SE Practical

SRS document

APPENDIX B - GROUP LOG

Software Requirement Specification For Hotel Management System

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the Hotel Management System (HMS). The HMS is designed to automate hotel operations such as reservations, check-ins, check-outs, billing, room management, and customer relationship management.

1.2 Scope of the Project

The Hotel Management System (HMS) aims to automate and simplify the daily operations of a hotel through an online platform. It will support three main users: customers, receptionists, and the hotel manager. Customers can check room availability, make reservations, and complete payments online. Receptionists will be able to manage and update booking details, while managers can view financial reports and modify room information such as price and category. The system will include a booking management module, a database server, and a report generator to ensure efficient service delivery. By replacing traditional manual processes, HMS will enable secure transactions, quick data access, error recovery, and improved user experience, ultimately enhancing hotel efficiency and customer satisfaction.

1.3 References

- Software Engineering 9th Edition, Ian Somerville
- Mohapatra, H., & Rath, A. K. (2020). Fundamentals of software engineering: designed to provide an insight into the software engineering concepts. BPB Publications.

1.3 Definitions, Acronyms, and Abbreviations

- HMS: Hotel Management System
- GUI: Graphical User Interface
- API: Application Programming Interface
- DBMS: Database Management System

1.4 Overview

This document includes the overall description of the system in Section 2, covering the product perspective, functions, user characteristics, assumptions, and constraints. Section 3 outlines the specific requirements, including functional and non-functional requirements, external interfaces, system attributes like performance, capacity, availability, safety, and the requirement traceability matrix.

2 OVERALL DESCRIPTION

2.1 Product Perspective

The Hotel Management System is an online platform designed to automate and manage hotel operations efficiently. It enables customers to book rooms, make payments, and check availability. Receptionists can manage bookings, while managers can view reports and update room details. The system aims to streamline hotel processes, reduce manual work, and enhance the overall user experience through secure, fast, and reliable digital services.

2.2 Product Functions

Our Product General functions are:

- Customer Registration
- Check for Availability of Rooms
- Confirmation of Booking
- Payment
- Set Room Details
- Manage Booking Details
- Customer Service

2.3 Constraints

I. Memory: System will have only 10GB space of data server.

II. Language Requirement: Software must be only in English.

III. Budget Constraint: Due to limited budget, HMS is intended to very simple and just for basic functionalities. UI is going to be very simple.

IV. Implementation Constraint: Application should be based on Java only.

2.4 Assumption and Dependencies

It is assumed that a system developed will work perfectly that’s going to be developed under the Windows OS, and Apache Server with MongoDB database. In case of any difficulties, SRS should be flexible enough to change accordingly.

3. SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface for the system shall be compatible with any type of web browser such as Mozilla Firefox, Google Chrome, and Internet Explorer.

3.2 Functional Requirements

3.2.1 Registration

FR1: Customers can register with their details (name, email, password, address, DOB).

FR2: The system sends a verification message to the provided email.

3.2.2 Logging In

FR3: The system verifies the customer's email and password upon login.

FR4: After successful login, the customer is directed to the Home screen.

3.2.3 Reservation

- FR5: The system allows customers to check room availability and view rates.
- FR6: The system allows customers to confirm or cancel bookings.
- FR7: The system records booking details in the database.

3.2.4 Receptionist Access

- FR8: Receptionist can update, add, or delete booking information.
- FR9: The system provides a customer desk portal for receptionist to respond to inquiries.

3.2.5 Manager Access

- FR10: The system generates financial and customer reports for the manager.
- FR11: Manager has full access to modify customer, booking, and room information.

3.2.6 Payment Management System

- FR12: Customers can pay bills online using credit or debit cards.

4. OTHER NON-FUNCTIONAL REQUIREMENTS

4.1 PERFORMANCE REQUIREMENTS

The system must ensure high availability and response times, especially during peak periods. The system should be capable of handling a high volume of simultaneous users, supporting at least 500 concurrent users without degradation in performance.

4.2 SAFETY AND SECURITY REQUIREMENTS

The system must implement secure authentication and authorization mechanisms for each user role (customer, receptionist, manager). Data encryption should be used for sensitive transactions, and all user data must comply with privacy regulations (e.g., GDPR). The system should also have a robust backup and disaster recovery plan to prevent data loss in case of system failure.

4.3 SOFTWARE QUALITY ATTRIBUTES

The system should be scalable, allowing for future expansion (e.g., more hotels or users). It should be reliable, with minimal downtime, and maintainable, supporting easy updates and bug fixes. The software should also be user-friendly, with an intuitive interface for all users, including customers and staff.

5 OTHER REQUIREMENTS

This section includes any additional requirements that don’t fall into the previous categories but are necessary for the successful operation of the system. It may include regulatory compliance, external interfaces, or hardware requirements.

- **Regulatory Compliance:** The system must comply with local data protection laws and industry standards for hotel management software.
- **External Interfaces:** The system should support integration with third-party payment gateways for booking transactions and external reporting tools for financial analysis.

APPENDIXES A - DATA DICTIONARY

Room Table

Field	Type	Description
room_id	INT	Room identifier
room_type	VARCHAR(50)	Type of room
price	DECIMAL(10,2)	Price per night

Booking Table

Field	Type	Description
booking_id	INT	Booking identifier
room_id	INT	Room ID
start_date	DATE	Start date

Customer Table

Field	Type	Description
customer_id	INT	Customer identifier
first_name	VARCHAR(50)	First name
email	VARCHAR(100)	Email

APPENDIX B – GROUP LOG

Week 1: Initial Planning

- **Date:** April 1, 2025
- **Activities:** Defined system scope, roles, and features.
- **Decisions:** Choose tech stack (Angular, Spring Boot, MySQL).
- **Issues:** Payment integration challenges.

Week 2: System Design

- **Date:** April 8, 2025
- **Activities:** Created wireframes, database schema.
- **Decisions:** Finalized tech stack.
- **Issues:** Data security clarification.

Week 3: Development & Testing

- **Date:** April 15, 2025
- **Activities:** Developed booking, login features.

- **Decisions:** Integrated payment gateway.
- **Issues:** Fixed session management bugs.

SCM- Software Configuration Management

```
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
#Write some code
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin <repo-url>
git push -u origin main
#make changes in code
git add .
git commit -m "2 commit"
git push -u origin main

git log
#get commit id from here
git diff <commit-id-1> <commit-id-2>
```

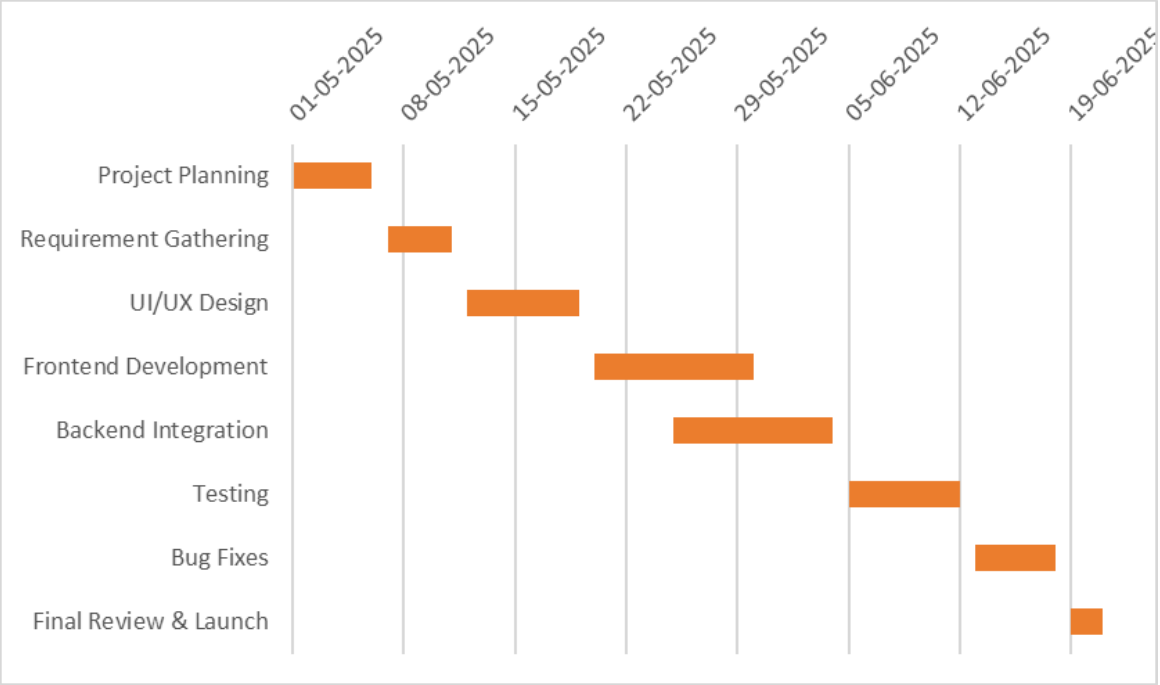
Selenium - Generate Test Scripts

1. Selenium extension in edge/ firefox
2. Record new test case: give name to folder then to the test
3. Record
4. Export the script
5. Take ss in between

Gantt Chart

Task ID	Task Name	Start Date	Duration (days)	End Date
1	Project Planning	01-05-2025	5	06-05-2025
2	Requirement Gathering	07-05-2025	4	11-05-2025
3	UI/UX Design	12-05-2025	7	19-05-2025
4	Frontend Development	20-05-2025	10	30-05-2025
5	Backend Integration	25-05-2025	10	04-06-2025
6	Testing	05-06-2025	7	12-06-2025
7	Bug Fixes	13-06-2025	5	18-06-2025
8	Final Review & Launch	19-06-2025	2	21-06-2025

1. Select task name, start date, duration
2. Insert stack chart
3. No fill for blue on, reverse order vertical axis, tilt the date axis
4. Change start, end duration



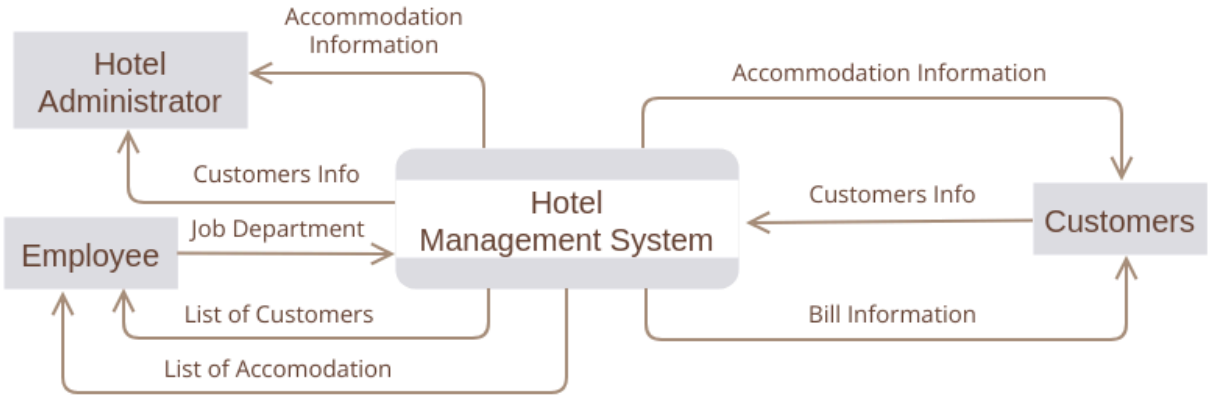
DFD

Level	Processes	Entities	Databases	Data Flow (Examples)
0	Hotel Management System (single process)	Guest, Admin, Receptionist	*Not shown*	Guest → Booking Request → System → Confirmation Admin → Report Request → System → Report Receptionist → Check-In/Out → System
1	1.0 Manage Bookings2.0 Manage Customers3.0 Manage Rooms4.0	Guest, Admin, Receptionist	Booking DB Customer DB Room DB Payment DB	Guest → 1.0 Booking Details → Booking DB Guest → 2.0 Customer Info → Customer DB

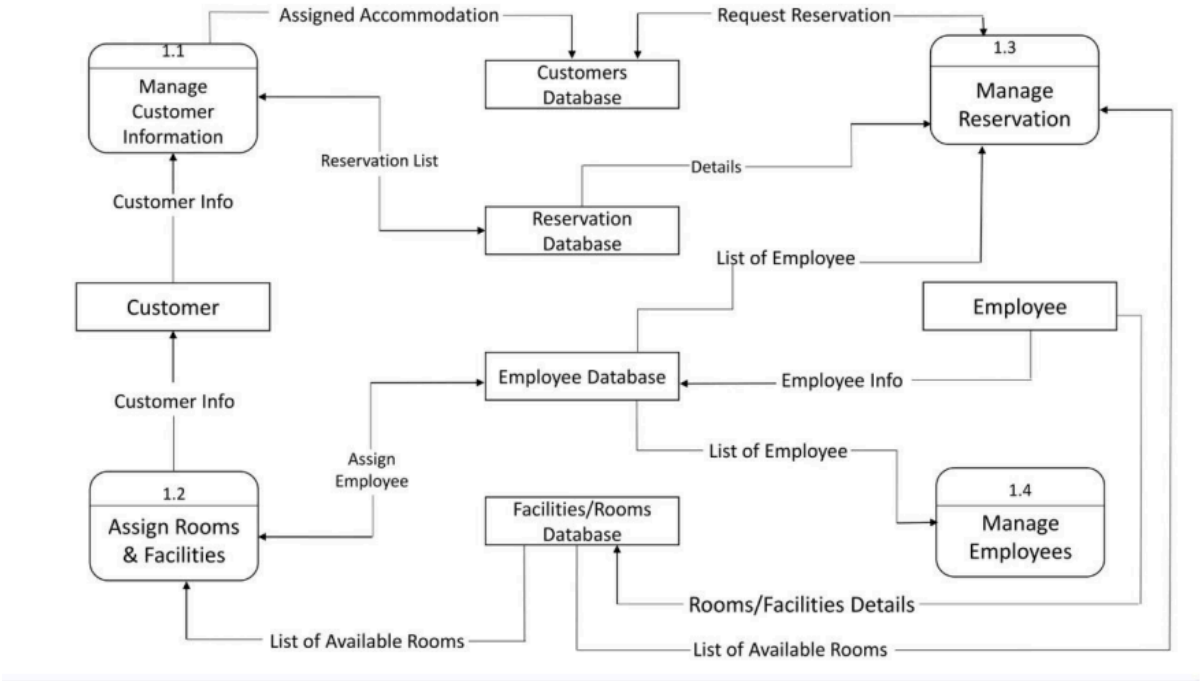
	Process Payments5.0 Generate Reports			Receptionist → 4.0 Payment Info → Payment DB Admin → 5.0 → Report
2	**1.0 Manage Bookings**1.1 Receive Booking Request1.2 Check Room Availability1.3 Confirm & Store Booking1.4 Send Confirmation **2.0 Manage Customers**2.1 Collect Customer Info2.2 Validate Identity2.3 Store Customer Info2.4 Retrieve Customer Profile **3.0 Manage Rooms**3.1 Add Room Info3.2 Update Room Status3.3 Check Room Availability3.4 Assign Room to Guest**4.0 Process Payments**4.1 Receive Payment Details4.2 Verify Payment Method4.3	Guest, Admin, Receptionist	Booking DB Customer DB Room DB Payment DB	Guest → 1.1 → Booking Request1.2 → Room DB Lookup1.3 → Store in Booking DB1.4 → Confirmation to Guest Guest → 2.1 → Info2.2 → Validate ID2.3 → Customer DB Receptionist → 3.2 Update Room3.3 → Query Room DB Receptionist → 4.1 Payment Info4.2 → Verify4.3 → Transaction4.4 → Invoice4.5 → Payment DB

	Process Transaction4.4 Generate Invoice4.5 Store Payment Record			

Level 0:

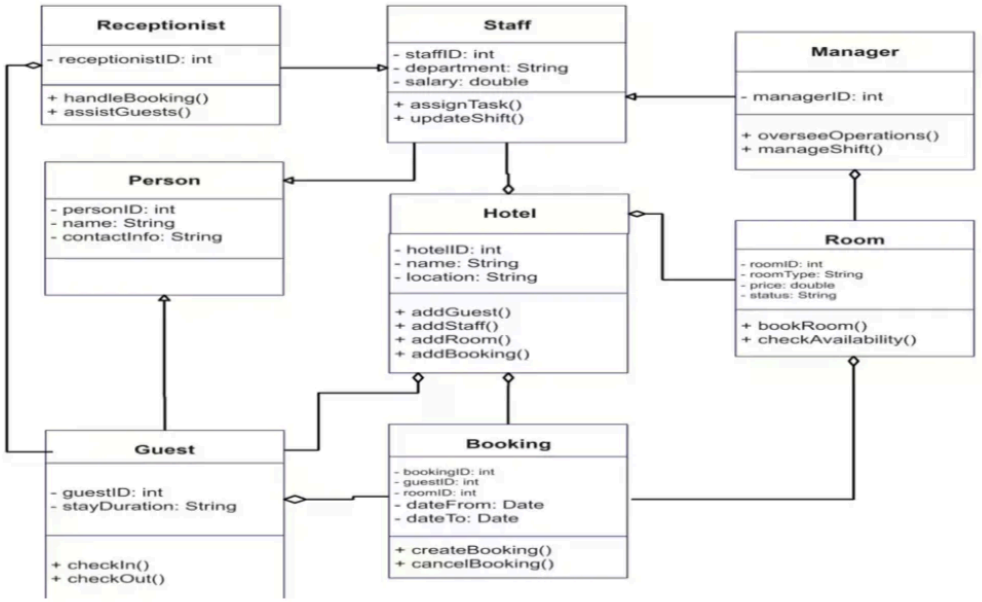


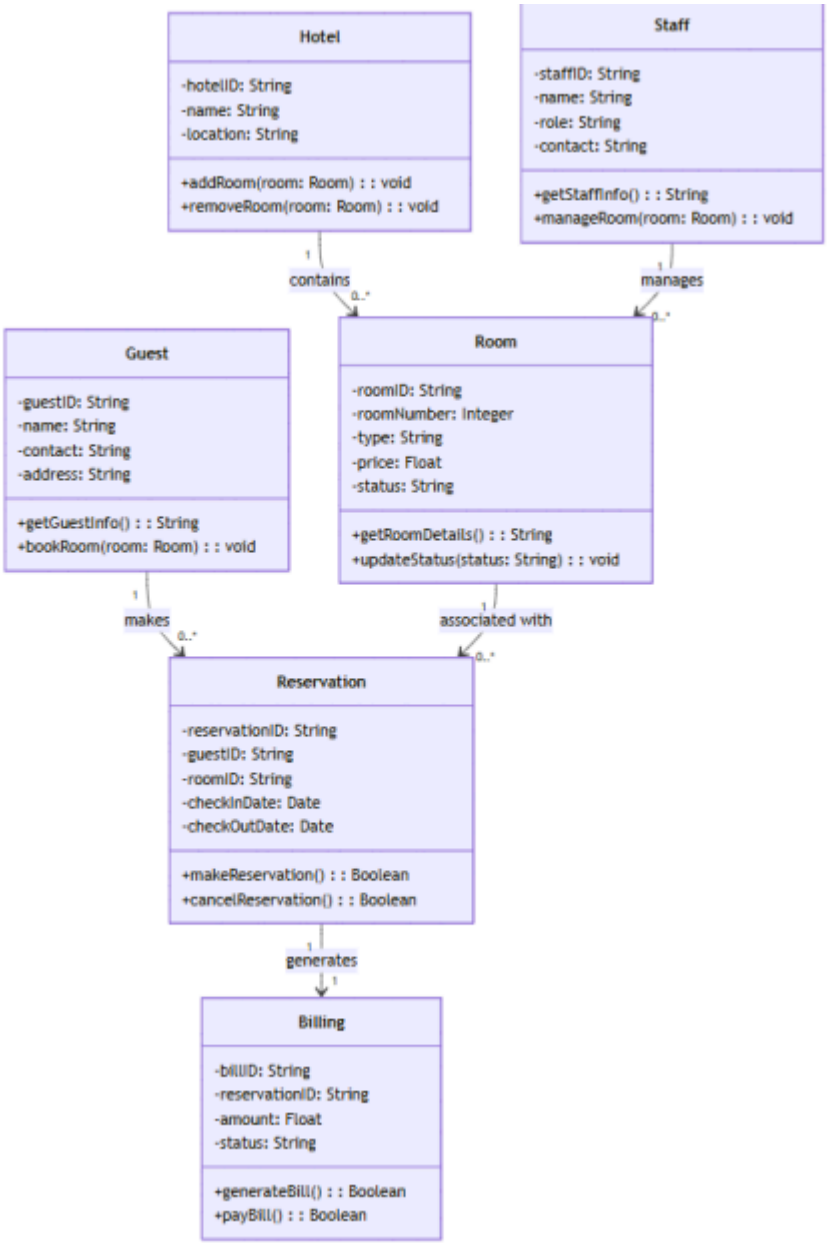
Level 2:



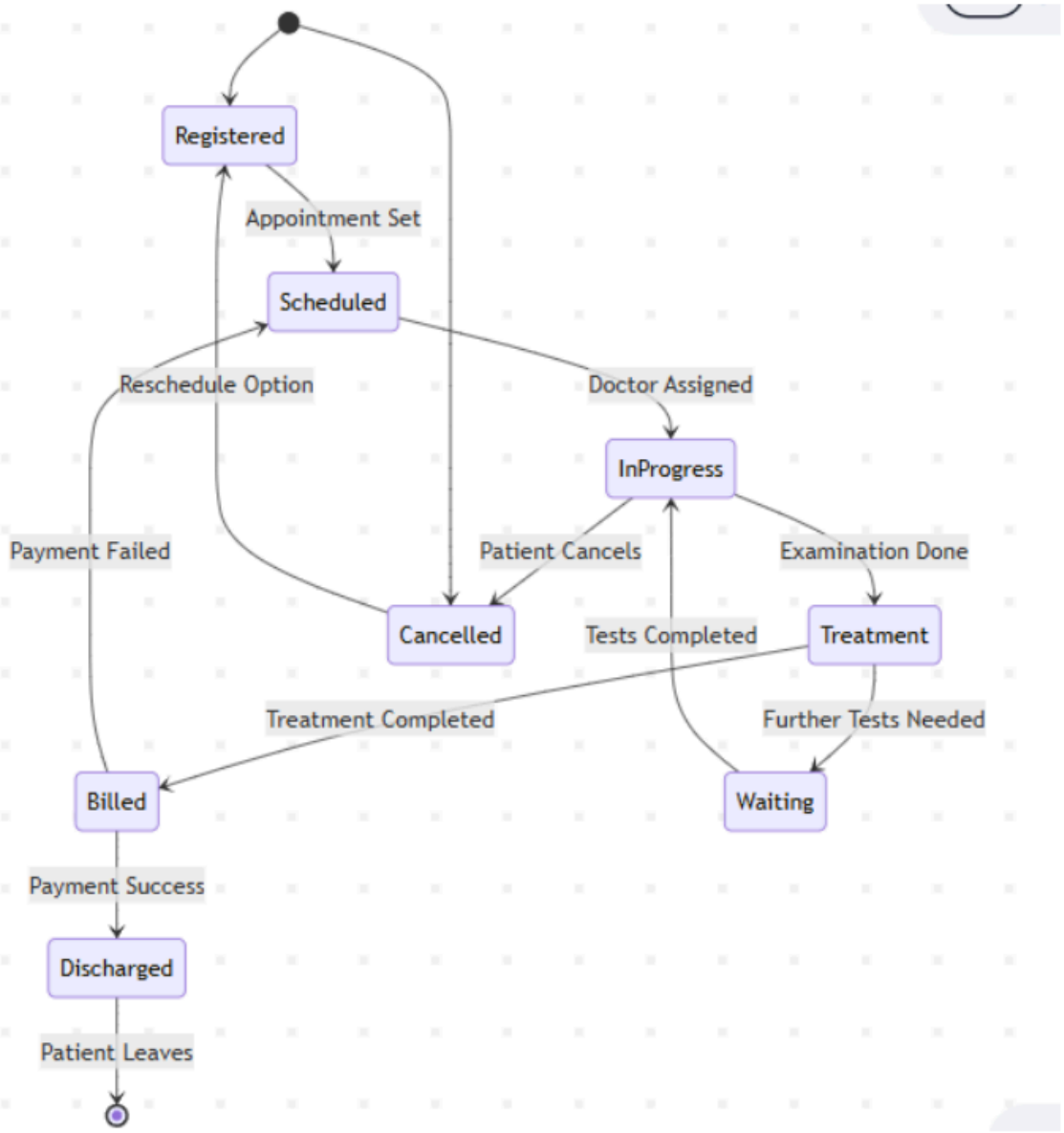
Class

Class	Attributes	Methods	Relationships
Guest	- guestID : int- name : String- email : String- phone : String	- register()- login()- bookRoom()- makePayment()	1 Guest → * Bookings
Room	- roomNumber : int- roomType : String- status : String (Available/Booked)- pricePerNight : double	- checkAvailability()- updateStatus()	1 Room ← * Bookings1 Admin → * Rooms
Booking	- bookingID : int- guestID : int- roomNumber : int- checkInDate : Date- checkOutDate : Date	- createBooking()- cancelBooking()	1 Booking → 1 Guest1 Booking → 1 Room1 Booking → 1 Payment
Payment	- paymentID : int- bookingID : int- amount : double- paymentDate : Date- paymentMode : String	- processPayment()- generateReceipt()	1 Payment ↔ 1 Booking
Admin	- adminID : int- name : String- email : String	- login()- manageRooms()- viewReports()	1 Admin → * Rooms

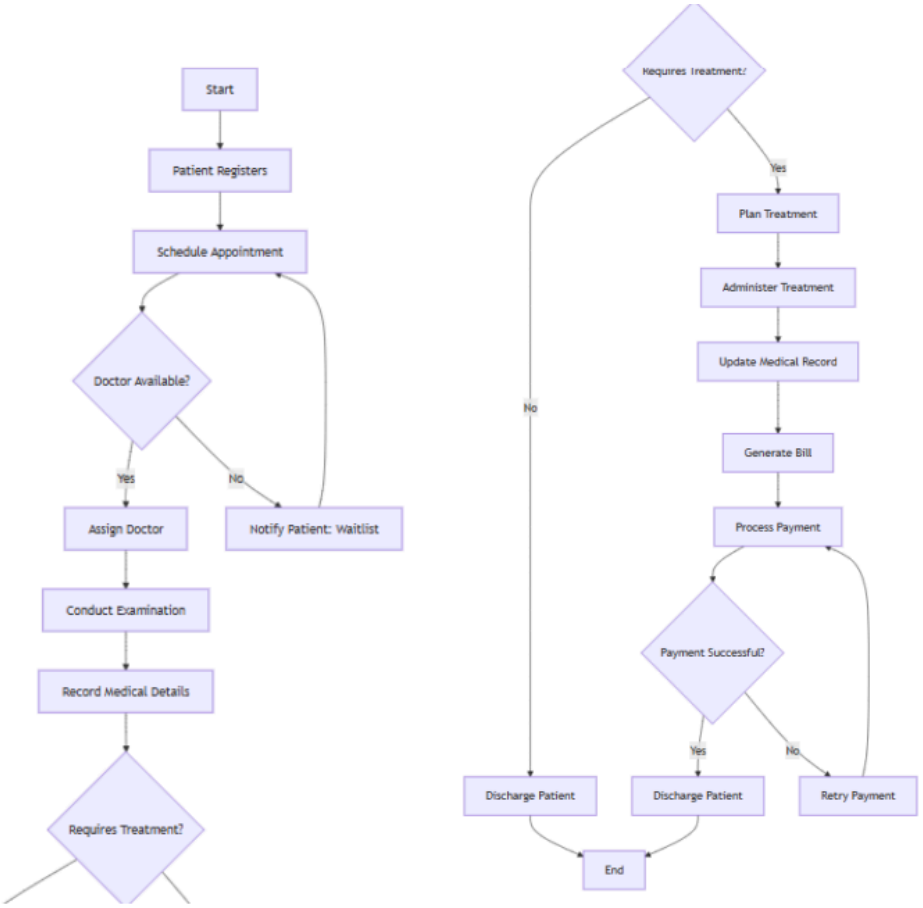




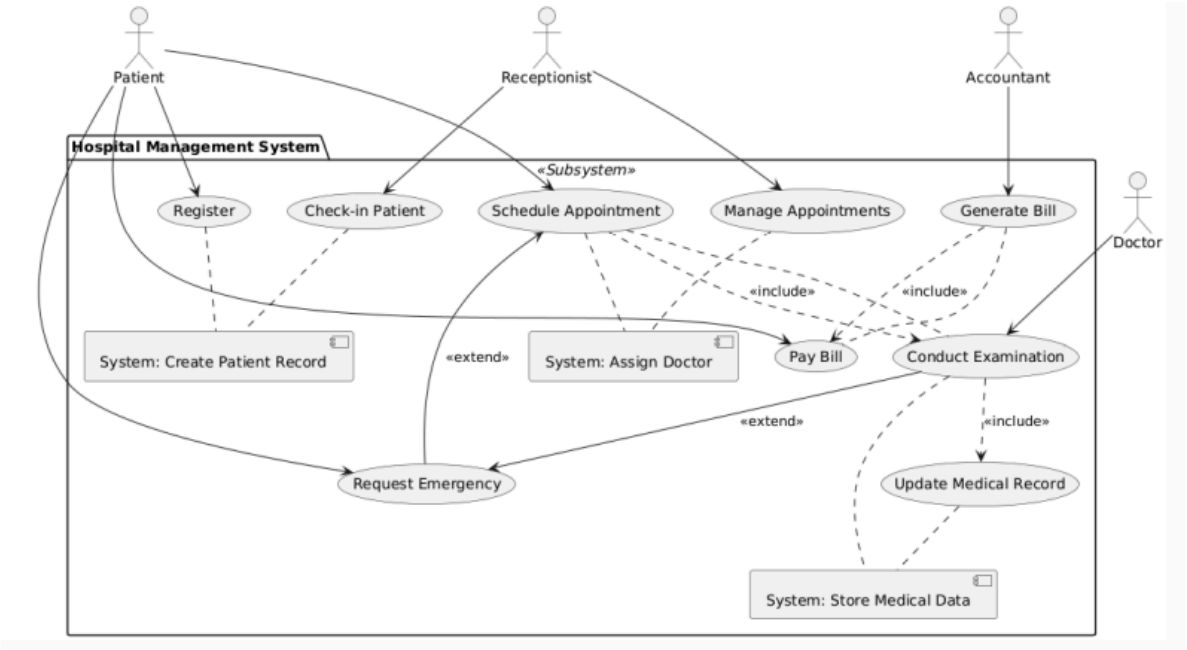
State Transition diagram - Hospital



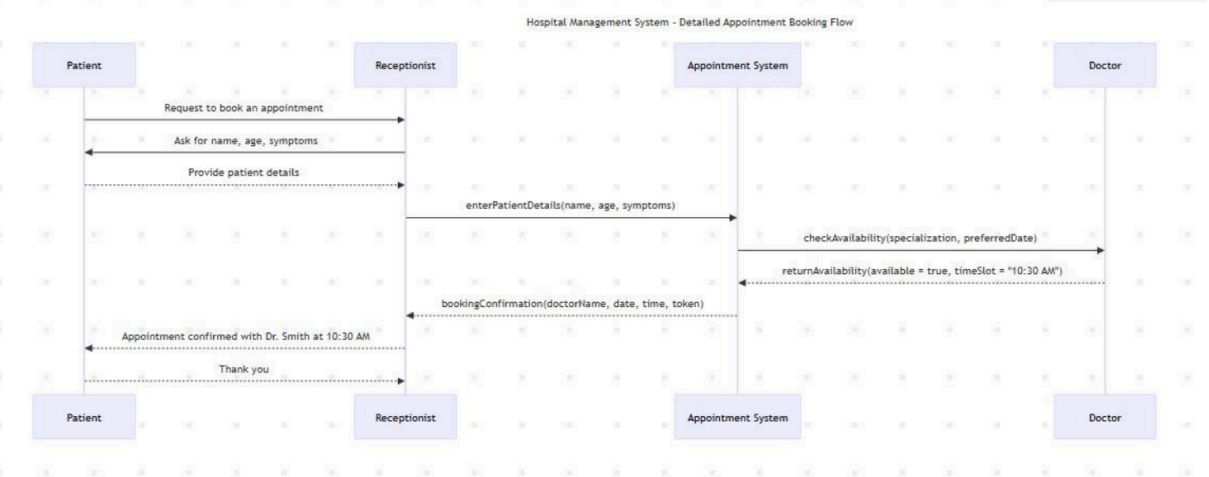
Activity Diagram - Hospital



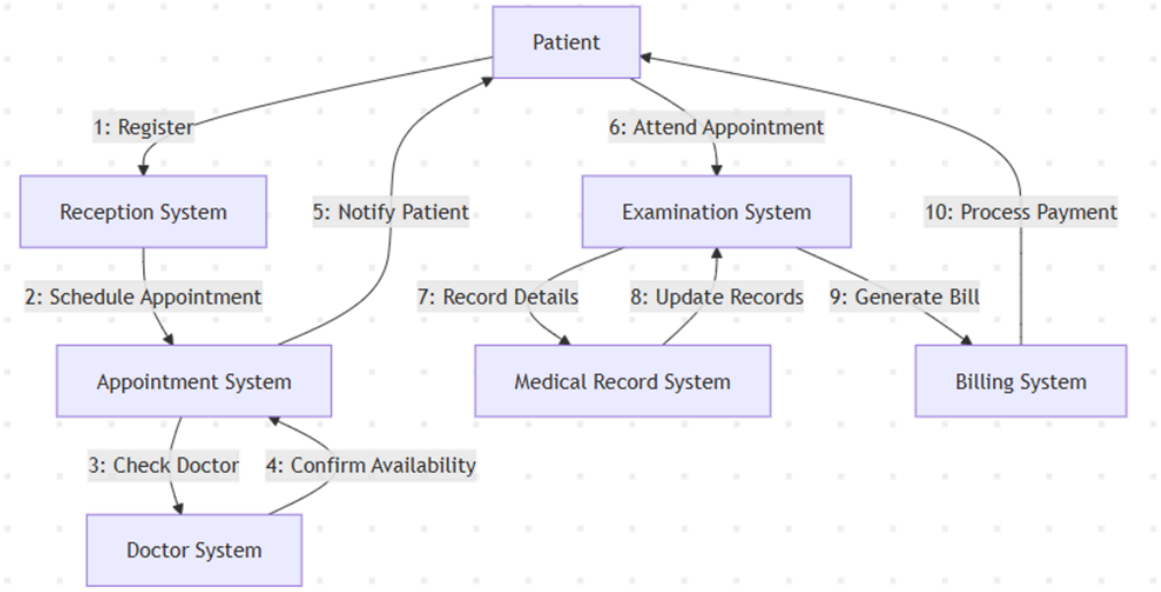
UML USE CASE - Hospital



SEQUENCE DIAGRAM - Hospital



COLLABORATION DIAGRAM - Hospital


Patient
1: Register
6: Attend Appointment
Reception System
5: Notify Patient
Examination System
10: Process Payment
2: Schedule Appointment
7: Record Details
8: Update Records
9: Generate Bill
Appointment System
Medical Record System
Billing System
3: Check Doctor
4: Confirm Availability
Doctor System