HOSPITAL MANAGEMENT SYSTEM

1. PARMAR JAI ATUL	-	22BD1A0517
2. P DEVI SAINATH	-	22BD1A051A
3. H S VISHAL	-	22BD1A050E
4. N SAI GARGEY	-	22BD1A0512
5. S SAI BHARATH	-	22BD1A051L

Problem Statement:

Traditional hospital management systems that rely on physical records introduce significant inefficiencies, such as time delays, paper wastage, and a higher risk of errors in managing patient information. Retrieving and updating paper-based records—like medical histories, prescriptions, and test reports—is cumbersome and often leads to treatment delays or misdiagnosis due to slow access to critical data. A digital hospital management system addresses these issues by securely centralizing patient records, accessible in real-time across multiple hospitals using a unique identifier like Aadhaar. This eliminates the need for redundant paperwork, allowing for faster decision-making and more accurate treatments. Automated updates ensure doctors have the most current patient information, reducing risks of outdated data leading to treatment errors. The system also streamlines administrative tasks, freeing up hospital staff to focus on patient care while providing a user-friendly interface for both hospital authorities and patients to manage and access medical data. Patients can easily view their medical history, fostering better communication and engagement in their healthcare. Furthermore, real-time synchronization across hospitals ensures that critical information is available when needed, improving treatment speed and reducing potential fatality risks. This digital solution enhances overall efficiency, reduces costs, and safeguards patient privacy through secure data management, making it a vital upgrade from traditional paper-based systems.

Software Requirement Specification

For

HealthNet

Version 0.3 approved

Prepared by:

1.	Vishal HS	- 22BD1A050E
2.	Sai Gargey Nakka	- 22BD1A0512
3.	Parmar Jai Atul	- 22BD1A0517
4.	P Devi Sainath	- 22BD1A051A
5.	S Sai Bharath	- 22BD1A051L

Keshav Memorial Institute of Technology

10-09-2024.

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
Week - 1	10-09-2024	Creation of SRS document (Introduction)	0.1
Week - 2	21-09-20024	Updation of SRS document (Overall Descriptions and External Interface Requirements)	0.2
Week - 3	23-09-20024	Updation of SRS document (System Features and Other Non-Functional Requirements)	0.3

1.Introduction

The goal of HealthNet is to provide an efficient, secure, and user-friendly digital platform to manage patient records, ensuring that medical history, prescriptions, doctor visit notes, test reports, and other vital health information are accessible and synchronized in real-time across hospitals in the network.

The system is intended to replace traditional paper-based record management, ensuring accuracy, privacy, and ease of access to patient data.

1.1 Purpose:

Our Hospital Management System project aims to offer a fully digital solution for the effective management of patient records, including test reports, medications, and medical visit records. Patient data is securely synced across all institutions using a unique identity, like an Aadhaar number, guaranteeing immediate access to correct medical information.

The inefficiencies of paper-based documentation are eliminated by this system, which also speeds up record retrieval and changes, enabling more rapid treatments and better patient care overall.

1.2 Document Convention:

Heading:

- Font-Size:16
- Font-Style: Bold
- Font: Times New Roman

Subheading:

- Font-Size:14
- Font-Style: Bold
- Font: Times New Roman

Content:

- Font-Size:12
- Font: Times New Roman.

1.3 Intended Audience and Reading Suggestions:

This document is a HealthNet will serve a variety of purposes for different audiences. Similar to having developers plan and carry out the project. Managers for keeping expenses and time in check. For creating the most effective advertisement regarding the characteristics and how it differs from other online retailers. Users are able to make any necessary changes and choose whether or not the program meets their needs. This document can be used by testers to conduct tests. For developers, both functional and non-functional features are immensely beneficial.

1.4 Product Scope:

The goal of the Hospital Management System is to offer a completely digitally integrated system for handling every facet of hospital operations and patient healthcare information. The HMS's scope includes a number of features that increase productivity, accuracy, and data accessibility. These features help hospitals run more efficiently while still providing higher-quality patient care. The system will guarantee that healthcare providers always have access to correct and current patient data by enabling safe storage, real-time access, and synchronization of patient data across various hospitals or clinics.

1.5 References:

https://www.researchgate.net/publication/367460409_The_Hospital_Management_System

https://www.researchgate.net/publication/375497520_MODERN_HOSPITAL_MANAGEMENT_SYSTEM

https://www.academia.edu/8831452/Software_Requirements_Specification_for_FSoft_D

2. Overall Description

2.1 Product Perspective

The Hospital Management System (HMS) is a digital platform designed to centralize patient records and streamline hospital operations, replacing inefficient paper-based systems. It supports real-time synchronization of medical data across multiple hospitals, ensuring secure access through a unique patient identifier like Aadhaar. Key features include patient record management, appointment scheduling, billing, insurance claims, and staff management. The system is accessible via web and mobile interfaces, catering to patients, doctors, and administrators. It integrates with diagnostic tools, government health databases, and insurance systems while complying with privacy regulations such as HIPAA. Cloud-based storage ensures data is available for authorized personnel, enhancing decision-making and patient care. The modular design allows hospitals to implement features as needed, ensuring scalability.

2.2 Product Functions

2.2.1 Administrator

- Administrators should be able to add, modify, or remove user access for hospital staff (e.g., doctors, nurses, administrative personnel) based on their roles and responsibilities.
- Admins can insert, modify, and delete patient records as necessary, ensuring that data privacy policies are maintained.
- Admins can add, edit, and organize hospital departments, medical services, and personnel assignments to ensure efficient hospital operations.
- Admins can approve or reject requests from external vendors for medical supplies or services, ensuring they meet hospital standards and regulatory requirements.
- Admins can send notifications to staff and patients regarding important updates, such as system maintenance schedules, hospital policy changes, or emergency procedures.
- Admins can track, record, and manage inventory, such as medical supplies, medications, and equipment. They can also track returned items or defective products in the hospital's inventory.

2.2.2 Customers/Users

- Patients must be able to view their medical history, prescriptions, diagnostic reports, and other health-related records in real-time. They should also be notified of new test results, appointments, or updates related to their treatments.
- Patients can review and update their personal account details, including contact information, medical insurance, and emergency contacts, to ensure up-to-date information.
- Patients should have the ability to search for doctors, medical departments, or services by specialty, location, or other relevant criteria (e.g., availability of specific treatments or tests).
- Patients must be able to request changes to their appointments, reschedule, or cancel them based on hospital policies. They should also be able to raise requests for clarification regarding treatments or test results.
- Patients should have the provision to review and rate doctors, hospital services, and treatments. Additionally, they can suggest improvements or additional services that can be incorporated into the hospital's system.
- Patients should have the option to return or modify prescriptions or health services they no longer need, based on hospital policies, such as canceling an upcoming test or returning prescribed medication if no longer required.

2.3 User Classes and Characteristics

Administrators:

- Highly proficient in hospital administration and management.
- Manage user permissions, medical services, and vendor approvals.

Doctors:

- Focus on patient care and diagnosis.
- Require quick access to patient medical histories.

Patients:

- Access their medical history and communicate with healthcare providers.
- Expect ease of use and secure access.

Support Staff:

- Handle administrative and non-clinical tasks.
- Operate the system daily to manage hospital workflows.

2.4 Operating Environment

- Web and mobile-friendly platform compatible with major browsers (Chrome, Firefox, Safari).
- Operating systems: Windows, MacOS, Linux, Android, iOS.
- Processor: Minimum Intel Pentium 4 or equivalent.
- RAM: 2 GB or above.
- Internet connectivity is required for real-time synchronization.

2.5 Design and Implementation Constraints

- Must comply with medical data privacy laws (e.g., HIPAA).
- High security requirements to protect sensitive patient information.
- Dependence on reliable internet connections for data synchronization.
- The system must support multiple languages and be adaptable to various regulatory standards.

2.6 User Documentation

- Comprehensive user manual for administrators, doctors, and patients.
- Online help system embedded within the platform.
- Tutorials and FAQs accessible via the user dashboard.

2.7 Assumptions and Dependencies

- Hospitals using this system must have reliable internet access.
- The system assumes hospitals will adopt the Aadhaar-based unique identifier for patients.
- Dependence on third-party vendors for medical supplies and services.

3. External Interface Requirements

3.1 User Interfaces

- A user-friendly web-based interface for doctors, patients, and administrators.
- Mobile app interfaces for real-time access.
- Common actions include viewing medical records, updating details, and scheduling appointments.

3.2 Hardware Interfaces

- Interacts with barcode scanners, medical devices, and point-of-sale (POS) systems for inventory management.
- Supports interaction with hospital management hardware, such as patient monitoring systems.

3.3 Software Interfaces

- Integration with hospital databases, medical diagnostic software, and government health databases.
- Must be compatible with cloud-based storage systems for real-time data synchronization.

3.4 Communications Interfaces

- Supports HTTP/HTTPS protocols for web communication.
- Secure encryption for all data transfers between hospitals, patients, and cloud systems.

4. System Features

4.1 Feature: Patient Record Management

4.1.1 Description and Priority:

- This feature enables the creation, update, and secure storage of patient medical records, including histories, prescriptions, and diagnostic reports. It allows authorized personnel to access the data and make real-time updates across multiple hospital locations.
- Priority**:** High

4.1.2 Stimulus/Response Sequences:

- A doctor or nurse requests patient data → The system retrieves the relevant records.
- A lab test result is entered → The system updates the patient record and notifies the assigned physician.

4.1.3 Functional Requirements:

- REQ-001: The system must store all patient records in a secure cloud database.
- REQ-002: The system must allow authorized users to view and update patient records in real time.
- REQ-003: The system must log every access and modification of patient records for audit purposes.

4.2 Feature: Appointment Scheduling and Management

4.2.1 Description and Priority:

- This feature allows patients and administrative staff to manage appointments. Patients can book, reschedule, or cancel appointments with healthcare professionals. It integrates with doctors' availability and sends automated reminders.
- Priority**:** Medium

4.2.2 Stimulus/Response Sequences:

- A patient requests an appointment → The system checks doctor availability and schedules the appointment.
- A patient reschedules an appointment → The system updates the doctor's calendar and notifies all parties involved.

4.2.3 Functional Requirements:

- REQ-004: The system must allow patients to book and reschedule appointments through both web and mobile interfaces.
- REQ-005: The system must send reminders to patients via email/SMS before scheduled appointments.
- REQ-006: The system must provide doctors with real-time access to their appointment schedule.

4.3 Feature: Billing and Invoicing

4.3.1 Description and Priority:

- This feature automates the billing process, including generating invoices for treatments, prescriptions, and lab tests. It supports insurance claims, allowing patients and administrators to manage billing efficiently.
- Priority**:** High

4.3.2 Stimulus/Response Sequences:

- A treatment is completed → The system generates an invoice for the patient and sends it via email.
- A patient submits insurance information → The system processes the claim and updates the outstanding balance.

4.3.3 Functional Requirements:

- REQ-007: The system must generate invoices automatically based on the services rendered.
- REQ-008: The system must support insurance claims and provide a breakdown of patient and insurance payments.
- REQ-009: The system must allow patients to view and pay their bills through a secure payment gateway.

4.4 Feature: Doctor and Staff Management

4.4.1 Description and Priority:

- This feature enables hospital administrators to manage staff schedules, roles, and permissions. It ensures that staff members have access to only the information they need to perform their duties.
- Priority**:** High

4.4.2 Stimulus/Response Sequences:

- A doctor is assigned to a new department → The system updates their access permissions accordingly.
- A nurse requests time off → The system checks availability and approves or denies the request based on hospital needs.

4.4.3 Functional Requirements:

- REQ-010: The system must allow administrators to assign roles and permissions to staff members.
- REQ-011: The system must track staff schedules and handle time-off requests.
- REQ-012: The system must restrict access to sensitive information based on user roles.

5. Other Non-functional Requirements

5.1 Performance Requirements

- The system must support up to 10,000 simultaneous users with minimal delay.
- Retrieval of patient records should not take more than 5 seconds.
- The system should handle at least 1,000 concurrent appointments and updates without slowing down.

5.2 Safety Requirements

- The system must include fail-safes for accidental deletion or loss of patient data.
- Backup and recovery mechanisms should be in place to prevent data loss in case of a system crash or hardware failure.

5.3 Security Requirements

- All sensitive patient and staff information must be encrypted during transmission and storage.
- Two-factor authentication should be implemented for access to sensitive features like patient records and billing.
- Access control should be role-based, ensuring that only authorized personnel can access sensitive information.

5.4 Software Quality Attributes

- **Availability:** The system must have 99.9% uptime to ensure continuous access to patient records.
- **Maintainability:** The system should be designed with modular architecture to allow easy updates and bug fixes.
- **Usability:** The system should be intuitive and require minimal training for hospital staff and patients to use.

5.5 Business Rules

- Only licensed medical professionals (e.g., doctors, nurses) are authorized to make changes to patient records.
- Patients must verify their identity using secure means (e.g., Aadhaar or biometric verification) to access personal medical data.
- The system must adhere to regional data protection regulations (e.g., HIPAA in the U.S., GDPR in Europe).

Software Development Life Cycle (SDLC)

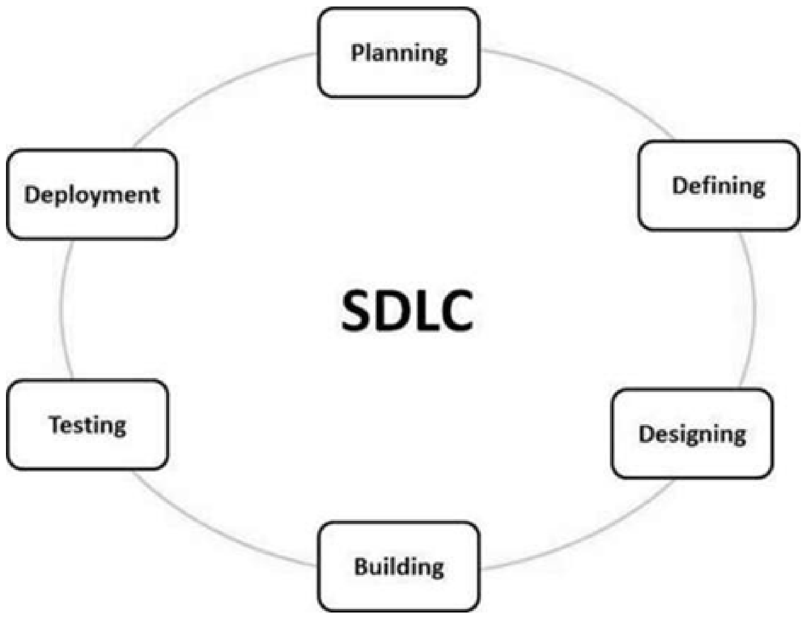
Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called the Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

What is SDLC?

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



A typical Software Development Life Cycle consists of the following stages

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters such as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

SDLC Models

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred to as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry −

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

SRS Document

1.Introduction

1.1 Purpose of Document

Provide an introductory paragraph explaining the purpose of this document. Its purpose is to explicitly cite all functions that the project shall do. This document is the primary document, upon which the design, source code, and test plan all base their content. This document is used to determine if the final delivered product provides everything that it was supposed to. The Client, User, and Software Engineering representatives often negotiate the content of this document.

1.2 Scope

Provide two paragraphs, the first describing the scope of the product, with the second describing the scope of this document. Remember that "scope" basically means the extent of activity or influence, or range of operation. Be sure that the two paragraphs in this section distinguish between the scope of the product, versus the scope of this document.

You will probably find that in most of the Software Engineering documents that you create in this course, the paragraph for scope of product will be identical (as expected). Specifically for this document, the scope includes all team members and their responsibilities for specifying the product's requirements.

1.3 Objective

A project objective describes the desired results of a project, which often includes a tangible item. An objective is specific and measurable, and must meet time, budget, and quality constraints. ... A project may have one objective, many parallel objectives, or several objectives that must be achieved sequentially.

1.4 Proposed System

The proposed system should have the following features. The transactions should take place in a secured format between various clients in the network. It provides flexibility to the user to transfer the data through the network very easily by compressing the large amount of file.

2. Requirements Specifications

2.1 Functional Requirements

functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs.

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases. Functional requirements are supported by non-functional requirements (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>."The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.

2.2 Non-Functional Requirements

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. Also known as system qualities, nonfunctional requirements are just as critical as functional Epics, Capabilities, Features, and Stories. They ensure the usability and effectiveness of the entire system. Failing to meet any one of them can result in systems that fail to satisfy internal business, user, or market needs, or that do not fulfill mandatory requirements imposed by regulatory or standards agencies. In some cases, non-compliance can cause significant legal issues (privacy, security, safety, to name a few).

2.3 Software Requirements

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

2.4 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

3. Literature Survey

A literature survey or a literature review in a project report is that section which shows the various analyses and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of the project.

It is the most important part of your report as it gives you a direction in the area of your research. It helps you set a goal for your analysis - thus giving you your problem statement.

When you write a literature review in respect of your project, you have to write the researches made by various analysts - their methodology (which is basically their abstract) and the conclusions they have arrived at. You should also give an account of how this research has influenced your thesis.

Descriptive papers may or may not contain reviews, but analytical papers will contain reviews. A literature review must contain at least 5 - 7 published researches in your field of interest.

4.System Designing

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

Diagrams in the UML

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system.

1. **Activity Diagrams –** We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on the condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.
2. **Use Case Diagrams –** Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high-level view of what the system or a part of the system does without going into implementation details.
3. **Sequence Diagram –** A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the

terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

4. **Class Diagram** – The most widely used UML diagram is the class diagram. It is the building block of all object-oriented software systems. We use class diagrams to depict the static structure of a system by showing the system's classes, their methods and attributes. Class diagrams also help us identify relationships between different classes or objects.

5. Implementation

The software implementation stage involves the transformation of the software technical data package (TDP) into one or more fabricated, integrated, and tested software configuration items that are ready for software acceptance testing. The primary activities of software implementation include the:

- Fabrication of software units to satisfy structural unit specifications.
- Assembly, integration, and testing of software components into a software configuration item.
- Prototyping challenging software components to resolve implementation risks or establish a fabrication proof of concept.
- Dry-run acceptance testing procedures to ensure that the procedures are properly delineated and that the software product (software configuration items (CIs and computing environment) is ready for acceptance testing.

6. Testing

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises Validation and Verification.

Software Validation

Validation is the process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which attempts all that user needs from this software?".
- Validation emphasizes on user requirements.

Software Verification

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by firmly following all design specifications?"
- Verifications concentrate on the design and system specifications.

7.Conclusion

SRS helps the customers to define their needs with accuracy, while it helps the development team understand what the customers need in terms of development. Investing time in writing the SRS document will lead to the successful development of the software the customer needs.

SOFTWARE REQUIREMENTS

Functional Requirements:

- These are statements of services the system should provide

  =>how the system should react to particular inputs and

  =>how the system should behave in particular situations

- In some cases, the functional requirements may also explicitly state

  => What the system should not do

- The functional requirements definition of a system should be both

  => Complete [i.e. It means that all services required by the user should be defined]

  => Consistent [i.e. it means that requirements should not has contradictory definitions]

Non- Functional Requirements:

- These are constraints on the services (Or) functions offered by the system
- They include

  => Timing Constraints

  => Constraint on development process

  => Standards and so on…

- Some non-functional requirements may be process rather than product requirements
- Customer imposes these process requirements for two reasons:

  => System Quality

  => System Maintainability

Non-Functional Requirements Types:

Product Requirements Process Requirements External Requirements

(i) Product Requirements:

These requirements result from the need for the delivered product, to behave in a particular way

Example:

- Requirements on how fast the system must execute and how much memory it

requires

- Reliability Requirements [i.e, acceptable failure rate]
- Portability Requirements

(ii) Organizational Requirements:

- These requirements are consequence of organizational policies and procedures

Example:

Implementation requirements such as programming language (Or) design method

used

- Delivery Requirements which specify when the product and its documentation to be

Delivered

(iii) External Requirements:

- These requirements arise from factors external to the system and its development

process

Example:

- Interoperability Requirements which specify how the system interacts with systems in

other organizations

- Legislative Requirements, which ensure that the system operates within the law

An Overview of UML

Unified Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language; it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system. UML helps software engineers, businessmen and system architects with modeling, design and analysis. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. It's been managed by OMG ever since. The International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

A Conceptual Model of UML

- A conceptual model can be defined as a model which is made of concepts and their relationships.
- A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements −

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

Object Oriented Concepts Used in UML –

1. **Class –** A class defines the blueprint i.e. structure and functions of an object.
2. **Objects –** Objects help us to decompose large systems and help us to modularize our system. Modularity helps to divide our system into understandable components so that we can build our system piece by piece. An object is the fundamental unit (building block) of a system which is used to depict an entity.
3. **Inheritance –** Inheritance is a mechanism by which child classes inherit the properties of their parent classes.
4. **Abstraction –** Mechanism by which implementation details are hidden from the user.
5. **Encapsulation –** Binding data together and protecting it from the outer world is referred to as encapsulation.
6. **Polymorphism –** Mechanism by which functions or entities are able to exist in different forms.

Diagrams in the UML

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system.

There are two broad categories of diagrams and they are again divided into subcategories −

1. **Structural Diagrams –** Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

2.Behavior Diagrams – Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

Structural Diagrams

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable.

These static parts are represented by classes, interfaces, objects, components, and nodes. The four structural diagrams are −

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

1.Class Diagram

Class diagrams are the most common diagrams used in UML. Class diagrams consist of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature.

Active class is used in a class diagram to represent the concurrency of the system.

Class diagrams represent the object orientation of a system. Hence, it is generally used for development purposes. This is the most widely used diagram at the time of system construction.

2.Object Diagram

Object diagrams can be described as an instance of class diagrams. Thus, these diagrams are closer to real-life scenarios where we implement a system. Object diagrams are a set of objects and their relationship is just like class diagrams. They also represent the static view of the system. The usage of object diagrams is similar to class diagrams but they are used to build a prototype of a system from a practical perspective.

3.Component Diagram

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system.

During the design phase, software artifacts (classes, interfaces, etc.) of a system are arranged in different groups depending upon their relationship. Now, these groups are known as components. Finally, it can be said component diagrams are used to visualize the implementation.

4.Deployment Diagram

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team.

Behavioral Diagrams

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspects can be further described as the changing/moving parts of a system.

UML has the following five types of behavioral diagrams −

- Use case diagram
- Sequence diagram
- Collaboration diagram
- Statechart diagram
- Activity diagram

1.Use Case Diagram

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, a use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

2.Sequence Diagram

A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.

Interaction among the components of a system is very important from implementation and execution perspective. Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

3.Collaboration Diagram

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links.

The purpose of the collaboration diagram is similar to a sequence diagram. However, the specific purpose of collaboration diagrams is to visualize the organization of objects and their interaction.

4.Statechart Diagram

Any real-time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system.

Statechart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface, etc. State chart diagram is used to visualize the reaction of a system by internal/external factors.

5.Activity Diagram

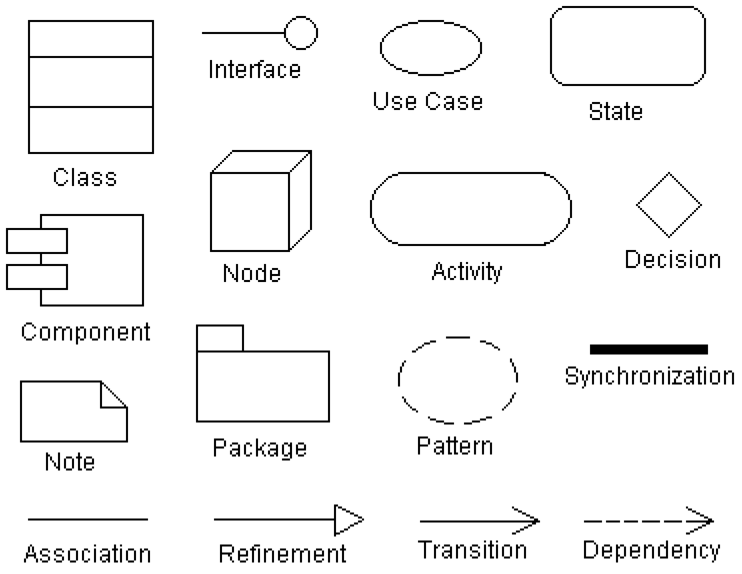
Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

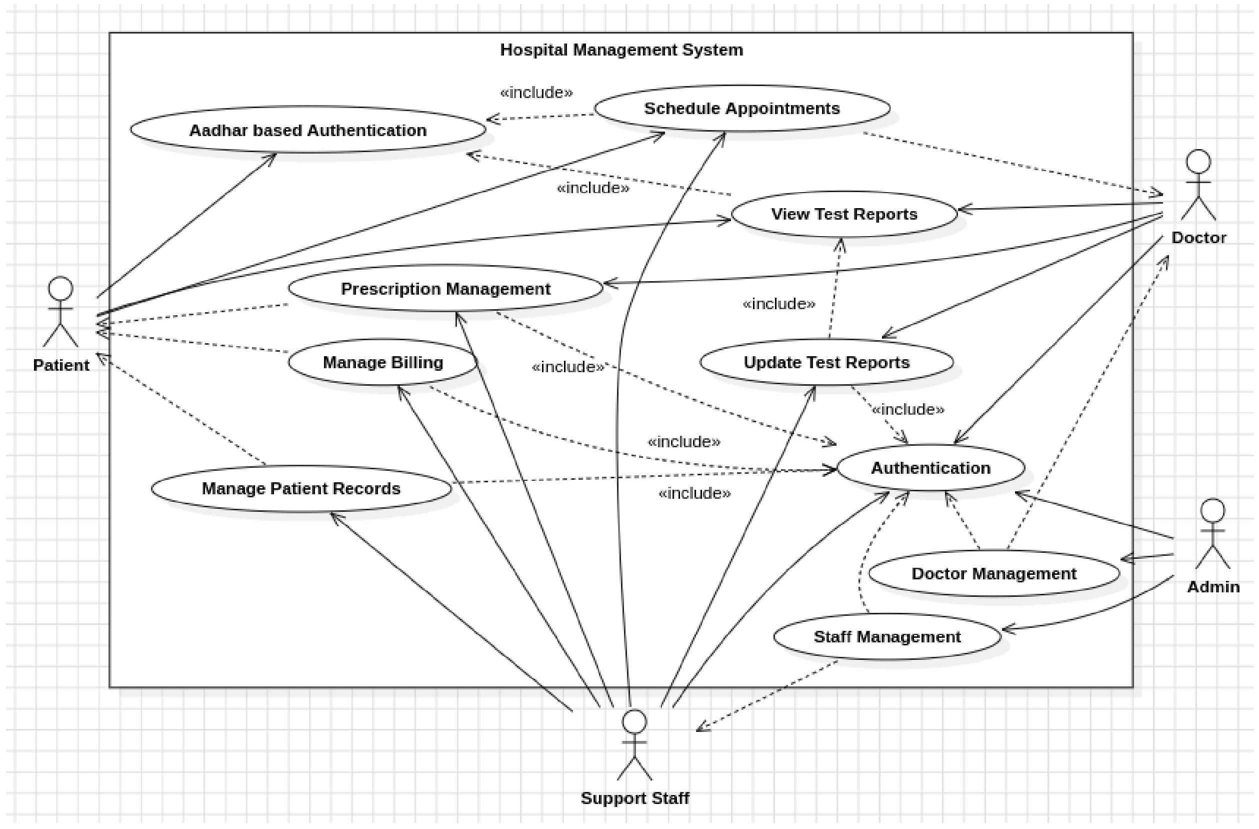
Diagram Elements

Some of the graphical constructs from which diagrams are made are:

- Icon: graphical symbol of fixed size and shape (doesn't hold contents)
- Two-dimensional symbols: have variable size and can expand to hold contents, may be divided into compartments
- Paths: sequences of line segments with attached endpoints. The endpoints are always symbols (no dangling paths). May also have icons at the end to qualify the meaning of the path symbol.
- Strings: text
- Name: A string that uniquely identifies some model element within some scope
- Label: A string attached to a graphic symbol
- Keyword: Text enclosed within "«" and "»" to convey some concept. There are many keywords so we don't need zillions of specialized graphical symbols.
- Expression: A linguistic formula that yields a value
- Some model elements:



Use case diagram:



Use case template

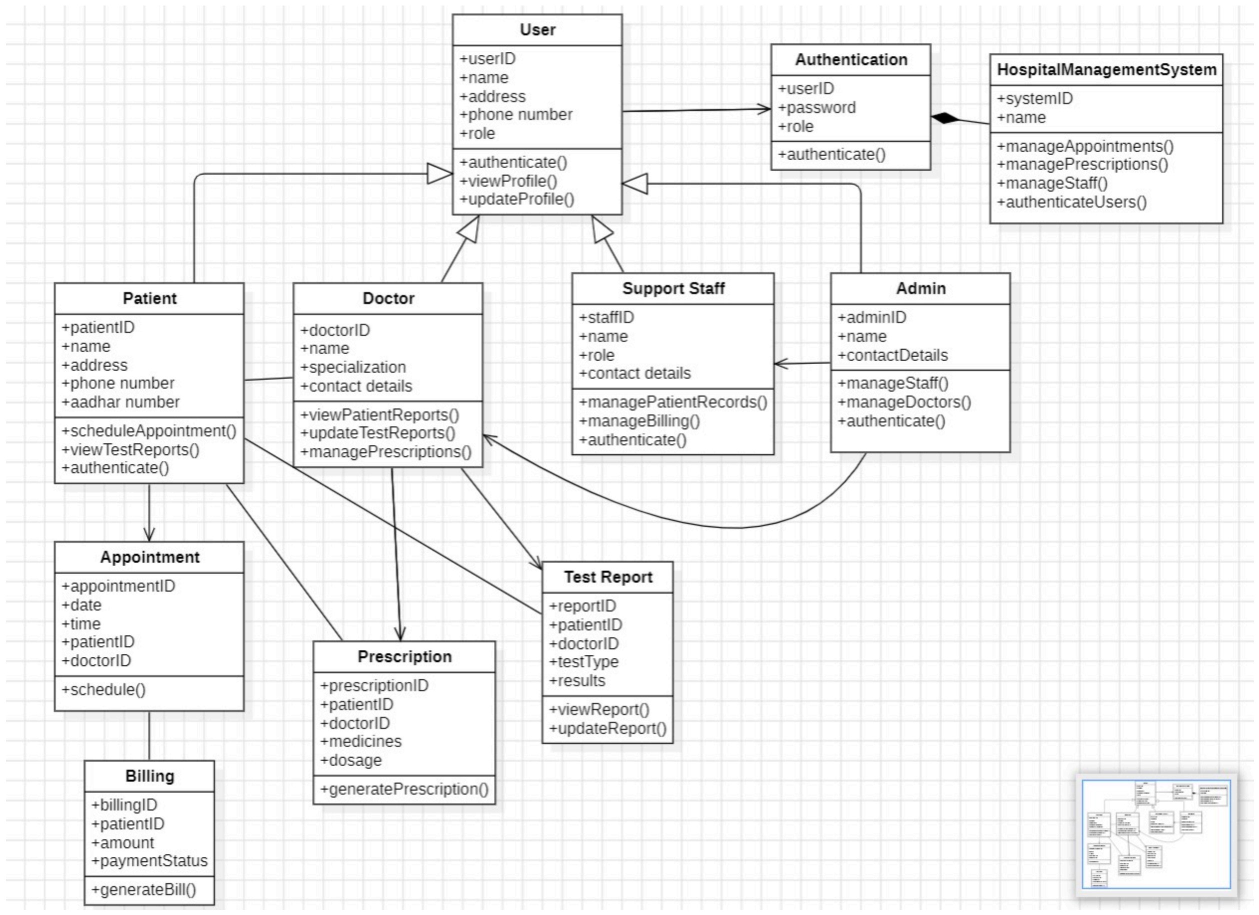
Use Case ID:	040816540236		
Use Case Name:	Hospital Management		
End Objective:	Automate and facilitate hospital operations and patient care management.		
Created by:	1. Vishal HS 2. Sai Gargey Nakka 3. Parmar Jai Atul 4. P Devi Sainath 5. S Sai Bharath	**On (date):**	September 30, 2024

User/Actor:	Patient, Doctor, Support Staff, Admin
Trigger:	Patient admitting to the Hospital

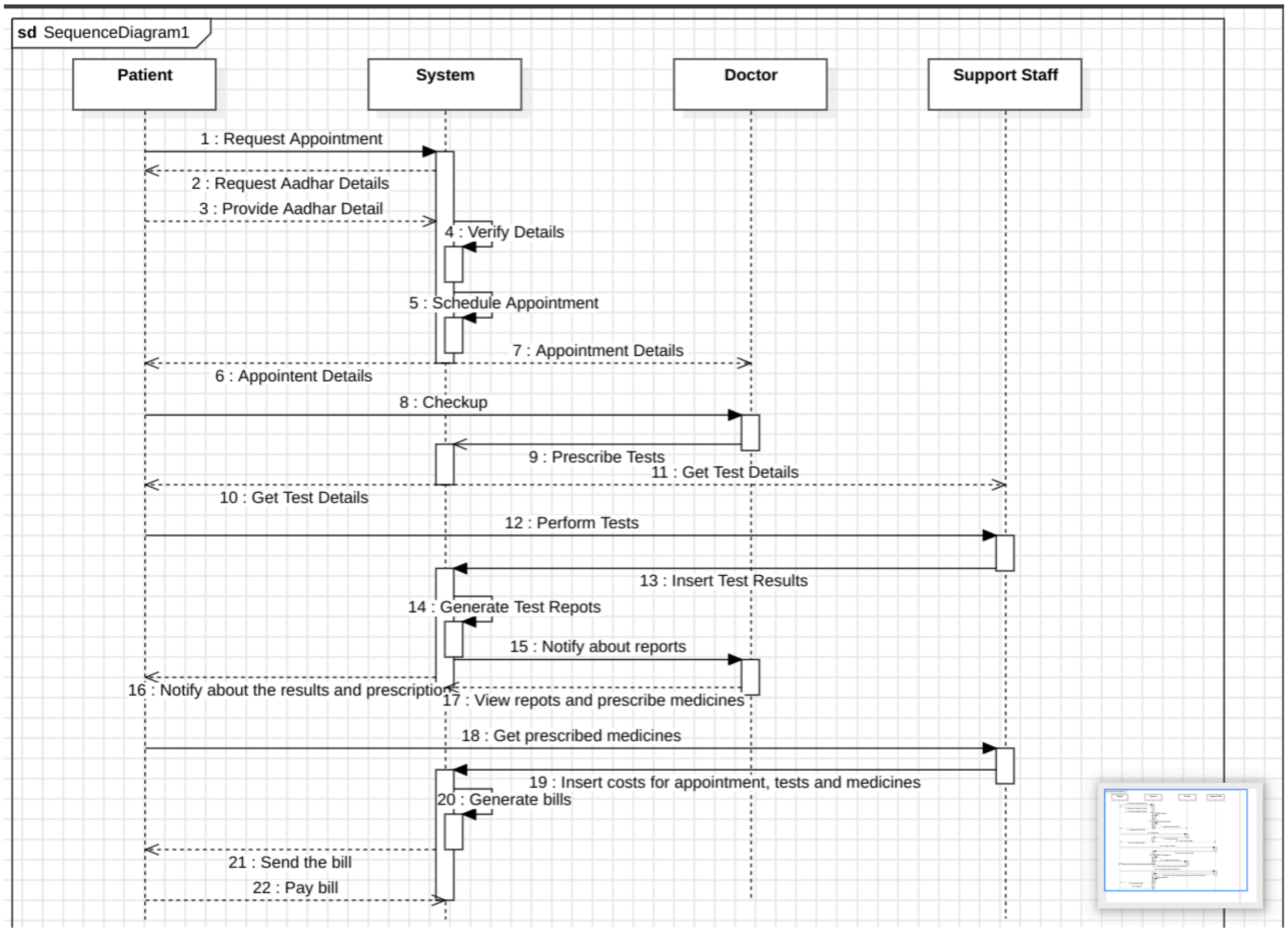
Basic/Normal Flows

User Actions	**System Actions**
The user logs into the system by entering their credentials.	The system validates the username and password and grants access.
The patient searches for doctors or views their medical history.	The system displays available doctors and past medical records.
The doctor views or updates patient records and treatment history.	The system retrieves and updates patient information in real-time.
The admin manages staff roles, schedules, and system access.	The system allows the admin to assign or revoke access permissions.
The support staff schedules appointments for patients.	The system provides available slots and confirms appointments.
The patient views and pays their bills online.	The system processes payments and provides confirmation and receipts.

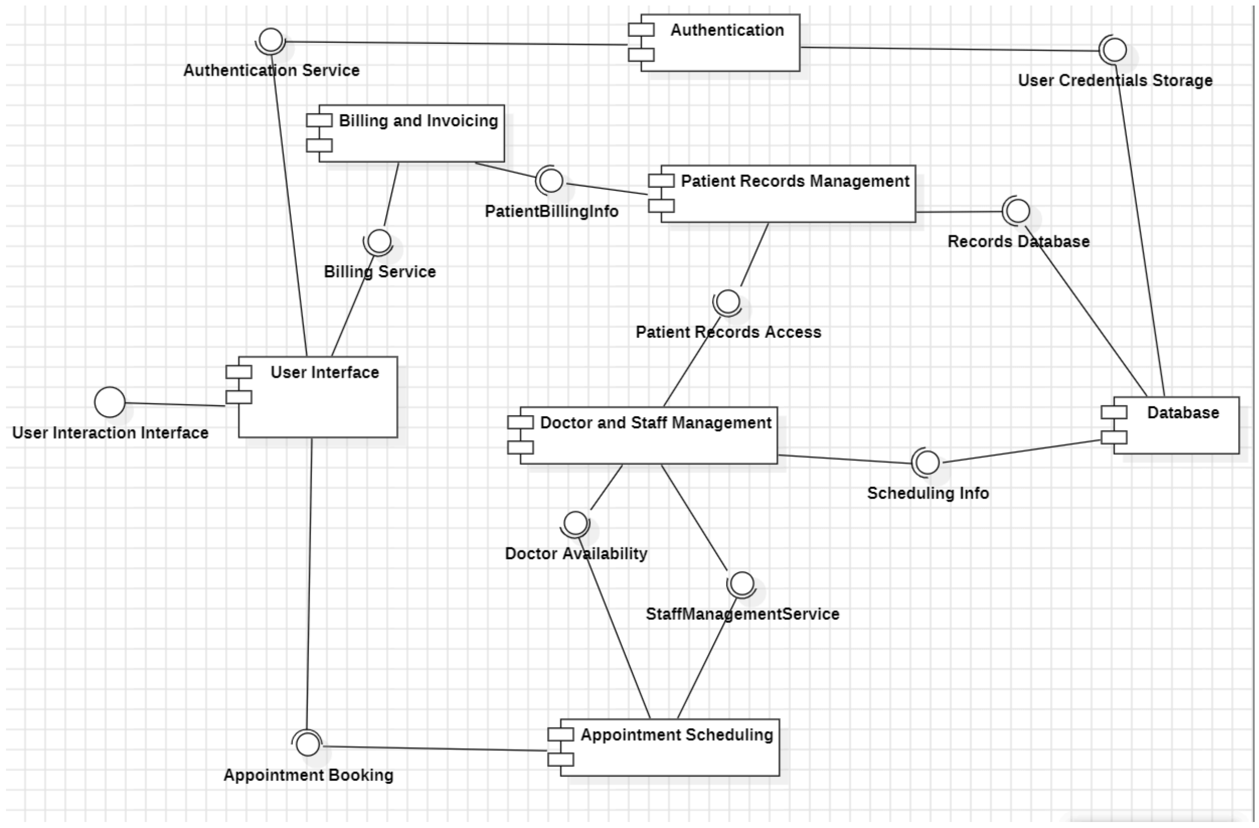
Class Diagram



Sequence Diagram



Component Diagram


Authentication
Authentication Service
User Credentials Storage
Billing and Invoicing
Patient Records Management
PatientBillingInfo
Records Database
Billing Service
Patient Records Access
User Interface
User Interaction Interface
Database
Doctor and Staff Management
Scheduling Info
Doctor Availability
StaffManagementService
Appointment Scheduling
Appointment Booking