* **System Architecture:** Describe the overall architecture of the Hospital Management System, such as client-server, three-tier, or microservices.
* **Database Architecture:** Explain the structure and design of the database, including tables, relationships, and data storage.
* **Security Architecture:** Detail the security measures in place, including authentication, authorization, and encryption.
* **Scalability and Performance:** Discuss how the system handles scalability and performance requirements.

**SRS DOCUMENT**

TOPIC

**HOSPITAL MANAGEMENT SYSTEM**

SUKRITI VIMAL

### Introduction

#### This introduction sets the stage for the Hospital Management System (HMS) SRS document, providing an overview of its purpose, scope, and intended audience. The subsequent sections of the SRS document will delve into greater detail, describing each requirement and specification for the HMS project.

#### a. Purpose of SRS Document

The purpose of this document is to provide a debriefed view of requirements and specifications of the project called "*Hostel Management System*".

The aim of this system is to carry out the activities of Hostel in an efficient way. It will provide faster access to data of Hostel and allow addition, upgradation, modification, and deletion of data in a systematic and reliable manner.

This document discusses about whole system from backend to user interactions.

#### b. Objective of the application

The Hospital Management System (HMS) is designed to encompass a wide range of functionalities and modules that cater to the needs of healthcare facilities, including hospitals, clinics, and medical centres. The system aims to address the following key areas:

* **Patient Management**: Registering and managing patient information, including appointments, medical history, and billing.
* **Appointment Scheduling**: Facilitating the scheduling of patient appointments with healthcare providers.
* **Medical Records**: Maintaining electronic health records (EHR) for patients, including diagnosis, treatment plans, and prescriptions.
* **Billing and Invoicing**: Managing financial aspects such as billing, insurance claims, and payment tracking.
* **Inventory Management**: Monitoring and controlling medical supplies, equipment, and pharmaceuticals.
* **Staff Management**: Handling the details of doctors, nurses, and administrative staff, including scheduling and payroll.
* **Reporting and Analytics**: Generating reports and analytics to aid decision-making and performance evaluation.
* **Security and Access Control**: Ensuring data security and defining user roles and permissions.

#### c. Scope of the Application

The Hospital Management System (HMS) is designed to encompass a wide range of functionalities and modules that cater to the needs of healthcare facilities, including hospitals, clinics, and medical centers. The system aims to address the following key areas:

* **Patient Management**: Registering and managing patient information, including appointments, medical history, and billing.
* **Appointment Scheduling**: Facilitating the scheduling of patient appointments with healthcare providers.
* **Medical Records**: Maintaining electronic health records (EHR) for patients, including diagnosis, treatment plans, and prescriptions.
* **Billing and Invoicing**: Managing financial aspects such as billing, insurance claims, and payment tracking.
* **Inventory Management**: Monitoring and controlling medical supplies, equipment, and pharmaceuticals.
* **Staff Management**: Handling the details of doctors, nurses, and administrative staff, including scheduling and payroll.
* **Reporting and Analytics**: Generating reports and analytics to aid decision-making and performance evaluation.
* **Security and Access Control**: Ensuring data security and defining user roles and permissions.

The software of Online Hostel Management System is designed to meet the following needs and provide the following benefits:

* Less human error
* Strength and strain of manual labor can be reduced
* Data redundancy can be avoided to some extent
* Data consistency
* It secures the data of students
* Easy to handle
* Easy data updating
* Easy record keeping
* Backup data can be easily generated

### Overall Description

#### a. Application Perspective

The Hostel Management System is a web application. It is a replacement of traditional hostel management system whose process takes enough time. With this application user can easily book their rooms in hostels. This web application has two interfaces administration and user interface.

#### b. Application Features

Hostel Management System has two modules: Admin and User

### Admin Panel

#### 1.Admin Login

Admin can login through login form.

**2. Admin Profile**

Admin can manage his own profile. Admin can also change his password

#### 3.Courses

Admin can create add course, edit courses and also delete the course.

1. **Rooms**

Admin can create rooms and allots seater to particular rooms and assign the fees.

1. **Registration**

Admin can create student profile and allot the rooms

##### 6. Manage the Registration

Admin can manage the all the student Profile. Take a print out of all profiles and also delete the profile.

**7. Forgot Password**

Admin can also retrieve the password if admin forgot the password.

### User Panel

1. **User Registration----** User can register through user registration form.
2. **User Login--** User can login through login form
3. **Forgot Password—**User can retrieve password through forgot password link

#### User Dashboard

1. **User Profile—**User can manage own profile
2. **Book Hostel** – User can book hostel
3. **Room Details-** Booked Room Details
4. **Change Password-** User Can change own password
5. **User access log-** User can watch last login details

##### c. User Classes and Characteristics

**Admin:**

They are responsible for maintaining the application. They have the access to manage :

Courses (add, update and delete)

Rooms (add, update and delete)

Student registration details

Manage students (view and delete)

Check user access logs

Admin Profile update and change passwords

**Users:**

Registration

Login

Password Recovery

Book hostels

Booked hostel details

User Profile

User access logs

##### d. Operating Environment

Windows 7 and newer version

##### e. Design and Implementation Constraints

Database Normalization: **Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divide larger tables into smaller tables and links them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

SQL constraints: SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

#### f. Assumptions and Dependencies

**Assumptions:**

1. **Hardware Availability:** Assume that the necessary hardware infrastructure, such as servers, computers, and networking equipment, is already in place or will be procured as needed.
2. **Software Dependencies:** Assume that any required third-party software components or libraries, such as operating systems, databases, or middleware, are available and compatible with the system.
3. **Data Availability:** Assume that historical patient data, including medical records, billing information, and other relevant data, will be accessible and can be migrated to the new system.
4. **Regulatory Compliance:** Assume that the system will comply with all relevant healthcare regulations and standards, such as HIPAA (Health Insurance Portability and Accountability Act) in the United States or similar regulations in other countries.
5. **User Availability:** Assume that the necessary hospital staff, including administrators, doctors, nurses, and other personnel, will be trained and available to use the system.
6. **Internet Connectivity:** Assume that the system will have access to a stable internet connection if cloud-based features or remote access are part of the system's design.
7. **Budget and Resources:** Assume that the project has the necessary budget and resources allocated for development, testing, and ongoing maintenance.
8. **Security Assumptions:** Specify assumptions about the security measures in place, such as firewall configurations and antivirus software, to protect the system and its data.

**Dependencies:**

1. **Database Systems:** Identify dependencies on specific database systems, such as MySQL, PostgreSQL, or Oracle, and ensure that they are available and properly configured.
2. **Operating Systems:** Specify the operating systems on which the system will run, such as Windows Server, Linux, or others, and ensure they are compatible.
3. **Programming Languages and Frameworks:** Identify dependencies on programming languages (e.g., Java, Python) and frameworks (e.g., Django, Ruby on Rails) and ensure that the development team is proficient in them.
4. **External Services:** If the system relies on external services like payment gateways or lab integrations, specify these dependencies and their availability.
5. **Authentication Services:** Specify any third-party authentication or identity management services that the system may use for user authentication.
6. **Data Migration Tools:** If data migration is required from legacy systems, specify any tools or scripts needed for this process.
7. **Compliance Standards:** Identify any dependencies on external compliance standards, such as industry-specific regulations or government guidelines.
8. **Training and Documentation:** Ensure that dependencies on training materials and documentation for end-users and administrators are met.
9. **Testing Environments:** Specify the required testing environments, including development, staging, and production, and ensure they are set up correctly.
10. **Vendor Support:** If the system relies on vendor-specific products or services, specify dependencies on vendor support and ensure that vendor relationships are established.

#### System Features

1. **Functional Requirements**

#### 1.Admin Login

Admin can login through login form.

**2. Admin Profile**

Admin can manage his own profile. Admin can also change his password

#### 3.Courses

Admin can create add course, edit courses and also delete the course

1. **Rooms**

Admin can create rooms and allots seater to particular rooms and assign the fees.

1. **Registration**

Admin can create student profile and allot the rooms

##### 6. Manage the Registration

Admin can manage the all the student Profile. Take a print out of all profiles and also delete the profile.

##### 7. Forgot Password

Admin can also retrieve the password if admin forgot the password.

**User Panel:**

1. **User Registration----** User can register through user registration form

1. **User Login--** User can login through login form

1. **Forgot Password—**user can retrieve password through forgot password link

##### 4. User Dashboard

1. **User Profile—**User can manage own profile

1. **Book Hostel** – User can book hostel

1. **Room Details-** Booked Room Details

1. **Change Password-** User Can change own password

1. **User access log-** User can watch last login details

1. **System Specifications**
2. **Hardware**

* **Server Hardware:** Detail the specifications of the server(s) required for hosting the system, including CPU, RAM, storage, and network requirements.
* **Client Hardware:** Describe the hardware requirements for the client devices, such as PCs, tablets, or smartphones.
* **Networking:** Specify the network infrastructure requirements, including bandwidth, routers, switches, and any other networking components.

1. **Software**

* **Operating System:** Identify the server and client operating systems required (e.g., Windows Server, Windows 10, Linux).
* **Database Management System (DBMS):** Specify the DBMS used (e.g., MySQL, PostgreSQL).
* **Web Server:** If applicable, mention the web server software (e.g., Apache, Nginx).
* **Programming Languages:** List the programming languages and frameworks used (e.g., Java, .NET, Python, Django).
* **Third-Party Libraries:** Include any third-party libraries or APIs that the system relies on.
* Browser Compatibility: State which web browsers are supported by the user interface.

1. **Architecture of System**

* **System Architecture:** Describe the overall architecture of the Hospital Management System, such as client-server, three-tier, or microservices.
* **Database Architecture:** Explain the structure and design of the database, including tables, relationships, and data storage.
* **Security Architecture:** Detail the security measures in place, including authentication, authorization, and encryption.
* **Scalability and Performance:** Discuss how the system handles scalability and performance requirements.

##### c. Non-functional Requirements

The non-functional requirements of the system are explained below:

**i. Performance Requirements**:

* **Quickness: -** System should be fast enough to respond to any of the user action in any way without any shattering or buffering else it will be not be a good experience.
* **Robustness:** - System should be robust to deal and act accordingly with common error scenarios and unsupported file types.
* **Failure Handling**: -In case of failures it should be able to handle failures and recover quickly.

**ii. Security Requirements:**

Any system developed should be secured and protected against possible hazards. Security measures are provided to prevent unauthorized access of the database at various levels. An uninterrupted power supply should be so that the power failure or voltage fluctuations will not erase the data in the files.

Password protection and simple procedures to prevent the unauthorized access are provided to the users. The system allows the user to enter the system only through proper user name and password.

1. **Safety Requirements:**

Hosting PHP applications on a server requires a careful and constant attention to deal with these security risks. There are advanced protection patches such as Suhosin and Hardening-Patch, especially designed for web hosting environments.

###### iv. Software Quality Attributes:

* **Memory Management:** - System should not leak memory.
* **Compatibility:** - System should peacefully co-exist with other software.
* **Error Handling**: - System should not cause or trigger any events that will leave Operating System in unrecoverable state.