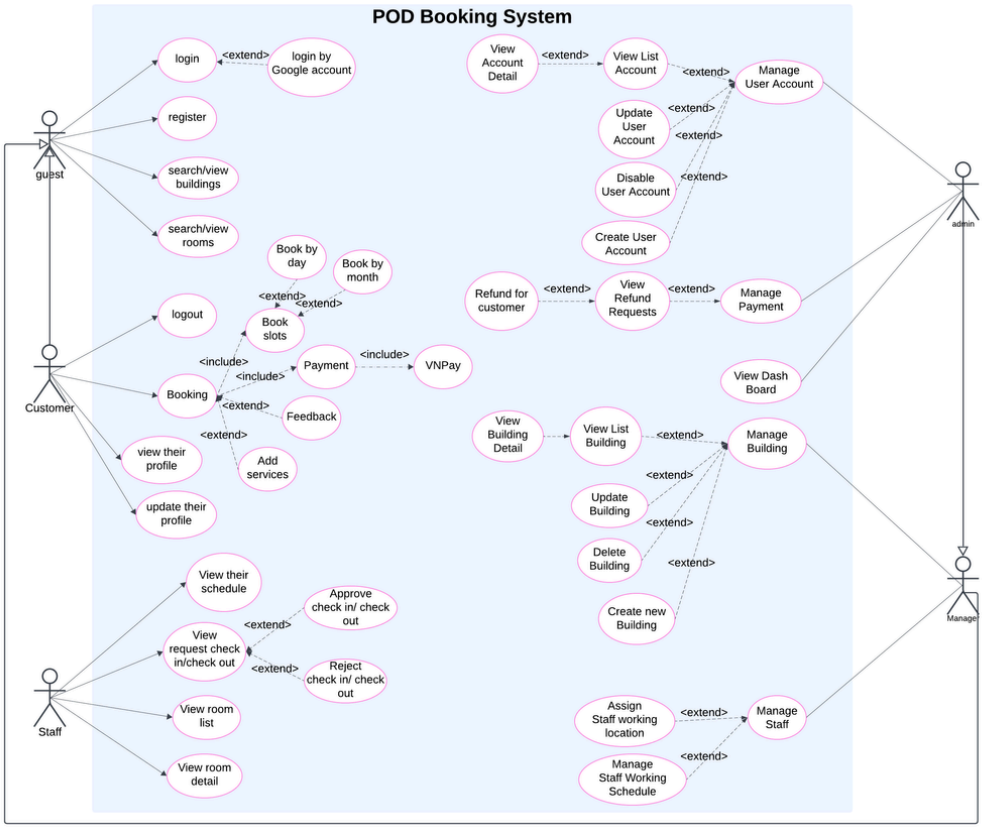
1. **User Access:** It is assumed that users will have access to the internet and a compatible device (desktop, laptop, tablet, or smartphone) to utilize the POD Booking System effectively.
   - **Risk:** If users lack reliable internet access or the necessary devices, it could limit system usage and customer engagement.
2. **User Familiarity with Technology:** It is assumed that the target users (customers, staff, managers, and admins) will have a basic understanding of using web applications.
   - **Risk:** A lack of technical proficiency among users may lead to difficulties in navigating the system, necessitating additional training or support.
3. **Payment Processing:** The project assumes that VNPay will function without significant issues and that users will be willing to use this payment method.
   - **Risk:** If VNPay experiences outages or if users prefer alternative payment methods not supported by the system, it could affect transaction volumes.
4. **Regulatory Compliance:** It is assumed that the system will remain compliant with local data protection and financial transaction regulations during its lifecycle.
   - **Risk:** Changes in legislation or regulatory requirements may necessitate unplanned system updates or redesigns to ensure compliance.
5. **Availability of Resources:** It is assumed that the development team will have consistent access to required resources, including hardware, software, and personnel.
   - **Risk:** Any disruption in resource availability could impact project timelines and deliverables.

2.5.2 Dependencies

1. **VNPay Payment Gateway:** The system's functionality depends on the availability and reliability of the VNPay payment gateway for processing transactions.
   - **Impact:** Any downtime or changes in VNPay's API could disrupt payment processing capabilities within the POD Booking System.
2. **Google Maps API:** The system depends on the Google Maps API for location-based services to help users find nearby office spaces.
   - **Impact:** Changes to the API, such as pricing or functionality, could impact the system's ability to provide location services.
3. **Browser Compatibility:** The system's operation is dependent on users accessing it via modern web browsers.
   - **Impact:** Changes in browser technologies or updates that impact compatibility could affect user experience and functionality.
4. **Compliance Frameworks:** The system depends on existing frameworks and standards for data security and privacy in Vietnam.
   - **Impact:** Changes in regulatory frameworks could require immediate adaptations to the system to maintain compliance and protect user data.

3. FUNCTIONAL REQUIREMENTS

3.1 Use case diagram



3.2 Use case List

ID	UseCase	Primary Actor	Secondary Actor
UC-01	Login	Guest	
UC-02	Register	Guest	
UC-03	Search/View buildings	Guest	
UC-04	Search/View Rooms	Guest	
UC-05	Logout	Customer	
UC-06	Booking	Customer	
UC-07	View their profile	Customer	
UC-08	Update their profile	Customer	
UC-09	View their schedule	Staff	
UC-10	View request check in/check out	Staff	
UC-11	View room list	Staff	
UC-12	View room detail	Staff	
UC-13	View list building	Manager	
UC-14	Update building	Manager	

UC-15	Delete building	Manager	
UC-16	Create new building	Manager	
UC-17	Assign staff working location	Manager	
UC-18	Manage staff working schedule	Manager	
UC-19	View List Account	Admin	
UC-20	Update user account	Admin	
UC-21	Disable user account	Admin	
UC-22	Create user account	Admin	
UC-23	Refund request	Admin	
UC-24	View dash board	Admin	

3.2 Use Case Detail

USE CASE-1 SPECIFICATION	
Use-case No:	UC-01
Use-case Version:	1.0
Use-case Name:	Login
Author:	Nguyễn Phạm Công Hậu
Date:	31/10/2024
Priority:	High
Actor:	Guest
Summary:	Users can log in to the system using their account information.
Goal:	Login successfully to access system features
Triggers:	The user wants to access their account.
Preconditions:	The user has registered an account and has valid login information.
Post Conditions:	User is redirected to main page after successful login
Normal Flow:	1. The user opens the login page. 2. The user enters their username and password. 3. The system validates the login information. 4. If valid, the user is redirected to the main page

Aternative Scenario:	If the login information is invalid, display an error message.
Exceptions:	The system does not respond due to a network error.
Business Rules:	Password must be at least 8 characters long, including uppercase, lowercase, and numbers.
USE CASE-2 SPECIFICATION	
Use-case No:	UC-02
Use-case Version:	1.0
Use-case Name:	Register
Author:	Nguyễn Phạm Công Hậu
Date:	31/10/2024
Priority:	High
Actor:	Guest
Summary:	The user can create a new account in the system.
Goal:	Successfully register an account to utilize the system's services.
Triggers:	The user wants to create a new account.
Preconditions:	The user does not have an account in the system.
Post Conditions:	The user has a new account and can log in.
Normal Flow:	1. The user opens the registration page. 2. The user enters the required information (name, email, password). 3. The system validates the information. 4. If valid, creates the account and sends a confirmation email.
Aternative Scenario:	If the information is invalid, display an error message.
Exceptions:	If the email is already in use, notify the user.
Business Rules:	The email address must be unique in the system

USE CASE-3 SPECIFICATION	
Use-case No:	UC-03
Use-case Version:	1.0
Use-case Name:	Search/View Buildings
Author:	Nguyễn Phạm Công Hậu

Date:	31/10/2024
Priority:	High
Actor:	Guest
Summary:	The user can search for and view information about available buildings.
Goal:	Provide comprehensive information about available buildings.
Triggers:	The user wants to search for buildings.
Preconditions:	The user has opened the search page.
Post Conditions:	Display a list of buildings matching the search criteria.
Normal Flow:	1. The user enters search criteria in the search bar. 2. The system queries the database and retrieves a list of matching buildings. 3. The list of buildings is displayed on the page.
Aternative Scenario:	If no buildings match, display a "not found" message.
Exceptions:	The system does not respond due to a database query error.
Business Rules:	Search criteria must be valid (e.g., cannot be empty).

USE CASE-4 SPECIFICATION	
Use-case No:	UC-04
Use-case Version:	1.0
Use-case Name:	Search/View Rooms
Author:	Nguyễn Phạm Công Hậu
Date:	31/10/2024
Priority:	High
Actor:	Guest
Summary:	The user can search for and view information about available rooms within buildings.
Goal:	Allow the user to find rooms that meet their specific needs and preferences.
Triggers:	The user wants to search for a room within a selected building.
Preconditions:	The user has selected a building to view available rooms or entered room search criteria.
Post Conditions:	Display a list of rooms matching the search criteria.

Normal Flow:	1. The user enters search criteria (e.g., room type, amenities) or selects a building. 2. The system queries the database for rooms that match the criteria. 3. A list of matching rooms is displayed, with details about each room. 4. The user can click on a room to view further details.
Aternative Scenario:	If no rooms match the criteria, display a message saying "No rooms found."
Exceptions:	The system fails to respond due to a network or database error.
Business Rules:	Room search filters must be clear and accurate to improve search relevancy.

USE CASE-5 SPECIFICATION	
Use-case No.	UC-05
Use-case Name	Logout
Author	Lê Hải Đăng
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Customer
Summary	This use case describes the process for a customer to log out of the POD Booking System. Logging out ensures that user session data is securely cleared and prevents unauthorized access to the user's account from shared or public devices.
Goal	The goal of this use case is to successfully terminate the user's session in the POD Booking System, ensuring that all sensitive data is cleared and the user is safely logged out.
Triggers	The use case is initiated when a customer decides to log out of their account, typically by clicking a "Logout" button in the user interface.
Preconditions	• The customer must be logged into the POD Booking System. • The session must be active and valid.
Post Conditions	• The customer's session is terminated. • The user is redirected to the login page or homepage.

	• Any sensitive data stored in the session is cleared.
Normal Flow	1. The customer navigates to the user interface of the POD Booking System. 2. The customer clicks on the "Logout" button. 3. The system confirms the logout action. 4. The system terminates the user session. 5. The system clears all session-related data. 6. The system redirects the customer to the login page or homepage. 7. The customer is presented with the message confirming successful logout.
Alternative Flow	If the user session has already expired, the system automatically redirects the user to the login page without requiring an explicit logout
Exceptions	If a network issue occurs during the logout process, the system will display an error message informing the user of the connectivity issue and suggest trying again.
Business Rules	• The system must ensure that any sensitive user data is securely cleared upon logout. • The system should implement session timeout rules; if the customer remains inactive for a predefined period, the system automatically logs them out. • Customers must be provided with clear feedback about their logout status to enhance user experience.

USE CASE-6 SPECIFICATION	
Use-case No.	UC-06
Use-case Name	Booking
Author	Lê Hải Đăng
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Customer
Summary	The user can book a room according to their needs through the system. The system will process the booking information and confirm the booking status.

Goal	Help customers successfully book a room for the desired date and time.
Triggers	The user selects the "Booking now!" option on the interface.
Preconditions	• The customer is logged into the system. • The customer has permission to book a room.
Post Conditions	• The booking is successfully made, and the information is saved in the system. • The customer receives a booking confirmation.
Normal Flow	1. The customer selects the desired time(Hours/Day/Month). 2. The customer confirms the booking information. 3. The system checks the room's availability. 4. The customer completes the payment. 5. The system confirms and saves the booking information.
Alternative Flow	N/A
Exceptions	If the selected room is not available for the specified date and time, the system will display an error message and suggest alternative dates or rooms.
Business Rules	1. Only logged-in customers with valid accounts can make a booking. 2. All bookings require full payment confirmation before they are finalized. 3. After payment, the system must confirm the booking and send a email message to the customer.

USE CASE-7 SPECIFICATION	
Use-case No.	UC-07
Use-case Name	View their profile
Author	Lê Hải Đăng
Date	31/10/2024
Priority	Normal
Use-case Version	1.0
Actor	Customer
Summary	The user can view their personal information and related account details.
Goal	Allow customers to view and check their account information.

Triggers	The user selects the "My Profile" option on the interface.
Preconditions	• The customer is logged into the system.
Post Conditions	• The customer’s profile information is displayed on the interface.
Normal Flow	1. The customer accesses the "My Profile" section. 2. The system displays the customer's personal information, including name, email, phone number, username, and other details.
Alternative Flow	N/A
Exceptions	If the user does not have permission to view the profile, the system will display an access denied message and prevent further actions.
Business Rules	Profile information must be secure and accessible only by logged-in customers.

USE CASE-8 SPECIFICATION	
Use-case No.	UC-08
Use-case Name	Update their profile
Author	Lê Hải Đăng
Date	31/10/2024
Priority	Normal
Use-case Version	1.0
Actor	Customer
Summary	The user can update their personal information, including username, name, email, password, and phone number.
Goal	Allow customers to modify and update their account information.
Triggers	The user selects the "Update Profile" option on the interface.
Preconditions	• The customer is logged into the system. • Updated information has been provided.
Post Conditions	• Profile information is successfully updated in the system.
Normal Flow	1. The customer accesses the "Update Profile" section. 2. The customer enters new information.

	3. The system verifies the information and saves the changes. 4. The customer receives a successful update notification.
Alternative Flow	N/A
Exceptions	If the user enters invalid information, the system will display an error message specifying the issue and prompt the user to correct the information before resubmitting.
Business Rules	Customers can only update their own personal information.

USE CASE-9 SPECIFICATION	
Use-case No.	UC-09
Use-case Name	View Their Schedule
Author	Lê Hải Đăng
Date	31/10/2024
Priority	Normal
Use-case Version	1.0
Actor	Staff
Summary	Staff members can view their daily or weekly work schedule, including assigned tasks and bookings they are responsible for managing.
Goal	Enable staff to check and manage their schedules efficiently.
Triggers	The staff selects the "View Schedule" option in their dashboard.
Preconditions	• The staff member is logged into the system. • The schedule information is available in the system.
Post Conditions	• The staff's schedule is displayed on the screen.
Normal Flow	1. The staff accesses the "View Schedule" section. 2. The system retrieves and displays the staff's schedule, including upcoming tasks and bookings.
Alternative Flow	N/A
Exceptions	If the staff member does not have permission to view their schedule, the system will display an access denied message and prevent further actions.

Business Rules	The schedule should be up-to-date and accurate for effective management.
USE CASE-10 SPECIFICATION	
Use-case No.	UC-10
Use-case Name	View Request Check-In/Check-Out
Author	Lê Hải Đăng
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Staff
Summary	Staff members can view the list of check-in and check-out requests submitted by customers, allowing them to approve or reject these requests.
Goal	Assist staff in managing customer check-in/check-out requests.
Triggers	The staff selects the "View Check-In/Check-Out Requests" option.
Preconditions	• The staff member is logged into the system. • Customer check-in/check-out requests are available in the system.
Post Conditions	• The list of check-in/check-out requests is displayed.
Normal Flow	1. The staff accesses the "View Check-In/Check-Out Requests" section. 2. The system retrieves and displays a list of pending check-in/check-out requests. 3. The staff reviews each request and takes the necessary action (approve/reject).
Alternative Flow	N/A
Exceptions	If there is a network connectivity issue while loading check-in/check-out requests, the system will display an error message informing the staff member of the connectivity problem and suggest refreshing the page or trying again later.
Business Rules	Only authorized staff members can approve or reject requests.

USE CASE-11 SPECIFICATION	
Use-case No.	UC-11

Use-case Name	View Room List
Author	Lê Hải Đăng
Date	31/10/2024
Priority	Normal
Use-case Version	1.0
Actor	Staff
Summary	Staff can view the list of available rooms and their statuses, helping them manage bookings and room assignments.
Goal	Provide staff with an overview of room availability and statuses.
Triggers	The staff selects the "View Room List" option.
Preconditions	• The staff member is logged into the system
Post Conditions	• The list of rooms and their statuses is displayed.
Normal Flow	1. The staff accesses the "View Room List" section. 2. The system retrieves and displays the list of rooms, including room status.
Alternative Flow	N/A
Exceptions	N/A
Business Rules	Only staff members can access the detailed room list view.

USE CASE-12 SPECIFICATION	
Use-case No.	UC-12
Use-case Name	View Room Detail
Author	Lê Hải Đăng
Date	31/10/2024
Priority	Normal
Use-case Version	1.0
Actor	Staff
Summary	Staff can view detailed information about a specific room, including room type, amenities, and maintenance history.
Goal	Help staff understand the details of each room for effective management and customer service.

Triggers	The staff selects a specific room.
Preconditions	• The staff member is logged into the system. • The room information is available in the system.
Post Conditions	• The detailed information of the selected room is displayed.
Normal Flow	1. The staff selects a specific room. 2. The system retrieves and displays detailed information about the selected room.
Alternative Flow	N/A
Exceptions	N/A
Business Rules	Only staff members can access detailed room information.

USE CASE-13 SPECIFICATION	
Use-case No.	UC-13
Use-case Name	View List Building
Author	Nguyễn Phạm Công Hậu
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Manager
Summary	This use case allows the manager to view a list of all buildings under their management. The manager can see general information about each building, such as name, location, and status.
Goal	To provide the manager with a comprehensive list of buildings, enabling them to quickly access building details.
Triggers	The manager selects the "View List Building" option in the system's interface.
Preconditions	1. The manager must be logged into the system. 2. The system should have at least one building record available.
Post Conditions	1. The system displays a list of buildings. 2. The manager can view the summary details of each building.
Normal Flow	1. The manager logs into the system. 2. The manager navigates to the building management section.

	3. The manager selects "View List Building." 4. The system retrieves the list of buildings from the database. 5. The system displays the list of buildings, showing relevant details for each one. 6. The manager reviews the list of buildings.
Alternative Flow	If there are no buildings in the database, the system displays a message indicating that there are no buildings to show.
Exceptions	If the manager is not logged in, the system redirects them to the login page. If there is an error retrieving building data, the system displays an error message and suggests retrying later.
Business Rules	1. Only authorized managers can access the list of buildings. 2. Building data should be accurate and updated in real-time if there are changes.

USE CASE-14 SPECIFICATION	
Use-case No.	UC-14
Use-case Name	Update Building
Author	Nguyễn Phạm Công Hậu
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Manager
Summary	This use case allows the manager to update information for an existing building. The manager can edit details such as the building name, location, description, and status.
Goal	To enable the manager to modify and maintain up-to-date information for buildings.
Triggers	The manager selects a building from the list and chooses the "Update" option.
Preconditions	1. The manager must be logged into the system. 2. The building record to be updated must exist in the database.
Post Conditions	1. The building record is updated with the new information.

	2. The updated building details are displayed in the system.
Normal Flow	1. The manager logs into the system. 2. The manager navigates to the building management section. 3. The manager selects a building from the list. 4. The manager chooses the "Update" option. 5. The system displays the building's current details in an editable form. 6. The manager modifies the desired information (e.g., name, location, description). 7. The manager submits the updated information. 8. The system validates the input data. 9. The system saves the changes and updates the building record in the database. 10. The system displays a confirmation message and shows the updated building details.
Alternative Flow	If the input data fails validation (e.g., invalid characters or missing required fields), the system highlights the issues and prompts the manager to correct the data before resubmitting.
Exceptions	1. If the building selected does not exist, the system displays an error message and returns to the building list. 2. If there is an error saving the changes to the database, the system displays an error message and suggests retrying later.
Business Rules	1. Only authorized managers can update building details. 2. All required fields must be completed with valid data for a successful update.

USE CASE-15 SPECIFICATION	
Use-case No.	UC-15
Use-case Name	Delete Building
Author	Nguyễn Phạm Công Hậu
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Manager
Summary	This use case allows the manager to delete an existing building record from the system. This action removes the building information permanently.

Goal	To enable the manager to remove buildings that are no longer needed or relevant..
Triggers	The manager selects a building from the list and chooses the "Delete" option.
Preconditions	1. The manager must be logged into the system. 2. The building record to be deleted must exist in the database.
Post Conditions	1. The building record is permanently removed from the system. 2. The updated list of buildings (excluding the deleted one) is displayed to the manager.
Normal Flow	1. The manager logs into the system. 2. The manager navigates to the building management section. 3. The manager selects a building from the list. 4. The manager chooses the "Delete" option. 5. The system displays a confirmation prompt to ensure the manager wants to delete the building. 6. The manager confirms the deletion. 7. The system deletes the building record from the database. 8. The system displays a confirmation message indicating successful deletion. 9. The updated list of buildings is displayed, excluding the deleted building.
Alternative Flow	If the manager cancels the deletion at the confirmation prompt, the system aborts the deletion process and returns to the building list without making changes.
Exceptions	1. If the building selected does not exist, the system displays an error message and returns to the building list. 2. If there is an error deleting the building from the database, the system displays an error message and suggests retrying later.
Business Rules	1. Only authorized managers can delete building records. 2. Deletion is permanent and cannot be undone, so confirmation is required.

USE CASE-16 SPECIFICATION	
Use-case No.	UC-16
Use-case Name	Create New Building

Author	Nguyễn Phạm Công Hậu
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Manager
Summary	This use case allows the manager to create a new building record in the system. The manager can input details such as building name, location, description, and status.
Goal	To enable the manager to add new buildings to the system's database.
Triggers	The manager selects the "Create New Building" option from the building management section.
Preconditions	The manager must be logged into the system.
Post Conditions	1. The new building is saved in the system’s database. 2. The updated list of buildings, including the newly created building, is displayed to the manager.
Normal Flow	1. The manager logs into the system. 2. The manager navigates to the building management section. 3. The manager selects the "Create New Building" option. 4. The system displays a form for entering building details (e.g., name, location, description). 5. The manager fills in the required fields and submits the form. 6. The system validates the input data. 7. The system saves the new building information in the database. 8. The system displays a confirmation message indicating successful creation. 9. The updated list of buildings is shown, including the newly added building.
Alternative Flow	If the input data fails validation (e.g., missing required fields or invalid entries), the system highlights the issues and prompts the manager to correct the data before resubmitting.
Exceptions	If there is an error saving the new building to the database, the system displays an error message and suggests retrying later.
Business Rules	1. Only authorized managers can create new building records.

	2. All required fields must be completed with valid data to create a new building successfully.

USE CASE-17 SPECIFICATION	
Use-case No.	UC-17
Use-case Name	Assign Staff Working Location
Author	Nguyễn Phạm Công Hậu
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Manager
Summary	This use case allows the manager to assign specific buildings as working locations for staff members. The manager can select a staff member and designate one or more buildings for their assignments.
Goal	To enable the manager to allocate staff members to specific building locations based on operational needs.
Triggers	The manager selects a staff member from the staff management section and chooses the "Assign Working Location" option.
Preconditions	1. The manager must be logged into the system. 2. The staff member to be assigned must exist in the system. 3. The building(s) to be assigned must exist in the system.
Post Conditions	1. The staff member is successfully assigned to the selected building(s) as their working location(s). 2. The updated assignment information is displayed to the manager.
Normal Flow	1. The manager logs into the system. 2. The manager navigates to the staff management section. 3. The manager selects a staff member from the list. 4. The manager chooses the "Assign Working Location" option. 5. The system displays a list of available buildings. 6. The manager selects one or more buildings as the staff member’s working location(s). 7. The manager confirms the selection and submits the assignment.

	8. The system saves the updated working location information for the staff member. 9. The system displays a confirmation message indicating successful assignment. 10. The updated working location assignments are shown in the staff member's profile.
Alternative Flow	If no buildings are available or displayed in the list, the system informs the manager and suggests checking the building list or adding new buildings.
Exceptions	If there is an error saving the assignment in the database, the system displays an error message and suggests retrying later.
Business Rules	1. Only authorized managers can assign working locations to staff members. 2. Each staff member can be assigned multiple buildings as working locations if necessary.

USE CASE-18 SPECIFICATION	
Use-case No.	UC-18
Use-case Name	Manage Staff Working Schedule
Author	Nguyễn Phạm Công Hậu
Date	31/10/2024
Priority	High
Use-case Version	1.0
Actor	Manager
Summary	This use case allows the manager to set, update, or view the working schedules for staff members assigned to specific buildings. The manager can assign work hours and days based on operational requirements.
Goal	To enable the manager to define or modify the working schedules for staff members.
Triggers	The manager selects a staff member from the staff management section and chooses the "Manage Working Schedule" option.
Preconditions	1. The manager must be logged into the system. 2. The staff member to be scheduled must exist in the system. 3. The staff member must have at least one assigned working location.
Post Conditions	1. The staff member's schedule is successfully updated and saved in the system.

	2. The updated schedule information is displayed to the manager.
Normal Flow	1. The manager logs into the system. 2. The manager navigates to the staff management section. 3. The manager selects a staff member from the list. 4. The manager chooses the "Manage Working Schedule" option. 5. The system displays the current working schedule for the staff member. 6. The manager selects the days and hours to assign or modify the schedule. 7. The manager confirms and submits the updated schedule. 8. The system saves the schedule updates in the database. 9. The system displays a confirmation message indicating successful scheduling. 10. The updated schedule is shown in the staff member’s profile.
Alternative Flow	IIf the manager does not select any days or hours, the system prompts for a valid schedule before proceeding.
Exceptions	If there is an error saving the schedule to the database, the system displays an error message and suggests retrying later.
Business Rules	1. Only authorized managers can manage staff schedules. 2. The staff working schedule must comply with company regulations regarding hours and workdays.

USE CASE 19 - SPECIFICATION	
Use-case No.	UC-19
Use-case Version	<1.0>
Use-case Name	View List Account
Author	Ha Xuan Khang
Date	10/31/2024
Priority	High
Actor	Admin
Summary	This use case allows the user (Admin) to view a list of all existing accounts in the system. This function

	supports the Admin in managing and tracking basic user information.
Goal	Provides users with the ability to view a list and details of all user accounts in the system.
Trigger	Admin selects the "View list of user accounts" function from the administration system interface.
Preconditions	• PRE-01: User login to system with role ADMIN
Postconditions	• POST-01: System show table include list of accounts
Normal Flow	1. Login with admin account 2. Click on “Accounts” Button
Alternative Flow	N/A
Exception	N/A
Business Rules	1. Only ADMIN can access list user account 2. System do not show user’s password in this table

USE CASE 20 - SPECIFICATION	
Use-case No.	UC-20
Use-case Version	<1.0>
Use-case Name	Update User Account
Author	Ha Xuan Khang
Date	10/31/2024
Priority	Normal
Actor	Admin
Summary	This use case allows the admin to update user account information within the system. This function supports the admin in managing and modifying user account details when necessary..
Goal	Provide the admin with the ability to edit and update details of user accounts within the system
Trigger	Admin selects the "Update User Account" button from the system’s administration interface.
Preconditions	• PRE-01: User login to system with role ADMIN
Postconditions	• POST-01: The system updates the account information and saves changes to the database. • POST-02: The updated information is displayed to the admin upon successful saving.
Normal Flow	1. Login with ADMIN account

	2. Admin accesses the account list and selects the user account to update 3. Admin edits the necessary account information, including fields like name, email, account status, address. 4. Admin clicks the "Save" button to save changes. 5. The system saves the changes and displays a success message.
Alternative Flow	N/A
Exception	N/A
Business Rules	1. Only ADMIN are permitted to update user account information. 2. The system does not allow the admin to update the user's password directly from this interface.

USE CASE 21 - SPECIFICATION	
Use-case No.	UC-21
Use-case Version	<1.0>
Use-case Name	Disable User Account
Author	Ha Xuan Khang
Date	10/31/2024
Priority	Normal
Actor	Admin
Summary	This use case allows the admin to disable a user account in the system. Disabling an account restricts the user from logging in or accessing any features until the account is reactivated by the admin
Goal	To provide the admin with the capability to disable user accounts as part of account management and security measures.
Trigger	Admin selects the "Disable User Account" button from the system's administration interface.
Preconditions	• PRE-01: User login to system with role ADMIN • PRE-02: The account to be disabled exists in the system and is currently active
Postconditions	• POST-01: The user account is marked as "disabled" in the system database. • POST-02: The user is prevented from logging in or performing any actions until reactivated.
Normal Flow	1. Login with ADMIN account

	2. Admin accesses the account list and selects the user account to disable 3. Admin clicks on the "Disable Account" button. 4. Admin clicks the "Save" button to save changes 5. The system prompts the admin to confirm the action. 6. Admin confirms the action. 7. The system disables the account, preventing the user from logging in or accessing any features 8. The system displays a success message indicating the account has been disabled.
Alternative Flow	N/A
Exception	1. If the account is already disabled, the system will display an error message and prevent further disabling actions.
Business Rules	1. Only ADMIN are permitted to disable user account. 2. Disabled accounts cannot log in or access any system features until reactivated

USE CASE 22 - SPECIFICATION	
Use-case No.	UC-22
Use-case Version	<1.0>
Use-case Name	Create User Account
Author	Ha Xuan Khang
Date	10/31/2024
Priority	Normal
Actor	Admin
Summary	This use case allows the admin to create a new user account in the system. This function supports the admin in managing users by adding new accounts as required
Goal	To enable the admin to create and configure new user accounts in the system
Trigger	Admin selects the "Create New User Account" function from the administration interface
Preconditions	• PRE-01: User login to the system with role ADMIN • PRE-02: User Phone Number is not existed in the database

Postconditions	• POST-01: A new user account is successfully created and stored in the system's database • POST-02: The new account is accessible in the account list and available for login
Normal Flow	1. Login with ADMIN account 2. Admin accesses the account list 3. Admin clicks on the "Create new Account" button. 4. Admin enters required information, such as phone, username, email, password, and role 5. Admin clicks the "Save" button to save changes 6. The system prompts the admin to confirm the action. 7. Admin confirms the action. 8. The system displays a success message
Alternative Flow	N/A
Exception	1. If phone number have been existed, the system show message "User have been existed" 2. If required field is missed, show message "Please enter this information" below each field
Business Rules	1. Only ADMIN are permitted to disable user account. 2. Phone number must be unique for each account 3. Password must at least 8 character include at least 1 number character
USE CASE 23 - SPECIFICATION	
Use-case No.	UC-23
Use-case Version	<1.0>
Use-case Name	Request for Refund
Author	Ha Xuan Khang
Date	10/31/2024
Priority	Normal
Actor	Admin
Summary	This use case allows the admin to send a refund request to the bank when a customer requests to cancel a reservation 24 hours before check-in time. Admin will search for order information based on the information provided by the customer to verify the order and process the refund request.
Goal	Ensure the refund process is implemented for valid requests, provides good customer service, and protects the interests of both the system and users.

Trigger	Admin selects the “Refund Request” function from the system admin interface
Preconditions	• PRE-01: User login to the system with role ADMIN • PRE-02: The customer has paid for the order and submitted a request to cancel the reservation • PRE-03: Cancellation must be 24 hours before check-in time
Postconditions	• POST-01: Refund request is sent to the bank • POST-02: Refund request information is stored in the system for later checking
Normal Flow	1. Login with ADMIN account 2. Admin selects the “Refund Request” function in the booking management section. 3. Admin enters the required information to search for the order. 4. The system verifies that the cancellation time is valid 5. The system retrieves the payment information from the database based on the customer’s details. 6. Admin confirms the order information and selects “Submit Refund Request.”. 7. The system submits the refund request to the bank 8. The system displays a success message and saves the refund request information
Alternative Flow	1. Login with ADMIN account 2. Admin Click on refund request on notification board 3. The system verifies that the cancellation time is valid 4. The system retrieves the payment information from the database based on the customer’s details. 5. Admin confirms the order information and selects “Submit Refund Request.”. 6. The system submits the refund request to the bank 7. The system displays a success message and saves the refund request information
Exception	1. If the system determines that the order is ineligible for a refund (e.g., cancellation not made within 24 hours of check-in), the system displays an error message and prompts the admin to inform the customer.

Business Rules	1. Only the Admin can submit refund requests. 2. Customers are only eligible for a refund if the cancellation request is made at least 24 hours before the check-in time 3. All refund requests must be recorded in the system for future reference.
USE CASE 24 - SPECIFICATION	
Use-case No.	UC-24
Use-case Version	<1.0>
Use-case Name	View Dash Board
Author	Ha Xuan Khang
Date	10/31/2024
Priority	High
Actor	Admin
Summary	This use case allows the admin to view the system's revenue summary on the dashboard. This function provides the admin with a quick overview of the system's financial performance.
Goal	To enable the admin to access and view the revenue summary of the system through the dashboard interface.
Trigger	Admin log in to admin home page
Preconditions	• PRE-01: User login to the system with role ADMIN • PRE-02: Revenue data is available in the system database
Postconditions	• POST-01: The dashboard successfully displays the revenue summary • POST-02: The displayed data is up-to-date and accurate.
Normal Flow	1. Login with ADMIN account 2. The system retrieves revenue data from the database.. 3. The system displays the revenue summary on the dashboard, including key metrics like total revenue, monthly trends, and profit margins. 4. Admin views the displayed information
Alternative Flow	N/A
Exception	1. If the revenue data is not available, the system will show the default value is 0

	2. If there is an error retrieving the data, the system displays "Error loading dashboard data."
Business Rules	1. Only the Admin can access the dashboard. 2. Revenue data displayed must be up-to-date 3. Dashboard metrics should be easily interpretable, with clear labels for each data point..

4. NON-FUNCTIONAL REQUIREMENTS

4.1 Usability

1. **Training Time:**
   - **Normal Users (Guests and Customers):** A new user should be able to navigate core functions—such as searching for workspaces, viewing availability, and making bookings—within 15 minutes of first use, without requiring formal training.
   - **Power Users (Staff, Managers, and Admins):** Users with administrative roles should become proficient in managing bookings, processing payments, and generating reports within 1-2 hours of training.
2. **Task Completion Time:**
   - **Booking Process:** A user should be able to complete a booking within 2-3 minutes, including selecting a workspace, choosing the date and time, and processing the payment.
   - **Search Functionality:** Searching for available workspaces or filtering results based on location, amenities, or availability should take no more than 10 seconds per query.
   - **Admin and Staff Functions:** Administrative tasks like approving bookings, generating reports, or updating workspace details should be manageable within 3-5 minutes each.
3. **User Interface Standards:**
   - The system interface will conform to common usability standards, aligning with established guidelines like Microsoft's GUI standards to ensure consistency and ease of navigation.
   - Clear icons, consistent layout, and a responsive design will be applied across desktop and mobile platforms to ensure accessibility and a seamless user experience.
4. **Error Prevention and Guidance:**
   - The system will provide users with immediate feedback and validation checks to prevent errors.
   - Tooltips, error messages, and clear instructions will guide users through each step, ensuring that tasks are completed with minimal confusion.
5. **Accessibility:**
   - The system will follow accessibility guidelines to accommodate users with disabilities, including support for screen readers, keyboard navigation, and color contrast adjustments for visually impaired users

4.2 Reliability

- **Availability**:
  - The system will maintain an availability of **99.9%**, allowing for approximately 8.76 hours of downtime annually for maintenance and updates.
  - Operating hours are 24/7 to accommodate global users, with scheduled maintenance occurring during off-peak hours (usually midnight to 4 AM local time).
- **Mean Time Between Failures (MTBF)**:
  - The system is expected to have a MTBF of at least **500 hours**, which aligns with an expectation of high uptime and reliability for continuous operation.
- **Mean Time to Repair (MTTR)**:
  - In the event of failure, the system's MTTR is set to be less than **1 hour**, aiming to restore full functionality swiftly to reduce user impact and maintain service continuity.
- **Accuracy**:

- All booking-related data, including workspace availability, booking time slots, and pricing, must be accurate within **99.99% precision**, ensuring that information presented to users is reliable and up-to-date.
- Payment processing accuracy is also critical, and integration with VNPay will ensure transactions are recorded with a failure margin below **0.01%**.

- **Maximum Bugs or Defect Rate**:
  - The defect rate is targeted to be no more than **0.5 bugs per thousand lines of code (KLOC)**, aiming for a minimal number of issues post-deployment.
- **Bugs or Defect Rate by Severity**:
  - **Critical Bugs**: Defined as defects causing complete service unavailability, data corruption, or inability to complete transactions. The goal is zero critical bugs during production, with any found requiring an immediate fix.
  - **Significant Bugs**: Issues affecting user experience or minor functionalities, such as delays in display or layout inconsistencies. These will be kept under **0.1 bugs per function-point**, with fixes prioritized within regular update cycles.
  - **Minor Bugs**: Cosmetic or non-critical errors, such as typos, which do not affect core functionality. These are expected to be below **0.05 bugs per function-point** and will be resolved in standard maintenance updates.

4.3 Performance

- **Response Time**:
  - **Booking Transaction**: The average response time for completing a booking transaction is expected to be under **2 seconds**, with a maximum of **5 seconds** during peak usage.
  - **Search Queries**: Searching for workspaces or viewing availability will respond within **1 second on average** and no more than **3 seconds** during high traffic.
  - **Payment Processing**: The response time for transactions processed through VNPay will be under **4 seconds** on average, ensuring a smooth and prompt payment experience.
- **Throughput**:
  - The system is expected to handle up to **100 transactions per second (TPS)**, including search requests, bookings, and other user interactions.
  - During peak usage, the system should support a minimum throughput of **80 TPS** without performance degradation.
- **Capacity**:
  - The system is designed to support **up to 10,000 concurrent users** without impacting performance, ensuring a stable experience even with high demand.
  - The system can manage **up to 1 million bookings per month**, meeting expected usage patterns and growth over time.
- **Degradation Modes**:
  - In degraded mode, such as during high traffic or partial system downtime, the system will continue to support essential functions (e.g., booking and payments) with **response times extending up to 7 seconds** temporarily.
  - Non-critical functions (e.g., reporting and analytics) may be temporarily paused or restricted during degraded modes to prioritize core functionalities.

4.4 Supportability

- **Coding Standards**:
  - All code will adhere to the **Java coding standards** for backend development, following best practices in readability, commenting, and modularization. Specific guidelines will ensure that classes, methods, and variables are clearly defined, facilitating easier debugging and collaboration.
- **Documentation and Commenting**:
  - Clear inline comments and structured documentation will be provided for each module, covering purpose, function, and integration points. This documentation aids maintenance and helps new developers understand the system faster.
  - Comprehensive **API documentation** will be generated using Swagger, providing a centralized reference for all available endpoints, methods, and data structures, essential for both developers and support staff.
- **Class Libraries and Reusable Components**:

    - The system will leverage **Spring Boot libraries** and reusable components to facilitate modular and consistent functionality across services. Standard libraries like Spring Security, Spring Data JPA, and Spring Web will be used to streamline development and support.
    - Reusable service and utility classes will be organized in the project to support common functionality, such as database access, input validation, and error handling, which helps reduce duplication and simplifies code maintenance.
- **Maintenance Access**:
    - An **admin dashboard** will provide authorized users with direct access to monitor system performance, track user activity, and manage booking data. This tool will enable quick troubleshooting and support, allowing maintenance staff to address issues efficiently.
    - Logging mechanisms using SLF4J and Logback will capture detailed records of system activities and errors, facilitating troubleshooting and proactive monitoring of potential issues.
- **Maintenance Utilities**:
    - The system will integrate **automated backup utilities** for regular data backup, ensuring minimal data loss in case of failure. Backups are scheduled during off-peak hours to avoid impacting performance.
    - A **scheduled task framework** will be in place for automated maintenance tasks such as clearing temporary data, managing session logs, and archiving old records, enhancing system performance and stability over time.

4.5 Design Constraints

- **Programming Language**:
    - The backend will be developed using **Java** and specifically the **Spring Boot framework** to provide a robust, modular, and maintainable application structure. This language choice aligns with existing system components and team expertise, promoting efficient development and maintenance.
- **Frontend Framework**:
    - The user interface will be built using **ReactJS** for responsive and interactive web applications, ensuring a user-friendly experience and supporting seamless integration with backend APIs.
- **Database Management System**:
    - **MySQL** has been selected as the relational database management system to store all application data. This choice is driven by its reliability, scalability, and support for complex queries, as well as the team's familiarity with MySQL.
- **Development Tools**:
    - **IntelliJ IDEA** is the mandated development environment for Java code due to its strong integration with Spring Boot, code debugging tools, and version control capabilities.
    - The project will use **Docker** for containerization, facilitating consistent deployment across environments and simplifying dependency management.
- **Authentication and Security Standards**:
    - User authentication and authorization processes will adhere to **OAuth 2.0 standards**, ensuring secure access management.
    - **HS512** (HMAC with SHA-512) will be used for encoding sensitive information, providing a high level of cryptographic security.
    - **BCryptPasswordEncoder** will be used for password hashing, adding an additional layer of security and protecting user credentials from unauthorized access.
- **API and Data Communication**:
    - All communication between frontend and backend will occur through **RESTful APIs** designed with scalability in mind. APIs must conform to RESTful standards, ensuring compatibility with HTTP methods (GET, POST, PUT, DELETE) and predictable interaction patterns.
    - The system will implement **JSON** for data exchange, which is lightweight, easy to parse, and widely supported.
- **Architectural Pattern**:
    - The project will follow the **microservices architecture** where each module (e.g., booking, payments, notifications) operates as an independent service. This design decision ensures scalability and simplifies the integration of future components.
- **Third-party Integrations**:

- The system will integrate with **VNPay** for online payment services, adhering to VNPay's API standards and security requirements to ensure secure and reliable payment transactions.
- **Google Maps API** will be used to enable location-based features, such as finding nearby buildings and viewing them on a map interface.

4.6 On-line User Documentation and Help System Requirements

- **User Manual**:
  - An online user manual will be provided, detailing system functionalities, step-by-step guides for using various features, and troubleshooting common issues.
  - The manual will be accessible from the main navigation menu within the application.
- **Searchable Knowledge Base**:
  - A searchable knowledge base will be implemented to allow users to quickly find answers to frequently asked questions (FAQs) and common operational queries.
  - The knowledge base will be organized by topics such as **Account Management**, **Booking Process**, **Payments**, and **Troubleshooting**.
- **Help and Support Contact**:
  - The system will include a help and support section that provides users with contact information for customer support, including email, phone number, and live chat options.
  - A feedback form will also be available to gather user suggestions or report issues.
- **Regular Updates**:
  - The online documentation will be regularly updated to reflect changes in the system, new features, and user feedback.
  - A version history will be maintained to inform users of recent changes and updates to the documentation.

4.7 Purchased Components

4.7.1. Payment Processing Component

- **Component Name**: VNPay Payment Gateway
- **Description**: VNPay will be integrated to facilitate secure online payments for bookings. This component will handle various payment methods, including credit cards, debit cards, and e-wallets.
- **Licensing**: The integration will be subject to VNPay's licensing terms, which allow for commercial use within the POD Booking System.
- **Compatibility**: The VNPay API is compatible with RESTful architecture, ensuring easy integration with the Spring Boot backend of the system.
- **Interoperability Standards**: The component will comply with PCI-DSS (Payment Card Industry Data Security Standard) to ensure secure handling of payment information.

4.7.2. Map Service Component

- **Component Name**: Google Maps API
- **Description**: The Google Maps API will be used for displaying building locations, helping users visualize and navigate to their chosen workspace.
- **Licensing**: This API requires a Google Cloud Platform account, and usage is subject to Google's pricing and usage limits based on the number of requests made.
- **Compatibility**: The API will be utilized in conjunction with both the frontend (ReactJS) and backend (Spring Boot) systems.
- **Interoperability Standards**: The component supports standard HTTP requests and JSON responses, ensuring seamless integration with the application.

4.7.3. Authentication and Security Component

- **Component Name**: Spring Security

- **Description**: Spring Security will be employed to manage user authentication and authorization, implementing OAuth 2.0 standards for secure access control.
- **Licensing**: Spring Security is an open-source component under the Apache License 2.0, allowing for free use and modification.
- **Compatibility**: It is compatible with Spring Boot, ensuring a smooth integration into the existing application architecture.
- **Interoperability Standards**: The component adheres to established security protocols, including OAuth 2.0 and JWT (JSON Web Token), ensuring interoperability with various identity providers.

4.8 Interfaces

4.8.1. User Interfaces

The user interface (UI) will be designed to be intuitive and user-friendly, facilitating seamless interactions for various user roles, including guests, customers, staff, managers, and administrators. The key components of the user interface include:

- **Web Application UI**: The primary interface will be a responsive web application developed using ReactJS. The UI will provide:
  - **Sign-Up and Login Pages**: For user authentication.
  - **Dashboard**: Displaying available buildings, rooms, and user-specific information.
  - **Booking Interface**: Enabling users to search, view, and book office spaces.
  - **Payment Interface**: Integrated VNPay payment gateway for processing transactions.
  - **Admin and Manager Dashboards**: For managing users, bookings, and reporting.
- **Accessibility Features**: Compliance with WCAG (Web Content Accessibility Guidelines) to ensure usability for users with disabilities.

4.8.2. Software Interfaces

The POD Booking System will interact with various software components to deliver its functionalities:

- **Payment Processing**: Integration with VNPay via RESTful API, facilitating secure payment transactions.
- **Mapping Service**: Integration with Google Maps API to display building locations and provide navigation support.
- **Authentication and Security**: Utilization of Spring Security for managing user authentication and authorization, including OAuth 2.0 standards.

4.8.3. Communications Interfaces

The application will require several communication interfaces to ensure smooth operation and data exchange:

- **HTTP/HTTPS Protocols**: The web application will utilize HTTP/HTTPS protocols for secure communication between the client and server.
- **RESTful APIs**: The system will expose RESTful APIs for external systems to interact with core functionalities, such as booking and payment processing.
- **WebSocket**: For real-time updates, such as booking status and notifications, a WebSocket interface will be implemented.
- **Local Area Network (LAN)**: If necessary, the application may interface with local network devices for data exchange and user management, ensuring compatibility with existing infrastructure.

5. SUPPORTING INFORMATION

5.1 Contents

    - Product Features
    - User Classes and Characteristics
    - Operating Environment
    - Design and Implementation Constraints
    - Assumptions and Dependencies
- **Specific Requirements**
    - Usability
    - Reliability
    - Performance
    - Supportability
    - Design Constraints
    - On-line User Documentation and Help System Requirements
    - Purchased Components
    - Interfaces
    - Licensing Requirements
    - Legal, Copyright, and Other Notices
    - Applicable Standards
- **Supporting Information**
    - Table of Contents
    - Index
    - Appendices
- **Appendices**
    - Use Case Storyboards
    - User Interface Prototypes
    - Glossary of Terms

5.2 Appendices

- **Appendix A: Use Case Storyboards**
  This appendix contains visual representations of key use cases within the POD Booking System, illustrating user interactions and system responses. These storyboards help stakeholders understand the system's functionality from a user perspective.
- **Appendix B: User Interface Prototypes**
  This appendix includes prototypes of the user interface designed for the POD Booking System. The prototypes provide a preliminary look at the layout, navigation, and user experience. They serve as a foundation for discussions on design and functionality during development.
- **Appendix C: Glossary of Terms**
  This section defines key terms and acronyms used throughout the SRS, ensuring clarity and a shared understanding among stakeholders.