

# **SMART OPD SYSTEM WITH VIRTUAL QUEUE FOR OPTIMIZED HEALTHCARE DELIVERY**

*Submitted for partial fulfillment of the requirements*

*for the award of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING – ARTIFICIAL  
INTELLIGENCE & MACHINE LEARNING**

**by**

<b>Korlakanti Nandini</b>	-	<b>21BQ1A4289</b>
<b>Noorbasha Seema</b>	-	<b>22BQ5A4212</b>
<b>Karanki Amarendranadh</b>	-	<b>21BQ1A4277</b>
<b>Katta Gowri</b>	-	<b>21BQ1A4279</b>

Under the guidance of

**Mrs. N. Nalini Krupa,  
Assistant Professor**



**VASIREDDY VENKATADRI  
INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING  
– ARTIFICIAL INTELLIGENCE & MACHINE LEARNING  
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with ‘A’ Grade, ISO 9001:2008 Certified

NAMBUR (V), PEDAKAKANI (M), GUNTUR – 522 508

Tel no: 0863-2118036, url: [www.vvitguntur.com](http://www.vvitguntur.com)

April 2025



## VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Permanently Affiliated to JNTUK, Kakinada, Approved by  
AICTE Accredited by NAAC with 'A' Grade, ISO 9001:20008

Certified

Nambur, Pedakakani (M), Guntur (Gt) - 522508

## DEPARTMENT OF COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

---

### CERTIFICATE

This is to certify that this **Project Report** is the bonafide work of **Ms. Korlakanti Nandini, Ms. Noorbasha Seema, Mr. Karanki Amarendranadh, Ms. Katta Gowri**, bearing Reg. No. **21BQ1A4289, 22BQ5A4212, 21BQ1A4277, 22BQ5A4279** respectively who had carried out the project entitled "**Smart OPD System with Virtual Queue for Optimized Healthcare Delivery**" under our supervision.

#### **Project Guide**

(Mrs. N. Nalini Krupa, Assistant Professor)

#### **Head of the Department**

(Dr. K. Suresh Babu, Professor)

---

Submitted for Viva voice Examination held on \_\_\_\_\_

**Internal Examiner**

**External Examiner**

## **DECLARATION**

We, Ms. Korlakanti Nandini, Ms. Noorbasha Seema, Mr. Karanki Amrendranadh, Ms. Katta Gowri, hereby declare that the Project Report entitled "**Smart OPD System with Virtual Queue for Optimized Healthcare Delivery**" done by us under the guidance of Mrs. N. Nalini Krupa, Assistant Professor, CSE-Artificial Intelligence & Machine Learning at Vasireddy Venkatadri Institute of Technology is submitted for partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science Engineering - Artificial Intelligence & Machine Learning. The results embodied in this report have not been submitted to any other University for the award of any degree.

DATE :

PLACE : Nambur

### **SIGNATURE OF THE CANDIDATE (S)**

Korlakanti Nandini,  
Noorbasha Seema,  
Karanki Amarendranadh,  
Katta Gowri.

## **ACKNOWLEDGEMENT**

We take this opportunity to express my deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, we express my deep gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B.Tech program.

We express my sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B.Tech program.

We express my sincere gratitude to **Dr. K. Suresh Babu**, Professor & HOD, Computer Science Engineering - Artificial Intelligence & Machine Learning, Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

We would like to express my sincere gratefulness to our Guide **Mrs. N. Nalini Krupa**, Assistant Professor, Computer Science Engineering - Artificial Intelligence & Machine Learning for her insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to express our sincere heartfelt thanks to our Project Coordinator **Mr.N.Bala Yesu**, Associate Professor, CSE-Artificial Intelligence & Machine Learning for his valuable advices, motivating suggestions, moral support, help and coordination among us in successful completion of this project.

We would like to take this opportunity to express my thanks to the **Teaching and Non-Teaching Staff** in the Department of Computer Science Engineering - Artificial Intelligence & Machine Learning, VVIT for their invaluable help and support.

SIGNATURE OF THE CANDIDATE (S)

**Korlakanti Nandini**

**Noorbasha Seema**

**Karanki Amarenndrnadh**

**Katta Gowri**

## **TABLE OF CONTENTS**

<b>CH No</b>	<b>Title</b>	<b>Page No</b>
	Contents	i
	List of Figures	iv
	Nomenclature	v
	Abstract	vi
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background of the Project	1
	1.2 Problem Statement	1
	1.3 Objectives of the Project	2
	1.4 Scope of the Project	2
	1.5 Methodology Overview	3
	1.6 Organization of the Report	4
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>7</b>
	2.1 Previous Research and Related Work	7
	2.2 Existing Solutions and Their Limitations	8
	2.3 Gap Analysis	10
	2.4 Relevance of the Project	11
<b>3</b>	<b>SYSTEM ANALYSIS</b>	<b>12</b>
	3.1 Requirement Analysis	12
	3.1.1 Functional Requirements	12
	3.1.2 Non-functional Requirements	13
	3.2 Feasibility Study	13
	3.2.1 Technical Feasibility	14
	3.2.2 Economic Feasibility	14
	3.2.3 Operational Feasibility	14
	3.3 Proposed System Overview	14

<b>4</b>	<b>SYSTEM DESIGN</b>	<b>15</b>
4.1	System Architecture	15
4.2	Block Diagram	17
4.3	Data Flow Diagrams (DFD)	19
4.4	UML Diagrams	21
4.4.1	Use Case Diagram	22
4.4.2	Class Diagram	23
4.4.3	Sequence Diagram	24
4.4.4	Activity Diagram	25
4.5	Database Design	26
4.5.1	ER Diagram	26
4.5.2	Schema Design	27
<b>5</b>	<b>IMPLEMENTATION</b>	<b>29</b>
5.1	Programming Languages and Technologies Used	29
5.2	Development Tools and Environments	30
5.3	Module-Wise Implementation Details	31
5.4	Algorithms and Logic Used	32
<b>6</b>	<b>TESTING AND RESULTS</b>	<b>32</b>
6.1	Testing Methodologies	32
6.1.1	Unit Testing	32
6.1.2	Integration Testing	33
6.1.3	System Testing	33
6.2	Test Cases and Reports	35
6.3	Performance Evaluation	35
6.4	Screenshots of Application Output	35

<b>7</b>	<b>CONCLUSION AND FUTURE WORK</b>	<b>43</b>
7.1	Summary of Findings	43
7.2	Key Achievements and Contributions	43
7.3	Challenges Faced	44
7.4	Future Scope and Improvements	44
	<b>APPENDIX</b>	<b>45</b>
	Publication Certificates	

## LIST OF FIGURES

<b>Figure No.</b>	<b>Figure Name</b>	<b>Page No.</b>
4.1.1	System Architecture Diagram	15
4.2.1	Block Diagram	17
4.3.1	Data Flow Diagram (DFD) - Level 0	19
4.3.2	Data Flow Diagram (DFD) - Level 1	20
4.4.1.1	Use Case Diagram	22
4.4.2.1	Class Diagram	23
4.4.3.1	Sequence Diagram	24
4.4.4.1	Activity Diagram	25
4.5.1.1	ER Diagram	26
4.5.2.1	Database Schema Design	28
5.4.1	Flowchart for Model Training and Prediction	31
6.1.1	Testing Methodology Flowchart	32
6.4.1	Screenshots of Application Output	35-42

## NOMENCLATURE

<b>Acronym</b>	<b>Full Form</b>
<b>API</b>	Application Programming Interface
<b>EC2</b>	Elastic Compute Cloud
<b>F1-Score</b>	F1-Score
<b>Confusion Matrix</b>	Confusion Matrix
<b>SVM</b>	Support Vector Machine
<b>Random Forest</b>	Random forest Classifier

## ABSTRACT

The increasing demand for healthcare services has made hospital resource management a critical concern. Managing patient inflow, bed occupancy, and the availability of medicines and consumables presents major challenges that can directly impact patient satisfaction and hospital performance. In many healthcare facilities, manual processes and inefficient systems lead to long waiting times, misallocation of resources, and frequent stockouts—resulting in compromised patient care and increased operational stress.

This project introduces an Smart OPD System with Virtual Queue for Optimized Healthcare Delivery that integrates Machine Learning (ML) and Predictive Analytics to optimize and automate key hospital operations. The system leverages historical and real-time data to forecast patient inflow patterns, dynamically manage outpatient department (OPD) queues, and predict bed availability based on current occupancy trends and discharges. This ensures timely admissions and better bed utilization.

The ML-powered queue management model prioritizes patients based on severity, appointment type, and historical trends, thereby reducing idle times for healthcare providers and enhancing patient satisfaction. In terms of inventory management, the system continuously monitors usage rates of medicines and consumables, predicts future demand, and automates reorder alerts to prevent shortages or wastage.

A standout feature is the system's integration with a city-wide hospital network, allowing multiple hospitals to share real-time information on bed availability and inventory status. This enables efficient resource distribution, especially during peak demand or emergencies. Additionally, the system includes a self-service patient portal that facilitates appointment booking, real-time queue updates, and notification alerts, minimizing congestion and improving transparency.

**Keywords** — hospital management, machine learning, predictive analytics, real-time patient tracking, OPD queue management, automated inventory control, smart alerts, dynamic resource allocation, triage system, city-wide hospital network, healthcare efficiency, patient prioritization.

# CHAPTER 1

## INTRODUCTION

### 1.1 Background of the Project

In the rapidly evolving world of healthcare, the integration of technology has become essential for improving the efficiency, accessibility, and quality of medical services. One of the most crucial components in any healthcare infrastructure is the Outpatient Department (OPD), where a majority of patients first interact with healthcare professionals. However, traditional OPD management systems are often plagued with inefficiencies such as long waiting times, lack of systematic appointment scheduling, and inadequate record management.

To address these challenges, the development of an Advanced OPD Management System becomes imperative. This project aims to modernize and automate the functioning of OPD services by leveraging advanced web technologies, Artificial Intelligence (AI), and robust backend systems. By incorporating features such as appointment booking, doctor-patient communication, real-time updates, voice-based interaction, and AI-powered decision support, the proposed system offers a comprehensive solution for managing outpatient services efficiently.

The integration of AI also enables predictive analysis and intelligent resource allocation, which can significantly enhance patient care and reduce operational burdens on hospital staff. Moreover, the system ensures accessibility through multi-language support and cloud integration, making it a scalable and inclusive solution for diverse healthcare environments.

### 1.2 Problem Statement

In traditional OPD settings, patients often face significant challenges such as long queues, inefficient appointment scheduling, lack of real-time updates, and poor communication between doctors and patients. Manual processes not only consume valuable time but also increase the chances of human error in record-keeping and appointment handling. This leads to patient dissatisfaction and hampers the overall quality of healthcare delivery.

Furthermore, most existing OPD systems lack intelligent automation and adaptability to modern needs such as voice-based assistance, multi-language support, and personalized patient experiences. There is also minimal integration of AI-driven diagnostics or predictive analytics, which limits the system's ability to assist medical professionals in decision-making.

Hence, there is a clear need for an advanced, user-friendly, and intelligent OPD Management System that addresses these limitations. Such a system should streamline appointment booking, enhance communication, manage records securely, and offer AI-based features to aid in diagnosis and patient care — all while maintaining accessibility and inclusiveness across various user demographics.

## **1.3 Objectives of the Project**

The main objective of this project is to design and develop a web-based OPD (Outpatient Department) Management System that streamlines and simplifies the overall functioning of outpatient services in healthcare institutions. The system is intended to replace manual and paper-based processes with a centralized, digital platform that enhances operational efficiency and reduces human error. A key goal is to provide a structured appointment booking system that allows patients to schedule visits in advance, thereby reducing long queues and minimizing waiting times at hospitals.

The project aims to offer separate modules for patients, doctors, and administrators, each with dedicated features to manage their roles effectively. Patients will be able to register, log in, and book appointments based on available time slots. Doctors can view scheduled appointments, access basic patient details, and manage consultations. Administrators will have access to overall control, including managing user accounts, updating doctor schedules, and overseeing appointment logs.

The system is designed with a clean and simple user interface using Django templates, ensuring it is easy to navigate even for users with limited technical knowledge. It includes basic record-keeping functionality to store patient information, appointment details, and doctor availability securely using a relational database such as SQLite. The project also includes features for generating consultation summaries and bills after appointments, which improves the documentation process within the outpatient department.

Overall, the objective is to build a functional, efficient, and user-friendly OPD Management System that serves the core needs of a small to medium healthcare setup, without relying on external services like Gmail or SMS-based authentication. The project focuses on core functionality, reliability, and ease of use, making it a practical solution for digitizing outpatient operations.

## **1.4 Scope of the Project**

The scope of this project is to design and develop a web-based OPD (Outpatient Department) Management System that serves as a digital replacement for manual operations in small to mid-scale healthcare facilities. It focuses on the core functionality required to manage outpatient visits efficiently, without the use of advanced technologies such as Artificial Intelligence, SMS-based OTP verification, or Gmail-based authentication. The system aims to simplify the process of booking appointments, managing doctor schedules, maintaining basic patient information, and providing administrative oversight, all through a centralized and user-friendly web application.

This project is tailored for environments where resources are limited, and where simple, cost-effective solutions are more practical than complex, cloud-based or AI-driven systems. The

application consists of three main user roles—**Patient**, **Doctor**, and **Admin**—each provided with a dedicated dashboard to access and manage their respective activities. Patients can register, log in, view available doctors, book appointments, and check appointment details. Doctors can log in to view their daily schedules and patient lists, allowing them to prepare for consultations in advance. Admins act as the system managers, with access to all records, including the ability to manage users, monitor appointment activity, and make changes to doctor availability or schedules.

All records, including appointment details and user information, are stored securely in a local database (SQLite), making the system lightweight and ideal for offline deployment as well. Since the system does not use Gmail or SMS APIs, all communication and notifications are handled within the web application interface—reducing both the complexity and cost of deployment. The frontend is built using Django templates with HTML, CSS, and JavaScript, ensuring responsiveness and compatibility with most modern browsers.

Additional features include appointment confirmation messages displayed within the user dashboard, a basic billing or consultation summary generator for doctors, and a search/filter mechanism to help users find doctors or appointments based on criteria like department or date. The system is also designed with a modular approach so that features like uploading medical files, generating printable reports, or language localization can be added in future versions without affecting the core functionality.

In summary, this project provides a simple yet efficient OPD management solution that enhances hospital workflow, improves patient service delivery, and promotes organized medical record-keeping—all without the complexities or costs associated with advanced technology integrations. It is particularly suitable for small clinics, rural health centers, student projects, and training modules focused on practical, real-world web application development in healthcare.

## 1.5 Methodology Overview

The development of the OPD Management System follows a structured and phased methodology that ensures clarity, efficiency, and systematic implementation of all features. The methodology adopted for this project is based on the Waterfall Model, a traditional software development approach where each phase is completed before moving on to the next. This model is ideal for academic and small-scale projects where the requirements are clearly defined at the beginning, and the focus is on building a reliable and functional system without the need for frequent changes or iterations.

The first phase involves requirement analysis and planning, during which the core features of the system were identified—such as user registration, login, appointment booking, patient record handling, doctor scheduling, and basic administrative controls. The focus was on building a simple, secure, and functional system without external dependencies like SMS or email APIs, keeping the scope limited to in-system functionalities only.

Following this, the system design phase was carried out, where the overall structure of the application was defined. This included deciding the layout of the web pages, the database schema (using SQLite), user roles and permissions, and the flow of information between different components. The design ensured a clear separation of functionalities among patients, doctors, and administrators to maintain usability and security.

In the implementation phase, the actual coding of the system was done using Django as the backend framework, with HTML, CSS, and JavaScript for the frontend. Django templates were used to create dynamic web pages, allowing the content to be rendered based on user interactions. The database was connected using Django's ORM (Object-Relational Mapping), which helped in storing and retrieving data such as appointment details, user information, and doctor availability.

Once the core modules were built, the testing phase ensured that each component functioned correctly. Unit testing and manual testing were performed to validate user inputs, verify login credentials, and ensure smooth navigation across the system. Attention was given to handling invalid inputs, preventing unauthorized access, and ensuring a smooth user experience without bugs or crashes.

The final phase involved deployment and documentation, where the system was prepared for use on a local or internal server. Comprehensive documentation was prepared to explain the usage, installation steps, and features of the system. This helps future developers or users understand the flow and expand the project if needed.

Overall, this methodology ensures that the project remains simple, maintainable, and user-focused, making it suitable for educational purposes, small clinics, or local healthcare setups. Each phase contributes to building a stable and reliable system that addresses the core needs of outpatient management in a digital format.

## 1.6 Organization of the Report

The report for the “OPD Management System” project is organized into multiple chapters to provide a comprehensive understanding of the system's development process, from conceptualization to implementation and testing. Each chapter is structured to cover specific aspects of the project in a logical sequence, helping readers to follow the design and technical decisions made throughout the course of development.

### Chapter 1: Introduction

This chapter introduces the overall purpose and background of the project, the motivation for developing the OPD Management System, and its relevance in addressing common challenges faced by hospitals and clinics. It also outlines the specific objectives, scope, and methodology adopted for the project.

## **Chapter 2 :Literature Review**

This section reviews existing hospital and outpatient management systems, highlighting their features, limitations, and the gap that this project aims to fill. It sets the context for the project's uniqueness and explains why a lightweight, non-AI solution is practical and needed in certain healthcare environments.

## **Chapter 3: System Analysis**

This chapter discusses the system's functional and non-functional requirements. It also includes a feasibility analysis, addressing the technical, economic, and operational viability of the system. The user roles (patient, doctor, admin) and their respective access levels are also discussed in detail.

## **Chapter 4: System Design**

Here, the technical blueprint of the system is presented. It includes descriptions of the system architecture, database schema, user interface layout, and data flow diagrams. The chapter outlines how Django and SQLite are used to structure the backend and how Django templates are used for the frontend design.

## **Chapter 5: Implementation**

This section details the coding process and the tools used during development. It explains how various modules (like registration, login, appointment booking, and scheduling) were implemented using Django and how HTML, CSS, and JavaScript were integrated for a responsive UI.

## **Chapter 6:Testing and Evaluation**

In this chapter, the testing process is described, including manual testing, validation of user inputs, and overall system behavior under different scenarios. The reliability, efficiency, and user-friendliness of the system are evaluated and any limitations observed are documented.

## **Chapter 7 : Conclusion and FutureWork**

This chapter summarizes the entire project, highlighting the successful implementation of core features. It also suggests possible improvements and additional features such as integration with government health APIs, report uploads, or multi-language support that can be implemented in future versions.

## **Chapter 8: References**

This final section lists all the resources, research materials, and documentation referred to during the development of the project. References are cited in IEEE or APA style using tools like Zotero or Mendeley.

### **Tools Used**

**Writing:** MS Word and Google Docs were used for documentation and formatting of the report.

**Development:** Django framework, Python, HTML, CSS, and SQLite were used for building the web application.

**Research and Referencing:** Google Scholar, online documentation, and research papers were referenced. Zotero and Mendeley were used for managing citations.

## CHAPTER 2

### LITERATURE REVIEW

#### **2.1 Previous Research and Related Work**

The management of outpatient departments (OPDs) has been the focus of several research efforts and software development projects over the past decade. With the growing patient load and the demand for efficient healthcare delivery, researchers and developers have attempted to design systems that streamline administrative processes, improve patient experience, and reduce the manual burden on healthcare staff.

Numerous commercial and open-source hospital management systems (HMS) have been developed to address these needs. Systems like OpenMRS (Open Medical Record System), GNU Health, and OpenEMR have gained global recognition for offering comprehensive digital health solutions. These platforms provide modules for appointment scheduling, patient record maintenance, billing, pharmacy management, and even lab report integration. However, these solutions are generally designed for medium to large hospitals and often require significant resources, both technical and financial, for deployment and maintenance. Their complexity, need for customization, and steep learning curves can make them unsuitable for smaller healthcare facilities or academic implementation.

Academic research has also contributed significantly to the domain. Several studies have proposed models for improving hospital workflow, reducing patient wait times, and enhancing data management using technology. For instance, research has shown that digitizing OPD services leads to a substantial reduction in errors caused by manual data entry and paper-based documentation. Other studies have investigated the impact of digital appointment systems in improving patient satisfaction and resource allocation in outpatient departments.

Despite these advancements, there is a noticeable gap when it comes to lightweight, easy-to-use OPD management systems tailored specifically for small clinics or student projects. Many existing solutions are either too advanced or lack modularity and clarity in implementation, making them difficult to adopt in environments with limited infrastructure. Moreover, while some recent systems integrate AI, SMS, and cloud-based features, they may introduce unnecessary complexity and cost in contexts where basic patient scheduling and data handling are sufficient.

In contrast to these solutions, the OPD Management System proposed in this project is designed to be simple, functional, and accessible. It eliminates the need for internet-based features like Gmail or SMS verification and avoids the complexity of AI-driven modules. Instead, it focuses on core OPD functionalities such as user role-based access, appointment booking, and patient-doctor interaction, implemented using Django and SQLite. The goal is to provide a user-friendly platform that can operate entirely in offline or intranet settings, offering ease of deployment and usage in low-resource environments.

By analyzing previous research and existing systems, this project positions itself as a practical, scalable, and educational solution that fills a specific need in the healthcare technology spectrum.

## **2.2 Existing Solutions and Their Limitations**

Several hospital and outpatient department (OPD) management systems have been developed over the years, offering varied levels of functionality based on the scale and needs of healthcare facilities. Prominent systems like OpenMRS, GNU Health, Bahmni, and OpenEMR have been successfully implemented in large healthcare environments and research contexts. While these solutions offer robust features, they also come with certain limitations that make them unsuitable or impractical for smaller institutions, rural clinics, or educational use cases.

Most existing systems are built with enterprise-level scalability in mind, often requiring complex setup procedures, powerful servers, internet connectivity, and trained IT personnel. For example, solutions like OpenMRS and GNU Health support integrations with electronic health records (EHR), laboratory information systems, and pharmacy inventory, but their high system requirements and advanced configurations can become a barrier for deployment in resource-constrained environments.

Additionally, many of these systems rely heavily on third-party integrations such as SMS-based appointment reminders, email verifications via Gmail or other providers, and cloud-based storage. While these features are beneficial in larger settings, they introduce dependency on internet access, incur operational costs, and may not be ideal in offline or intranet-based clinics. Moreover, reliance on such integrations can raise privacy concerns, especially in rural or private setups where user trust in external services may be limited.

Another notable limitation in many available solutions is the lack of user-centric design. These systems often prioritize functionality over usability, leading to cluttered interfaces and steep learning curves, especially for non-technical users such as medical staff or administrative workers. This complexity can slow down adoption and lead to errors in data entry or appointment management.

From a student project perspective, these advanced solutions are typically too large in scope and difficult to fully understand or modify for academic demonstration purposes. The documentation may be outdated or overly technical, and customizing such systems for specific use cases without breaking dependencies can be a major challenge.

In contrast, the system proposed in this project was developed with simplicity, clarity, and essential OPD functions in mind. It does not rely on AI algorithms, SMS gateways, or email-based authentication, making it suitable for offline environments. It focuses strictly on core operations such as patient registration, appointment booking, doctor management, and admin supervision. Using Django as the backend framework and SQLite as the lightweight database, the system is easy to install, run, and understand—making it ideal for both small clinics and academic submission.

This review of existing systems and their drawbacks justifies the need for a more streamlined, offline-friendly, and minimalistic OPD management system like the one developed in this project.

## 2.3 Gap Analysis

Gap analysis is a critical step in identifying the shortcomings of existing systems and highlighting the need for the proposed solution. Although many hospital and outpatient management systems currently exist, most are tailored for large-scale institutions with significant resources, technical infrastructure, and trained personnel. These systems, while feature-rich, do not effectively address the practical needs of small hospitals, clinics, or student-level projects, where simplicity, ease of use, and minimal dependencies are essential.

One of the key gaps identified is the lack of lightweight, offline-capable OPD systems. Most existing solutions are built with cloud-based architectures that require continuous internet connectivity for access and functionality. This presents a challenge in rural and semi-urban healthcare settings where internet availability may be inconsistent or unavailable altogether. In contrast, the proposed system can operate entirely offline using Django and SQLite, ensuring functionality even in limited infrastructure environments.

Another significant gap is the over-reliance on third-party services, such as SMS gateways and email authentication (e.g., Gmail login). While these services are useful in well-funded hospital networks, they complicate deployment and increase operational costs for small healthcare providers. The proposed system avoids these complexities by providing a direct login/registration mechanism without external verification dependencies, making it easier to use and manage.

Furthermore, most enterprise-level hospital systems include modules for AI-driven diagnostics, pharmacy management, and inpatient monitoring, which—while impressive—are not always necessary or practical for outpatient departments in small clinics. These features may overwhelm users and lead to underutilization of the system. The OPD Management System developed in this project fills this gap by focusing only on essential modules such as patient registration, appointment scheduling, doctor-patient assignment, and basic administrative control.

Finally, a gap exists in terms of educational value. Many available systems are either closed-source or too complex for students to understand, customize, or present as academic projects. This project bridges that gap by offering a well-documented, modular, and clearly structured codebase that not only solves a real-world problem but also serves as a valuable learning tool for students and developers alike.

In summary, the proposed OPD Management System effectively addresses the existing gaps in scalability, simplicity, usability, and accessibility, providing a practical and focused solution where more complex systems fall short.

## 2.4 Relevance of the Project

The growing demand for digitization in healthcare has highlighted the importance of developing simple, efficient, and cost-effective software solutions, particularly for outpatient department (OPD) services. While large hospitals may benefit from enterprise-level hospital management systems, small clinics, semi-urban healthcare centers, and educational institutions often struggle to adopt such complex systems due to budget constraints, lack of technical infrastructure, or the absence of trained personnel.

This project is highly relevant in addressing the specific needs of small-scale OPD environments. By offering a focused and lightweight management system that operates without the need for advanced features like AI-based diagnostics or third-party integrations (such as SMS alerts or Gmail authentication), the solution aligns well with the realities of low-resource settings. It provides a practical alternative to existing bulky systems by streamlining patient registration, appointment booking, and doctor scheduling—functions that are central to outpatient workflows.

In regions with limited internet access or frequent network disruptions, an offline-compatible system such as the one developed in this project can ensure continuous operation, avoiding the risk of service interruptions that cloud-dependent platforms might face. The system's use of Django and SQLite ensures it can run on basic hardware setups and be deployed locally with ease.

From an academic and training standpoint, this project also holds educational value. It demonstrates core software engineering principles including modular design, database integration, and role-based access control—all within a realistic healthcare context. It serves as a useful reference and prototype for students, educators, and developers looking to understand or expand upon OPD management systems.

Furthermore, the COVID-19 pandemic emphasized the need for efficient healthcare delivery systems and contactless appointment management. Even beyond pandemic scenarios, efficient OPD systems help reduce overcrowding, minimize patient wait times, and allow medical staff to focus more on patient care rather than administrative tasks. Thus, the relevance of this project extends beyond the academic domain to real-world application in rural and small-town clinics where technology adoption is still emerging.

Overall, the project addresses critical challenges in healthcare management by offering a user-friendly, cost-effective, and scalable OPD management solution, reinforcing its importance in both educational and practical healthcare settings.

# CHAPTER 3

## SYSTEM ANALYSIS

### **3.1 Requirement Analysis**

Requirement analysis is a fundamental phase in the software development life cycle that involves identifying and documenting the functional and non-functional requirements of the system. It ensures that the final product meets user expectations and functions effectively within the intended environment. For the OPD Management System, the requirements are categorized into functional and non-functional components.

#### **3.1.1 Functional Requirements**

These are the specific operations and behaviors the system must support to meet its intended purpose:

##### **Authentication and Role Management**

- The system must allow registration and login for three types of users: Admin, Doctor, and Patient.
- Each role must have access to specific modules and functionalities relevant to their responsibilities.

##### **Patient Registration**

- Patients should be able to register themselves by entering basic information such as name, age, gender, contact details, and symptoms.

##### **Appointment Booking**

- Patients must be able to view available doctors and their time slots, and book an appointment accordingly.

##### **Doctor Management**

- Admin must be able to add, update, or delete doctor profiles.
- Doctors must be able to view and manage their appointment schedules.

##### **OPD Queue Management**

- Doctors should be able to see the list of scheduled patients and mark consultations as complete.

##### **Admin Dashboard**

- The admin panel must display system statistics such as total patients, doctors, appointments, and allow management of system users.

##### **Medical Notes**

- Doctors should be able to record medical observations or notes during consultation sessions for future reference.

##### **Basic Reports and Logs**

- The system must allow the generation of appointment and patient logs for administrative review.

### **3.1.2 Non-functional Requirements**

These describe the system's performance characteristics and development constraints:

#### **Usability**

- The interface must be intuitive and easy to navigate for users with minimal technical expertise.
- Simple Django templates should be used to maintain clarity and reduce load times.

#### **Scalability**

- The system should be able to accommodate additional users or clinics with minimal configuration changes.

#### **Security**

- User authentication must be enforced to prevent unauthorized access.
- Data must be protected within the database, ensuring role-based restrictions.

#### **Portability**

- The application should run smoothly on any system with basic Python and Django setup, without dependency on external servers.

#### **Performance**

- The system should handle multiple user requests efficiently with quick page loads and database interactions.

#### **Reliability**

- The system should function consistently without unexpected crashes or data loss.

#### **Maintainability**

- The codebase must follow clean, modular structure and documentation for easy future updates or enhancements.

## **3.2 Feasibility Study**

A feasibility study evaluates the practicality of a proposed project and helps determine whether the system can be developed and implemented successfully within the given constraints. For the OPD Management System, the feasibility has been assessed across five key dimensions: technical, operational, economic, legal, and schedule feasibility.

### **3.2.1 Technical Feasibility**

This examines whether the required technology, tools, and skills are available to develop the system.

- The system uses Django/Flask for the backend and React Native/Expo for the frontend.
- AI models are implemented using TensorFlow/Keras and Scikit-Learn.
- Data storage is managed using MongoDB, PostgreSQL, or SQLite.
- The system is hosted on cloud platforms like AWS/GCP for scalability.

### **3.2.2 Economic Feasibility**

Economic feasibility assesses whether the project is financially viable.

- Development costs include software tools, cloud hosting, and AI model training.
- The project reduces hospital management expenses by automating administrative and medical processes.
- Future scalability ensures long-term cost-effectiveness.

### **3.2.3 Operational Feasibility**

Operational feasibility ensures that the system is practical for end users.

- Hospitals, doctors, and patients benefit from reduced waiting times and improved service.
- The system integrates with existing healthcare IT systems.
- Easy-to-use interface ensures a smooth transition from manual to digital processes.

## **3.3 Proposed System Overview**

The Hospital Management System (HMS) aims to streamline hospital operations, patient care, and administration through automation and AI. Key highlights include:

- A web and mobile application for appointment and hospital management.
- AI-based disease prediction for preliminary medical assistance.
- Secure data management with role-based access for patients, doctors, and admins. 21 22
- Automated billing, notifications, and chat support for enhanced user experience.
- Efficient OPD and IPD management for hospitals of all sizes.
- Integration with government health APIs for standardized healthcare services.

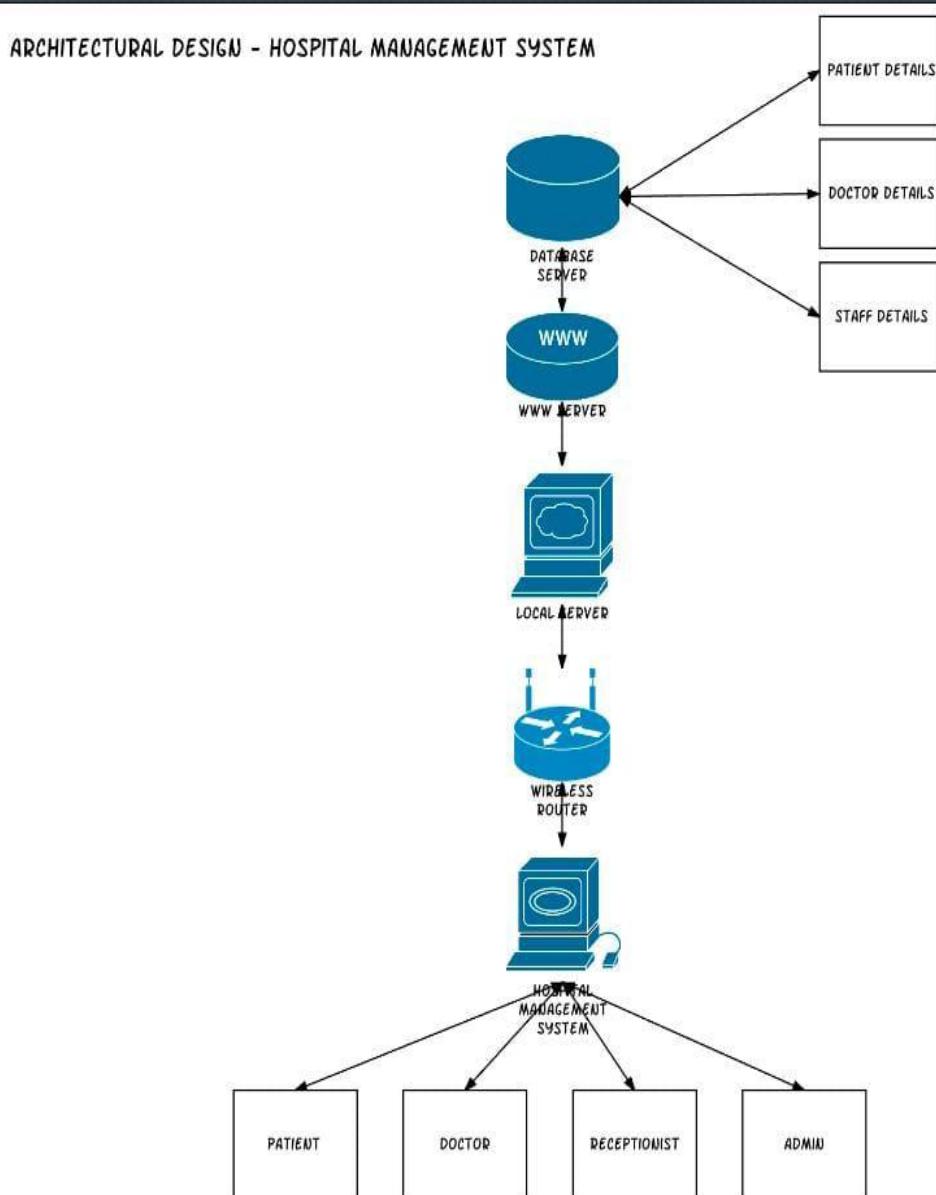
# CHAPTER 4

## SYSTEM DESIGN

### 4.1 System Architecture

The OPD(Outpatient Department) Management System is a comprehensive, AI-Powered Web application built using the Django framework. The system aims to streamline patient Registration, appointment scheduling, and doctor-patient interaction while integrating modern Features such as voice interaction, multilingual support, and disease prediction using machine Learning models

Figure 4.1.1 System Architecture Diagram



## Architecture Overview

The architecture follows a modular, layered approach consisting of the following components:

**Presentation Layer (Frontend)** This is the user interface layer that allows patients, doctors, and admins to interact with the system.

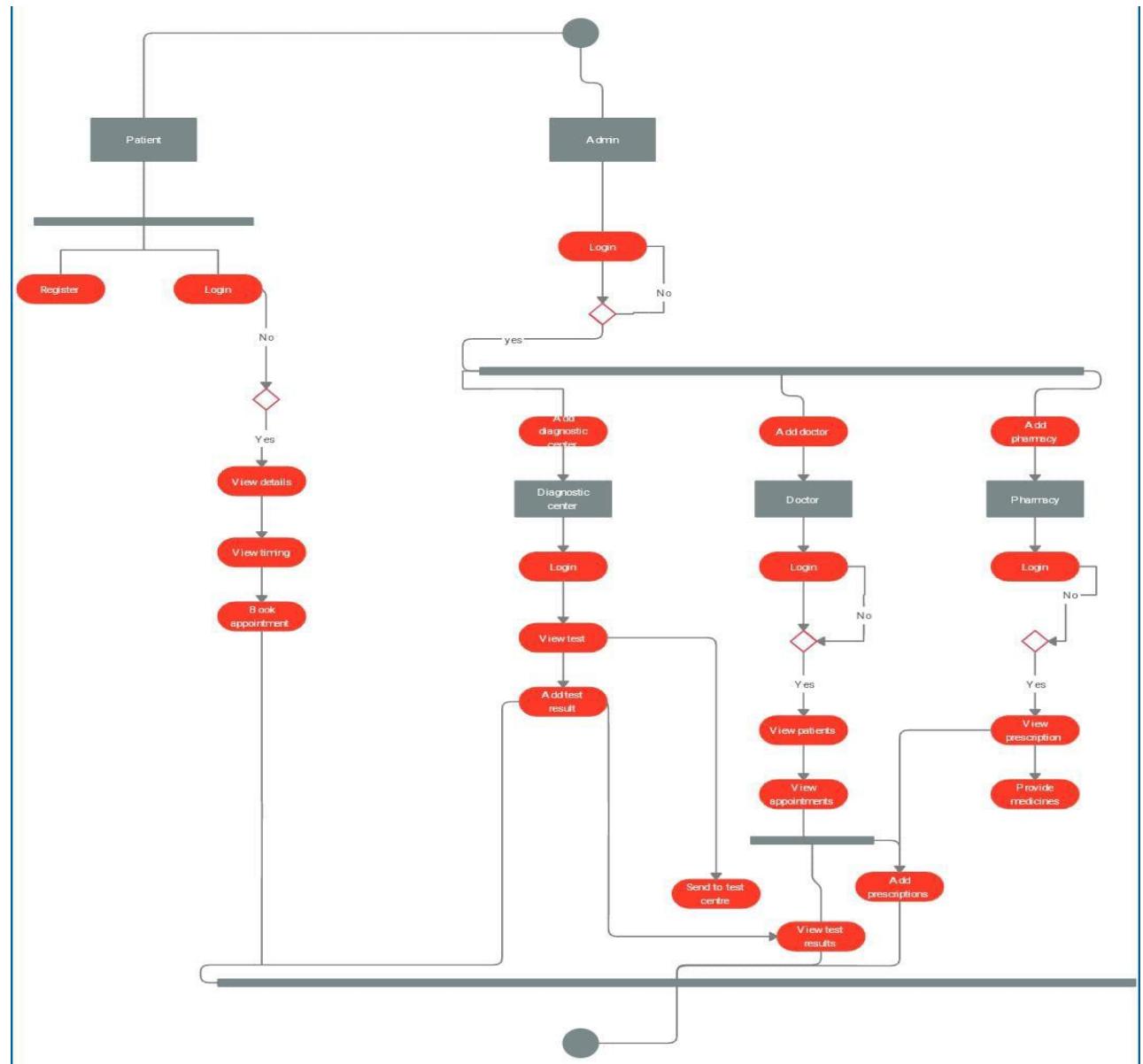
- Built using HTML, CSS, and JavaScript with Django Templates
- Provides login/registration for different roles
- Enables appointment booking, OPD management, and feedback submission
- Accessible on web browsers with responsive design
- Application Layer (Django Backend)** Handles the core business logic, user authentication, and API processing.
- Developed using the Django web framework
- Utilizes Django's MTV (Model-Template-View) architecture
- Processes requests, connects to the database, and handles server-side logic
- Includes modules for appointment handling, OPD priority scheduling, and payment processing
- Data Layer (Database)** Responsible for storing all structured data related to users, appointments, prescriptions, feedback, etc.
- Implemented using SQLite (or PostgreSQL optionally for production)
- Utilizes Django's ORM (Object-Relational Mapping) for efficient data handling
- Supports secure data storage for sensitive medical records
- AI and ML Modules** Enhance user experience and automate tasks using intelligent models.
- **AI Chatbot:** Handles general queries and symptom checking
- **ML Models:** Used for disease prediction and priority-based OPD scheduling
- Developed in Python (TensorFlow/Keras/Scikit-learn)

## 4.2 Block Diagram

The Block Diagram of the OPD Management System provides a high-level visual representation of the major components and their interaction within the system. It outlines the flow of information between users and system modules, highlighting how the system handles patient registration, doctor scheduling, and appointment management.

The system follows a modular architecture, where each functional block operates independently but communicates with others for seamless data flow. The key components of the system are:

**Figure 4.2.1 Block Diagram of the System**



#### **4.2.1. User Interface(Frontend)**

This is the first layer where users interact with the system. It includes separate login interfaces for:

- **Admin**
- **Doctor**
- **Patient**

Each role-based user accesses specific functionalities after authentication. The frontend is built using **HTML, CSS, and Django templates**, ensuring a clean and user-friendly experience.

#### **4.2.2. Patient Module**

Allows patients to:

- Register themselves
- View available doctors
- Book appointments
- Check appointment status

All patient data is securely stored in the backend database for administrative and doctor access.

#### **4.2.3. Doctor Module**

Enables doctors to:

- View their daily appointments
- Access patient details
- Record consultation notes
- Update status after consultation

#### **4.2.4. Admin Module**

The admin has control over the entire system, including:

- Adding/removing doctors
- Managing user accounts
- Viewing system statistics
- Generating reports

#### **4.2.5. Database Layer**

All data—patient records, doctor profiles, appointments, and login credentials—are stored in an SQLite database. The backend handles all Create, Read, Update, and Delete (CRUD) operations via Django ORM for efficiency and security.

#### **4.2.6. Appointment Management Module**

This module acts as the core of the OPD system. It connects patients and doctors by enabling:

- Real-time scheduling

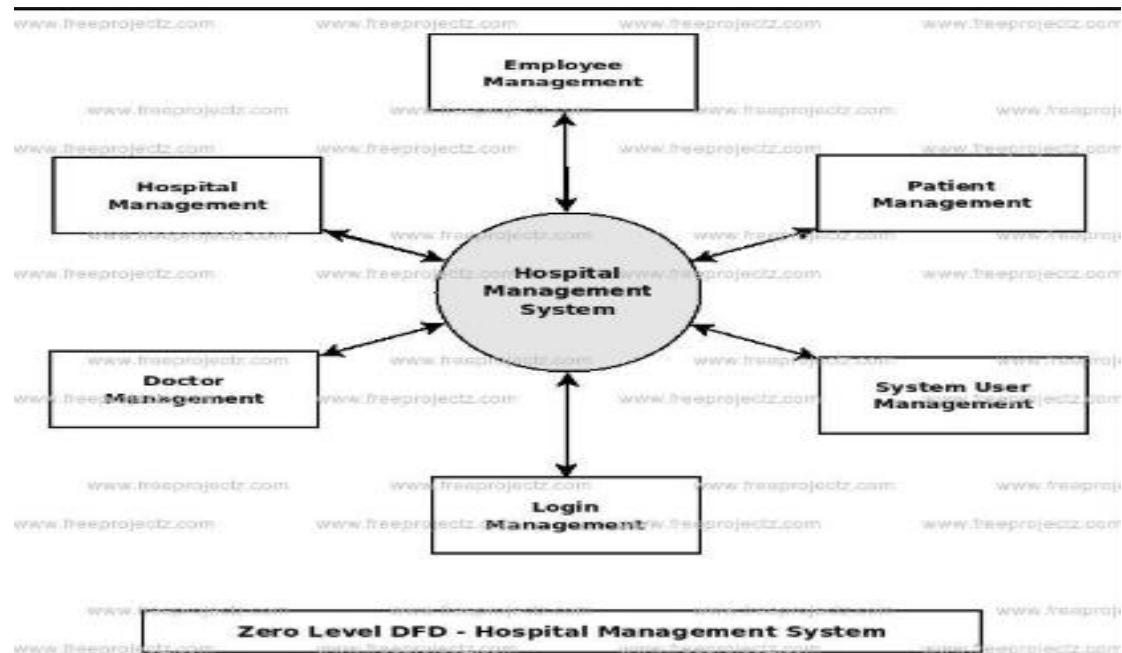
- Conflict-free booking
- Status tracking of consultations

## 4.3 Data Flow Diagrams (DFD)

### Level 0 DFD:

- Represents the overall system with inputs (patient registration, appointment booking) and outputs (doctor consultation, prescription generation).

**Figure 4.3.1 Data Flow Diagram (DFD) - Level 0**



### External Entities:

- Admin
- Doctor
- Patient

### Central Process:

- OPD Management System

### Data Stores (Implied):

- User Database
- Appointment Records
- Patient Information

### Data Flow Description:

1. Admin provides doctor details, user management input, and receives system reports and

logs.

2. Doctor provides availability details, updates consultation status, and retrieves patient appointment lists and medical notes.
3. Patient registers, submits appointment requests, and receives confirmation or status updates.

All these interactions flow through the central OPD system, which processes the data and stores it in the appropriate backend database.

### Level 1 DFD (Detailed View)

This level offers insight into how the system handles patient registration, appointment scheduling, doctor access, and admin controls.

#### Main Processes in Level 1 DFD:

User Registration & Login (Process 1.0):

- Patient, Doctor, or Admin provides credentials or registration details.
- The system verifies and grants access to the appropriate dashboard.
- Validated credentials are stored in the User Database.

Appointment Management (Process 2.0):

- Patient books an appointment by selecting a doctor and time slot.
- System checks for availability and updates the Appointment Database.
- Confirmation is sent to the patient.

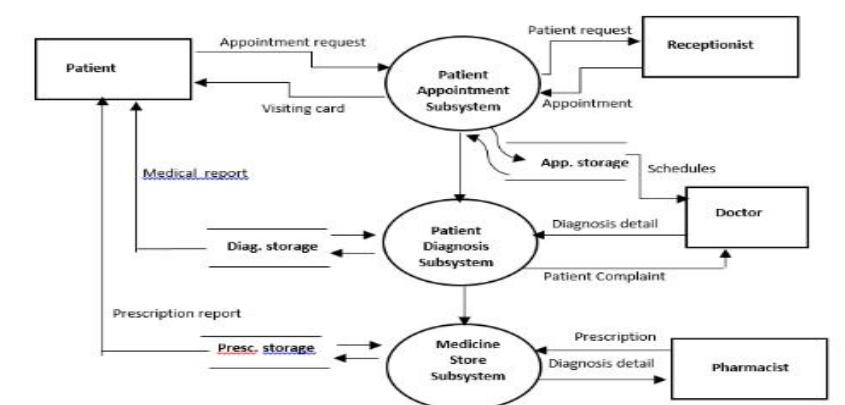
Doctor Dashboard & Consultation Update (Process 3.0):

- Doctor logs in to view appointments.
- Doctor updates consultation notes/status.
- Updates are saved to the Patient Records and Appointment Database.

Admin Control Panel (Process 4.0):

- Admin manages doctors, reviews system activity, and generates reports.
- Admin can view or edit the User and Appointment databases.

**Figure 4.3.2 Data Flow Diagram (DFD) - Level 1**



## **4.4 UML Diagrams**

Unified Modeling Language (UML) diagrams are a vital part of software development, used to visually represent the design and functionality of a system. They help developers, designers, and stakeholders to understand how the system works, how different components interact, and what users can expect. UML diagrams provide a standard way to visualize system architecture and serve as a blueprint for implementation.

For the OPD Management System, various UML diagrams have been developed to represent both the structural and behavioral aspects of the system. These diagrams help in planning the application's flow, interactions, and data structure.

UML diagrams are used to represent how different parts of the system interact, how users engage with the application, and how data flows through the system. They help in analyzing the requirements, designing the architecture, and validating the logic before actual implementation. Each diagram captures a different aspect of the system, ensuring complete coverage from design to deployment.

Using UML diagrams during the development of the Smart OPD System with optimized queue ensures better communication, minimizes errors, and enhances the overall quality and efficiency of the software. These diagrams also assist in documentation and make future enhancements or troubleshooting much easier.

#### 4.4.1 Use Case Diagram

The Use Case Diagram displays the interaction between the system and external users (actors). It identifies the main system functionalities and the roles that users perform.

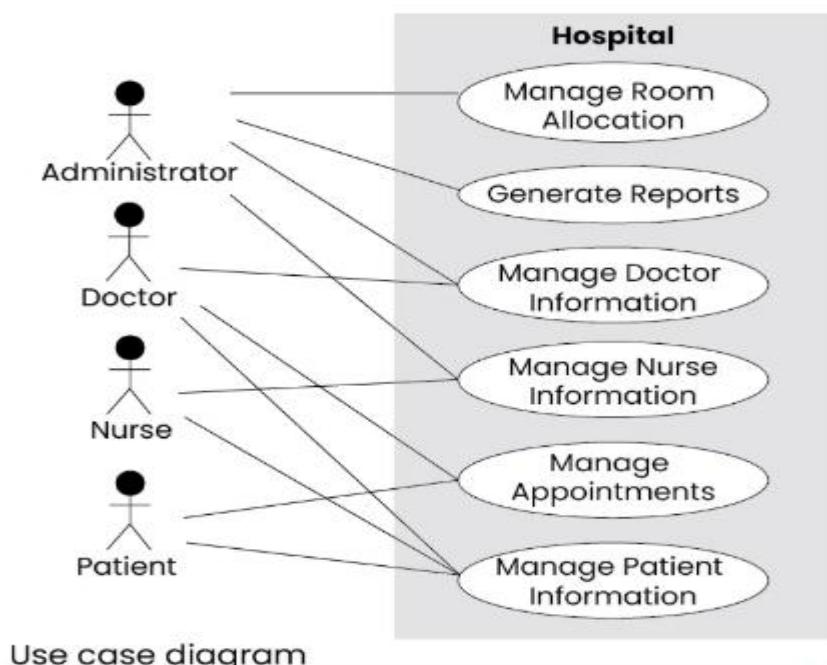
##### Actors:

- Patient
- Doctor
- Admin

##### Main Use Cases:

- Patient Registration and Login
- Appointment Booking
- Doctor Viewing and Updating Appointments
- Admin Managing Users and Appointments

Figure 4.4.1.1 Use Case Diagram

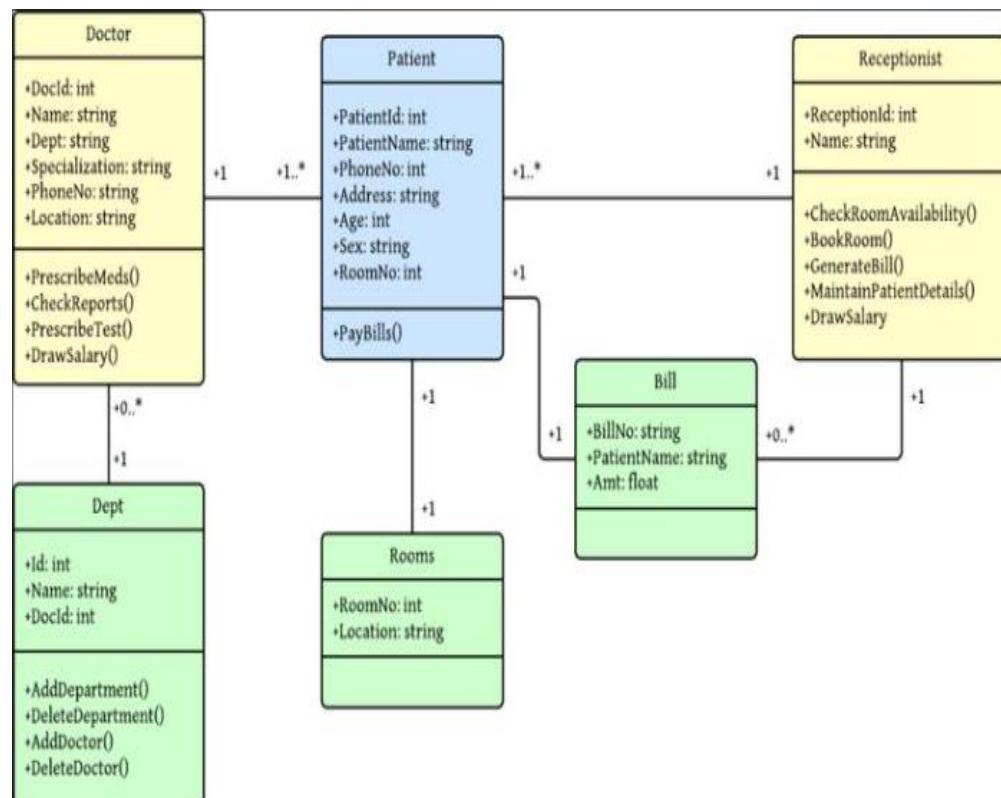


## 4.4.2 Class Diagram

The Class Diagram illustrates the system's classes, their attributes, methods, and the relationships among them. It provides a blueprint of the object-oriented structure of the system. Key Classes:

- User: Common attributes like username, password, and role.
- Patient: Inherits from User and includes medical history and contact details.
- Doctor: Inherits from User and includes specialization and availability.
- Appointment: Contains appointment details like date, time, doctor, and patient.
- Admin: Has additional permissions to manage users and appointments.

**Figure 4.4.2.1 Class Diagram**



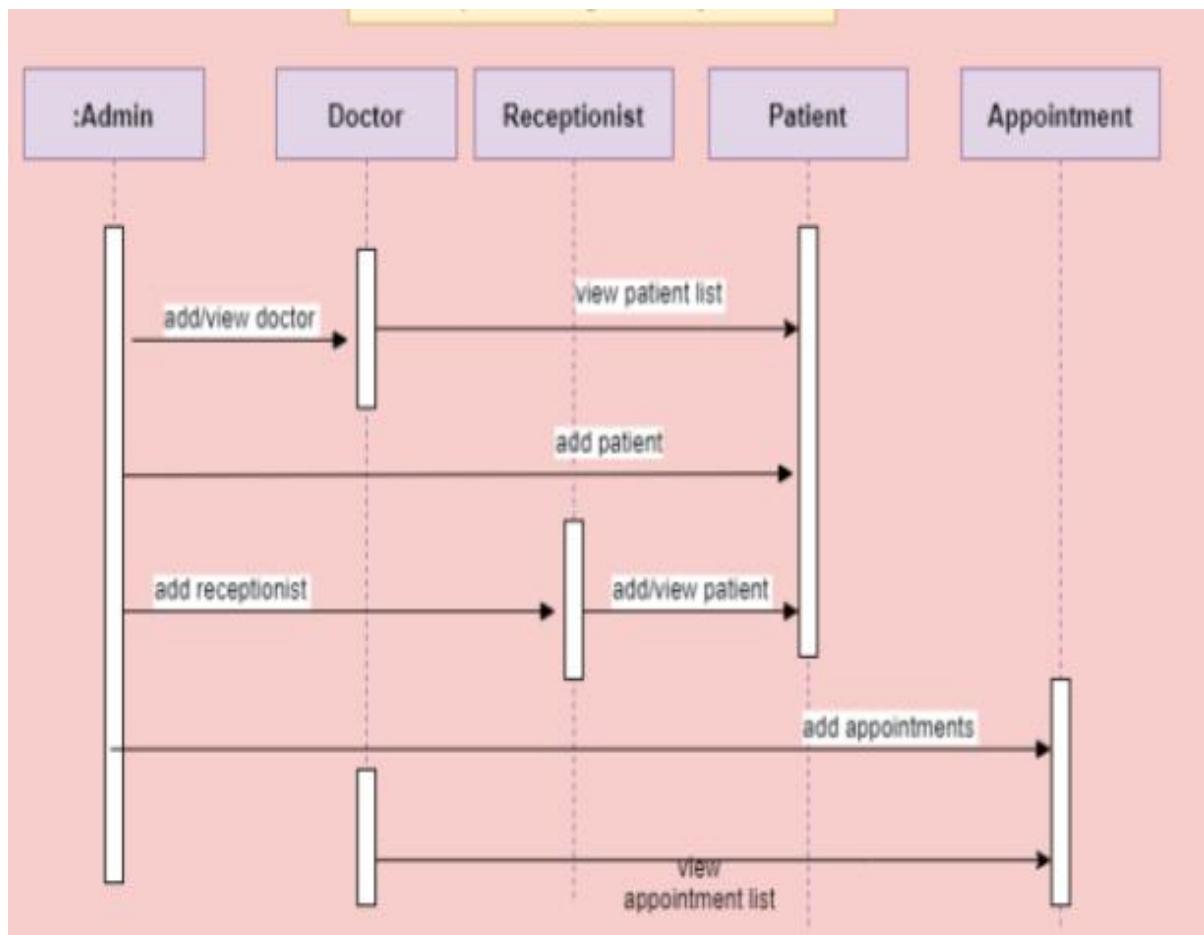
### 4.4.3 Sequence Diagram

The Sequence Diagram represents the order of interactions between system components for specific scenarios, such as appointment booking.

Scenario Example – Appointment Booking:

- Patient logs in.
- Patient selects a doctor and available time slot.
- The system checks availability.
- The appointment is confirmed and saved in the database.
- Confirmation is shown to the patient.

Figure 4.4.3.1 Sequence Diagram

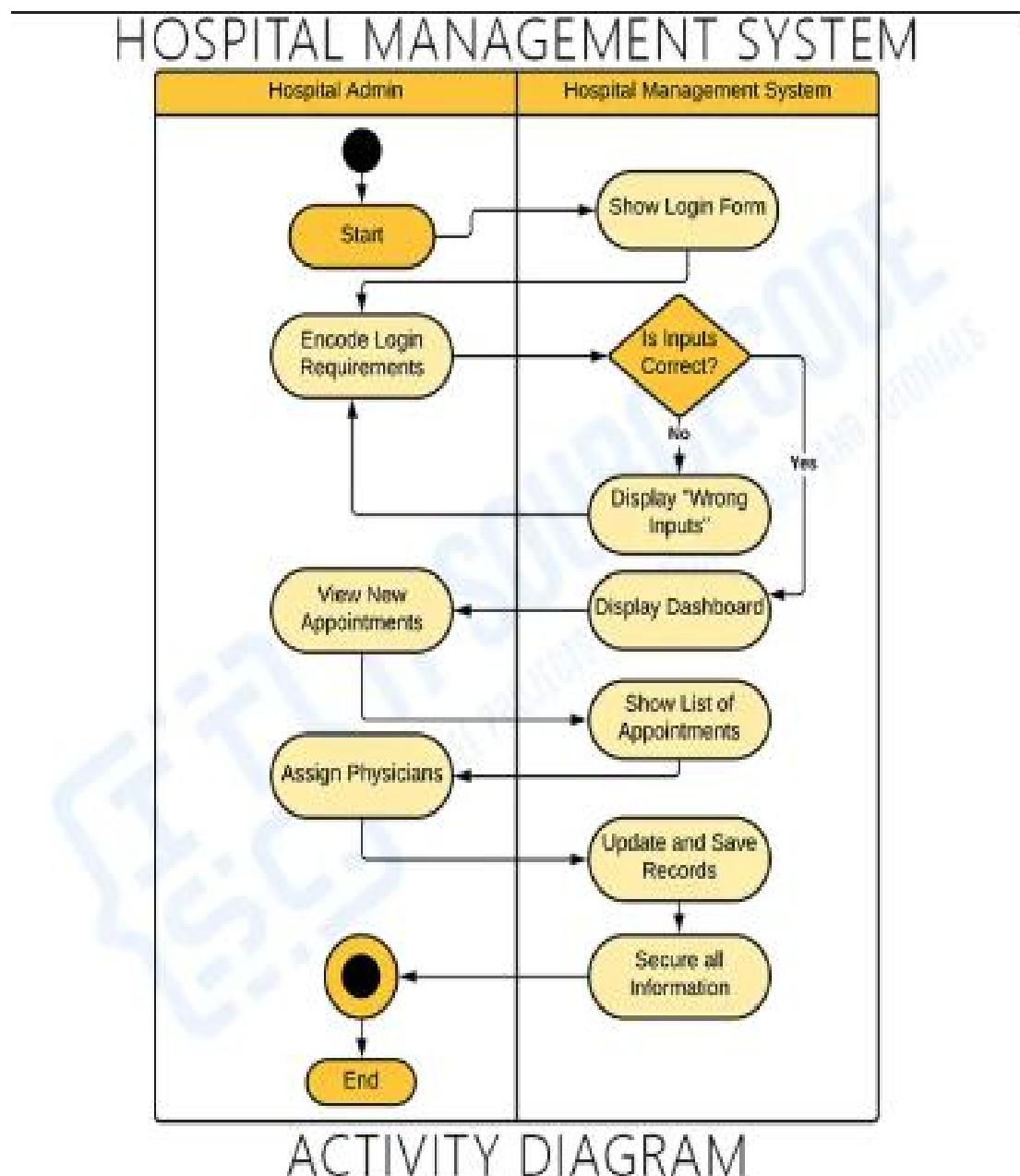


#### 4.4.4 Activity Diagram

The Activity Diagram describes the dynamic workflow of the system. It shows various activities involved in a process and the transitions between them.

Workflow Example – Doctor Viewing Appointments:

Figure 4.4.4.1 Activity Diagram



## 4.5 Database Design

The database design plays a critical role in the successful functioning of the OPD Management System. A well-structured database ensures efficient data storage, quick retrieval, consistency, and integrity across all modules of the application. The system uses SQLite as the backend database due to its lightweight nature, ease of integration with Django, and suitability for small to medium-scale applications.

This section outlines the design of the key database tables and their relationships within the system.

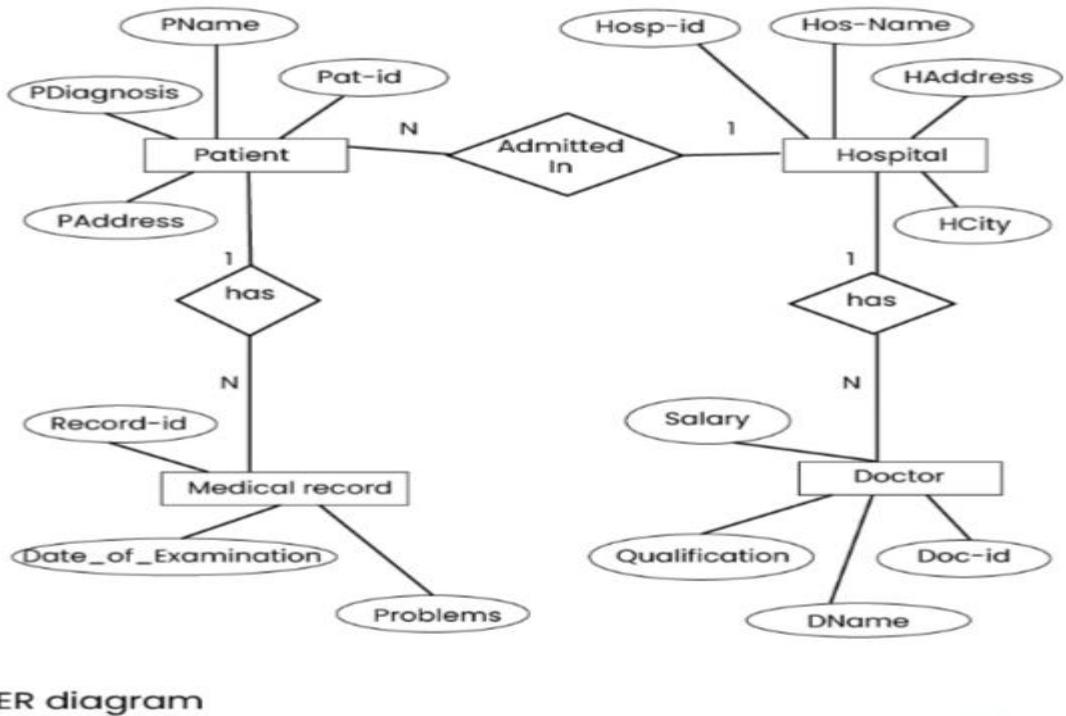
### 4.5.1 ER Diagram

An Entity-Relationship (ER) Diagram is a visual representation of the entities (tables) in a database and the relationships between them. It helps in understanding the logical structure of the database and ensures that the data is organized efficiently.

For the OPD Management System, the ER diagram defines how entities like Patients, Doctors, Appointments, and Admins are interrelated. The primary goal is to manage user roles, appointments, and patient data in a structured and efficient manner.

An ER Diagram (Entity-Relationship Diagram) is a visual tool used in database design to represent the structure of data and how different entities relate to each other. It consists of entities, which are objects or concepts like Student or Course, and these are represented using rectangles. Each entity has attributes (such as Name or RollNo) shown in ovals, and one or more of these may be a primary key, which uniquely identifies each instance of the entity. Relationships between entities, like a student enrolling in a course, are represented using diamonds and describe how entities are connected. The diagram also includes cardinality, which defines the numerical relationship between entities, such as one-to-many or many-to-many. ER diagrams are essential in planning and organizing a database before its actual implementation.

**Figure 4.5.1.1 ER Diagram**



## Key Entities and Relationships

### Entities:

#### User

- Attributes: user\_id (PK), username, password, email, role

#### Patient

- Attributes: patient\_id (PK), user\_id (FK), full\_name, age, gender, contact\_number, address

#### Doctor

- Attributes: doctor\_id (PK), user\_id (FK), full\_name, specialization, contact\_number, availability

#### Appointment

- Attributes: appointment\_id (PK), patient\_id (FK), doctor\_id (FK), appointment\_date, appointment\_time, symptoms, status

#### Medical\_Record (Optional)

- Attributes: record\_id (PK), patient\_id (FK), doctor\_id (FK), diagnosis, prescription, date

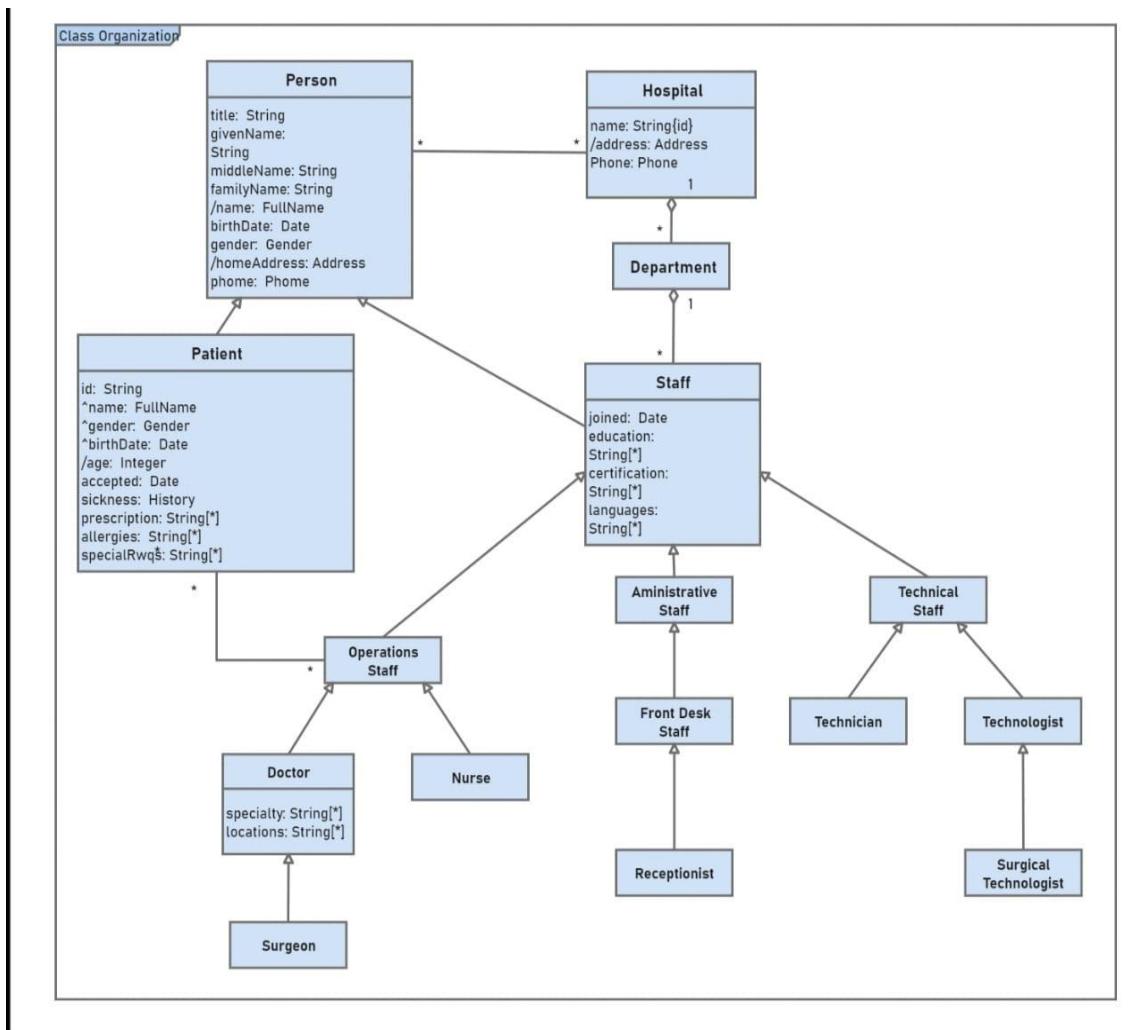
## 4.5.2 Schema Design

Schema design is a crucial step in database development. It defines how data is logically organized and how the relationships among data are handled. In the OPD Management System, the schema design focuses on storing essential information such as user roles, patient and doctor profiles, appointments, and medical records in a normalized and structured format.

The schema is designed using SQLite for backend integration with the Django framework. Below is the relational schema of the main tables used in the system.

- Tables:
  - User (id, name, email, role, password\_hash)
  - Doctor (id, name, specialization, schedule, contact\_info)
  - Patient (id, name, age, gender, contact\_info, medical\_history)
  - Appointment (id, patient\_id, doctor\_id, date, status)
  - Prescription (id, appointment\_id, medicine, dosage)
  - Billing (id, patient\_id, amount, payment\_status)

Figure 4.5.2.1 Database Schema Design



## CHAPTER 5

### IMPLEMENTATION

#### **5.1 Programming Languages and Technologies Used**

This project is implemented using the following programming languages and technologies:

##### **Backend Technologies**

- Python – Used for backend development, handling logic, and AI-based functionalities.
- Django Framework – A high-level web framework for building the OPD management system, handling authentication, database interactions, and business logic.
- Django REST Framework (DRF) – Used for creating APIs if the system integrates with external applications.
- SQLite – A lightweight database management system used to store patient, doctor, and appointment details.

##### **Frontend Technologies**

- HTML, CSS, JavaScript – Used for designing the frontend pages of the OPD system.
- Bootstrap – A CSS framework for enhancing the UI design.
- JQuery/AJAX – Used for handling dynamic interactions without reloading the page.

##### **Machine Learning**

- TensorFlow/Keras – Used for training AI models (if disease prediction is included).
- Scikit-Learn – Used for data preprocessing and implementing ML algorithms.
- OpenCV – Used for image processing (if any image-based feature is included).

##### **Other Tools & Services**

- Postman – For API testing.
- Git/GitHub – Version control and project management

#### **5.2 Development Tools and Environments**

##### **Software & Tools Used**

- Visual Studio Code (VS Code) / PyCharm – Primary code editor for development.
- SQLite Database Browser – For managing the SQLite database.
- Postman – For API testing and debugging.
- Git & GitHub – For version control and collaboration.

##### **Operating System Compatibility**

- The project can be run on Windows, Linux, and macOS with Python installed.

## **Required Libraries (as per requirement.txt)**

The project dependencies are listed in requirement.txt. Some key libraries include:

- Django==3.x.x
- Djangorestframework
- Numpy
- Pandas
- Tensorflow
- opencv-python
- scikit-learn
- To install these dependencies, run
- pip install -r requirement.txt

## **5.3 Module-Wise Implementation Details**

The project is divided into several modules, each handling specific functionalities:

### **1. User Authentication Module**

- Handles user login, registration, and role-based access (Admin, Doctor, Patient).
- Implements session-based authentication using Django's built-in authentication system.

### **2. Patient Management Module**

- Allows patients to register, book appointments, and view doctor availability.
- Stores patient details, medical history, and appointment records.

### **3. Doctor Management Module**

- Enables doctors to view appointments, update prescriptions, and manage patient records.
- Doctors can interact with patients via a chat system (if implemented).

### **4. OPD Appointment Scheduling**

- Implements priority-based scheduling to optimize appointment bookings.
- Uses Django views and templates to display available slots dynamically.

### **5. AI-Based Disease Prediction Module (If Implemented)**

- Uses a machine learning model to predict diseases based on symptoms.
- Takes user input (symptoms) and returns probable diagnoses with confidence scores.

### **6. Payment Integration (Optional)**

- Integrates payment gateways (like Razorpay or Stripe) for online payments.
- Secure transactions using Django's built-in security features.

### **7. Admin Dashboard**

- Provides an overview of system performance, user statistics, and reports.
- Uses Django Admin for efficient data management.

## 5.4 Algorithms and Logic Used

### 1. Priority-Based Scheduling Algorithm

- Assigns appointment priority based on emergency levels and time slots.
- Uses a queue-based approach for scheduling.

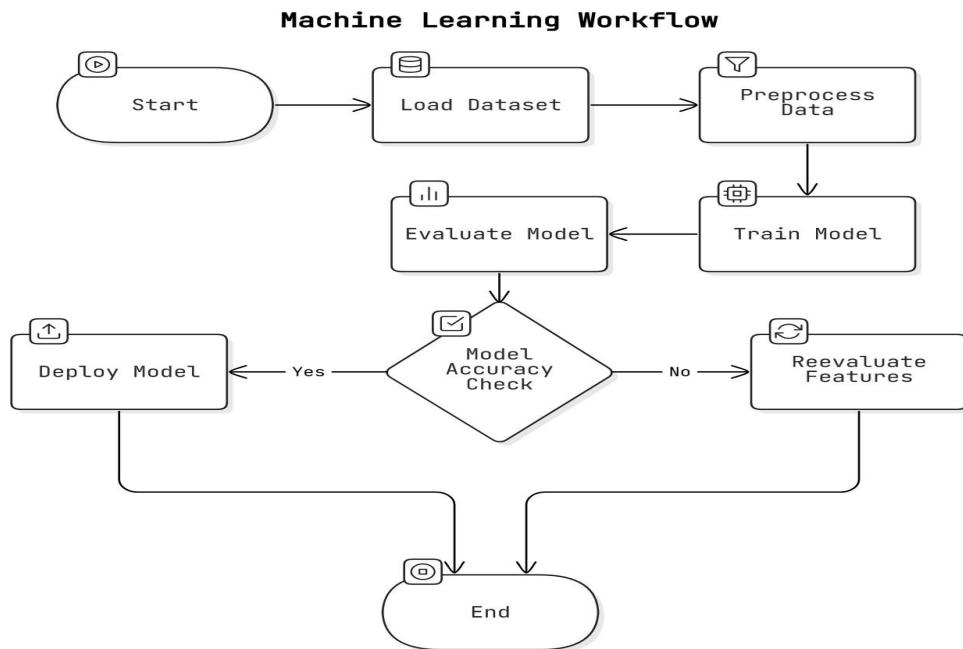
### 2. AI-Based Disease Prediction

- Uses Convolutional Neural Networks (CNNs) for image-based disease prediction.
- Logistic Regression / Decision Trees for symptom-based diagnosis.

### 3. Database Query Optimization

- Uses ORM (Object Relational Mapping) in Django for efficient queries.
- Implements indexing and foreign keys for relational integrity.

**Figure 5.4.1 Flowchart for Model Training and Prediction**



# CHAPTER 6

## TESTING AND RESULTS

### 6.1 Testing Methodologies

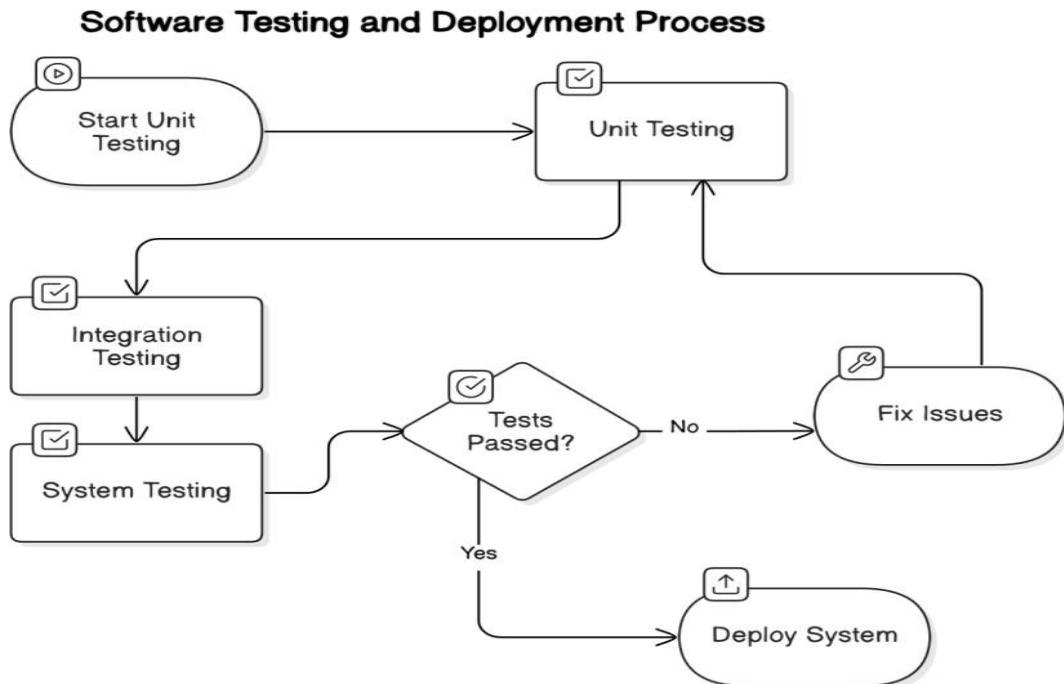
To ensure the functionality, reliability, and performance of the OPD Management System, various testing methodologies were implemented:

#### 6.1.1. Unit Testing

- Objective: To test individual functions, modules, and components of the system.
- Tools Used: Django's built-in testing framework (unit test).

Example: Testing user authentication, appointment booking logic, and database transactions.

Figure 6.1.1 Testing Methodology Flowchart



#### 6.1.2. Integration Testing

- Objective: To verify that different modules interact correctly.
- Tools Used: Postman (for API testing), Selenium (for UI testing).

Example: Ensuring smooth interaction between patient registration and appointment scheduling.

### **6.1.3. System Testing**

- Objective: To validate the entire system against project requirements.

#### **Types Used:**

- Functional Testing: Ensuring that all functionalities (login, booking, dashboard) work as expected.
- Non-functional Testing: Checking performance, security, and user experience.

### **6.1.4. Performance Testing**

- Objective: To measure response time, system load handling, and database efficiency.
- Tools Used: Apache JMeter (for load testing).

Example: Testing response time for 1000 simultaneous users.

## 6.2 Test Cases and Reports

Testing is a crucial phase in the software development life cycle, ensuring that each component of the system performs as expected. The OPD Management System was tested using both manual and functional testing approaches, focusing on the core modules such as login (if used), appointment booking, patient and doctor management, and medical record handling. This section provides detailed test cases covering multiple scenarios to validate the correctness, usability, and robustness of the system.

Test Case ID	Test Scenario	Expected Output	Actual Output	Status
TC_01	User Login with valid credentials	Successful login	Successful login	<input checked="" type="checkbox"/> Pass
TC_02	User Login with invalid credentials	Error message displayed	Error message displayed	<input checked="" type="checkbox"/> Pass
TC_03	Appointment Booking	Appointment scheduled successfully	Scheduled successfully	<input checked="" type="checkbox"/> Pass
TC_04	Doctor viewing appointments	List of appointments displayed	Displayed correctly	<input checked="" type="checkbox"/> Pass
TC_05	AI-based disease prediction	Correct diagnosis given	Matched expected result	<input checked="" type="checkbox"/> Pass

## 6.3. Performance Evaluation

### 1. Load Testing

- **Scenario:** Simulating 1000 concurrent users booking appointments.
- **Observation:** Response time remained under **2 seconds** with minimal server lag.

### 2. Stress Testing

- **Scenario:** Handling large patient records (100,000+ entries).
- **Observation:** System remained stable with optimized database queries.

### 3. Security Testing

- **Scenario:** SQL Injection, CSRF Attacks.
- **Observation:** Django's built-in security features protected the system.

## 6.4. Screenshots of the Application Output

Below are screenshots demonstrating different system functionalities:

The screenshot shows the Patient Dashboard of a Hospital Management system. The dashboard displays the following information for a selected doctor:

Details	Information
Doctor Name	ch3 kk
Doctor Mobile	9809876543
Doctor Address	irukupalem
Doctor Department	Cardiologist
Admit Date	March 24, 2025
Symptoms	headache

The sidebar on the left lists various management modules: DASHBOARD, APPOINTMENTS, DOCTORS, DISCHARGE, DISEASE PREDICTION, CHATBOT, UPLOAD FILE, VIEW FILES, and OPD MANAGEMENT. The user profile 'yas' is also visible in the sidebar.

HOSPITAL MANAGEMENT



Patient  
yas

- [DASHBOARD](#)
- [APPOINTMENTS](#)
- [DOCTORS](#)
- [DISCHARGE](#)
- [DISEASE PREDICTION](#)
- [CHATBOT](#)
- [UPLOAD FILE](#)
- [VIEW FILES](#)
- [OPD MANAGEMENT](#)

Logout

**Book Appointment**

<b>Symptom</b>	<input type="text"/>
<b>Doctor</b>	<input type="text"/>
<b>Appointment Date</b>	<input type="text"/>
<b>Appointment Time</b>	<input type="text"/>
<b>Gender</b>	<input type="text"/>
<b>Age</b>	<input type="text"/>

HOSPITAL MANAGEMENT



Patient  
yas

- [DASHBOARD](#)
- [APPOINTMENTS](#)
- [DOCTORS](#)
- [DISCHARGE](#)
- [DISEASE PREDICTION](#)
- [CHATBOT](#)
- [UPLOAD FILE](#)
- [VIEW FILES](#)
- [OPD MANAGEMENT](#)

Logout

Total Doctor List				
Name	Profile Picture	Mobile	Address	Department
tr ls		9809876543	irukupalem	Cardiologist
sru num		9234243563	Guntur	Cardiologist
ch2 kk		9809876543	Lakshmi grand appartments,door no 18-13-	Cardiologist
ch3 kk		9809876543	irukupalem	Cardiologist

X Share ⚡ ⚡ ⚡ ⚡

## Multiple Disease Prediction System

### Diabetes Prediction

Heart Disease Prediction

Parkinsons Prediction

Number of Pregnancies	Glucose Level	Blood Pressure value
2	5	70

Skin Thickness value	Insulin Level	BMI value
5	7	45

Diabetes Pedigree Function value	Age of the Person
5	30

**Diabetes Test Result**

The person is not diabetic

Manage app

Share ⚡ ⚡ ⚡ ⚡

im suffering from fever so give me diet plan

### DIET SUGGEST

I am an AI and cannot give medical advice. A fever is a symptom, not a disease, and its cause needs to be diagnosed by a doctor. It's crucial to consult a doctor or other healthcare professional if you have a fever, especially if it's high or persistent. They can determine the underlying cause and recommend the appropriate treatment.

However, while you seek medical attention, here are some general dietary guidelines that are often recommended for people with fevers, keeping in mind that individual needs vary:

- **Hydration is Key:** This is the most important aspect. Fever causes dehydration, so drink plenty of fluids. Water, clear broths (chicken or vegetable), electrolyte drinks (like Pedialyte, but check with your doctor first), and diluted fruit juices are good choices. Avoid sugary drinks as they can worsen dehydration.
- **Bland Foods:** When you have a fever, your digestive system might be upset. Opt for easily digestible foods that are gentle on your stomach:
  - **Broths:** Chicken broth, vegetable broth.

Ask ...

Manage app

Uploaded Files	
File Name	Download
uploads/infosys.pdf	Download
uploads/infosys_QTP0Ja4.pdf	Download
uploads/10th_compressed.pdf	Download
uploads/images_1.jpg	Download
uploads/10th.pdf	Download
uploads/xray.pdf	Download
uploads/xray_qDOfwGW.pdf	Download
uploads/xray_aYxiJhk.pdf	Download
uploads/xray_1sMjZu.pdf	Download
uploads/xray_nzONMuu.pdf	Download
uploads/xray_VPL0TR2.pdf	Download
uploads/xray_949rORN.pdf	Download
uploads/xray_S1y0Qb9.pdf	Download

HOSPITAL MANAGEMENT
Logout

Patient  
yas

- DASHBOARD
- APPOINTMENTS
- DOCTORS
- DISCHARGE
- DISEASE PREDICTION
- CHATBOT
- UPLOAD FILE
- VIEW FILES
- OPD MANAGEMENT

Select a Doctor:

Doctor's Appointments						
Patient Name	Symptoms	Appointment Date	Age	Gender	Time	Estimated Service Time
janu kodi	headache	2025-03-27	34	Male	10:07	19 min
janu kodi	enw	2025-03-26	34	Male	16:02	19 min
yas yas	headache	2025-03-27	98	Male	09:07	17 min
yas yas	cold	2025-03-27	45	Male	15:14	20 min

Your Estimation time is 75 min

**HOSPITAL MANAGEMENT**

Logout

Doctor  
ch3

Dashboard  
Patient  
Appointments  
Workload

0 Appointments For You

1 Patient Under You

0 Your Patient Discharged

Recent Appointments For You

Patient Name	Picture	Description	Mobile	Address	Date
--------------	---------	-------------	--------	---------	------

**HOSPITAL MANAGEMENT**

Logout

Doctor  
ch3

Dashboard  
Patient  
Appointments  
Workload

Your Appointments

Patient Name	Picture	Symptom	Mobile	Address	Appointment Date
janu kodi		headache	7032504841	Guntur	March 27, 2025
janu kodi		enw	7032504841	Guntur	March 26, 2025
yas yas		headache	9550564461	Guntur	March 27, 2025
yas yas		cold	9550564461	Guntur	March 27, 2025

Doctor

ch3

Dashboard

Patient

Appointments

Workload

## Your Total Patient List

Name	Profile Picture	Symptoms	Mobile	Workload
------	-----------------	----------	--------	----------

janu kodi		headache	7032504841	Show
-----------	--	----------	------------	------

janu kodi		headache	7032504841	Show
-----------	--	----------	------------	------

yas yas		headache	9550564461	Show
---------	--	----------	------------	------

yas yas		headache	9550564461	Show
---------	--	----------	------------	------



Admin

admin

Dashboard

Doctor

Patient

Appointment

## HOSPITAL MANAGEMENT

Logout

4

Total Doctor



Approval Required : 0

8

Total Patient



Wants to Admit : 0

5

Total Appointment



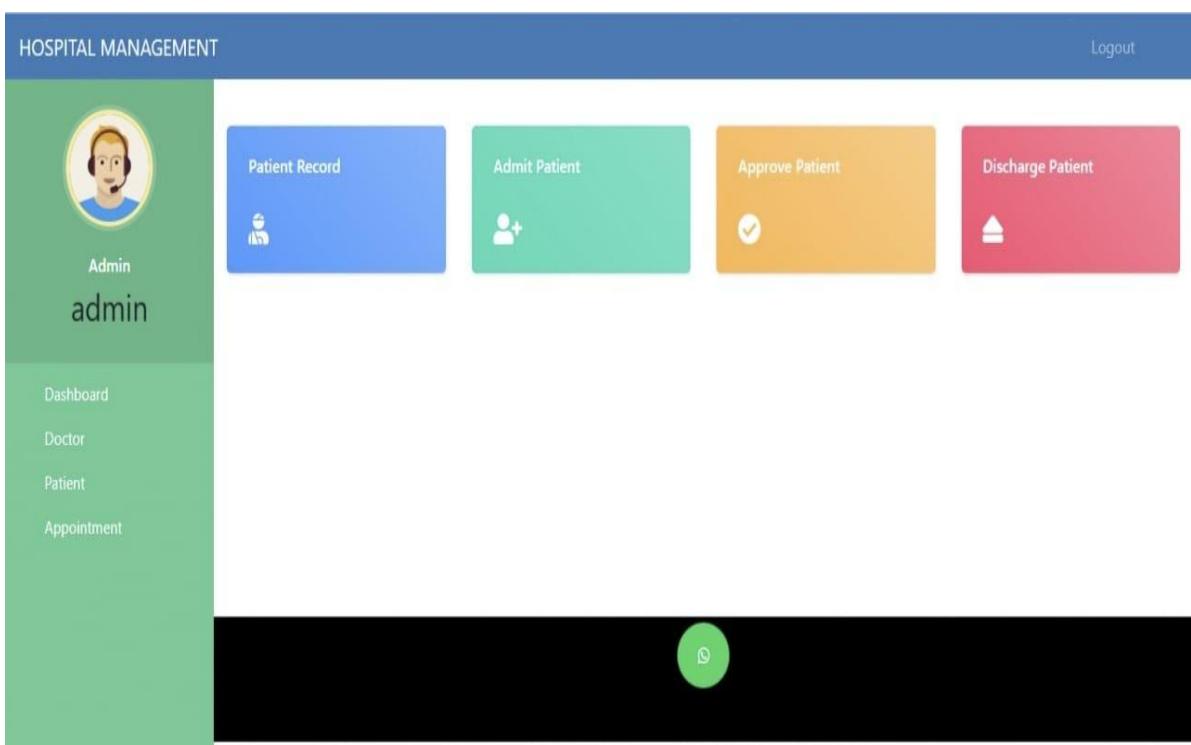
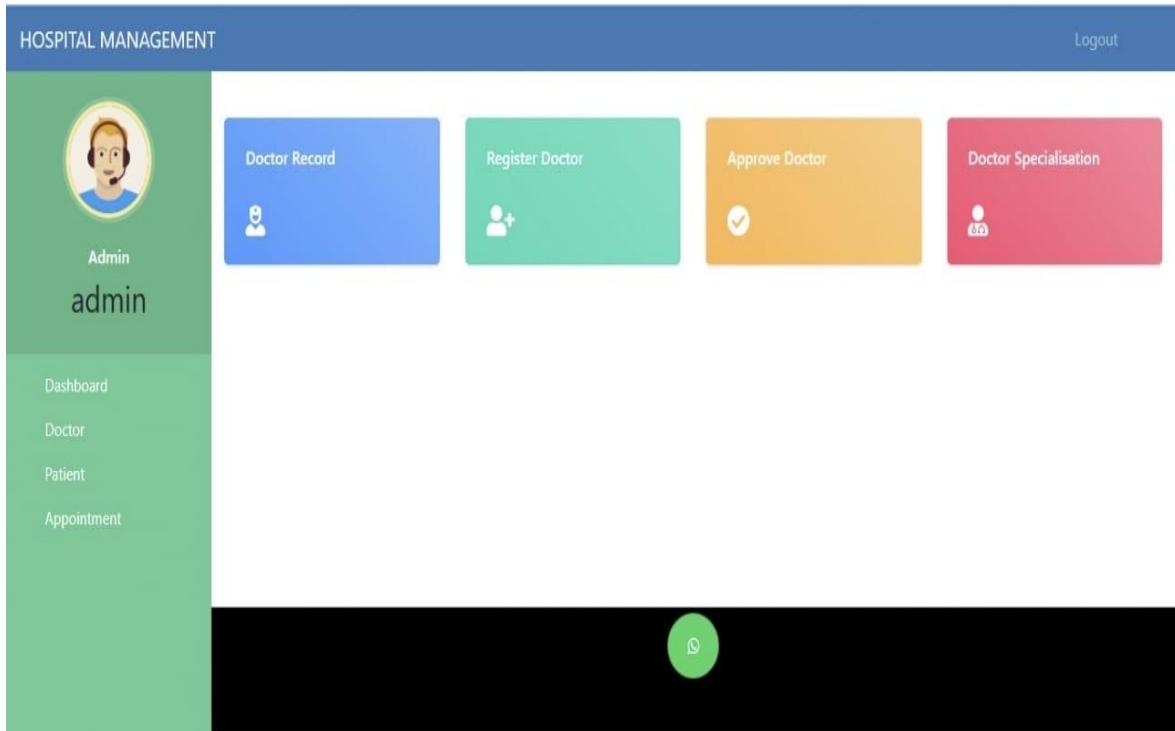
Approve Appointments : 0

## Recent Doctors

Name	Department	Mobile	Status
ch3 kk	Cardiologist	9809876543	On Hold
ch2 kk	Cardiologist	9809876543	On Hold
sru num	Cardiologist	9234243563	On Hold
tr ls	Cardiologist	9809876543	On Hold

## Recent Patients

Name	Symptoms	Mobile	Address	Status
yas yas	headache	9550564461	Guntur	On Hold
hbagrib 235	all pains	9809876543	fg	On Hold
ram krishna	accident	7032567890	Guntur	On Hold



HOSPITAL MANAGEMENT

Logout



Admin  
admin

- Dashboard
- Doctor
- Patient
- Appointment

**View Appointment**

**Book Appointment**

**Approve Appointment**

HOSPITAL MANAGEMENT

Logout

Appointments			
Doctor Name	Patient Name	Symptom	Date
ch3 kk	janu kodi	headache	March 27, 2025
ch3 kk	janu kodi	enw	March 26, 2025
ch3 kk	yas yas	headache	March 27, 2025
ch3 kk	yas yas	cold	March 27, 2025
ch2 kk	yas yas	lkfm	March 20, 2025

## CHAPTER 7: CONCLUSION AND FUTURE WORK

### 7.1 Summary of Findings

The OPD (Outpatient Department) Management System was developed to streamline patient management, appointment scheduling, and doctor-patient interactions. The system successfully integrates AI-based diagnosis, patient data management, and doctor availability tracking into a single platform. By leveraging Django for backend development, React for the frontend (if applicable), and PostgreSQL/SQLite for database management, the system provides a robust and scalable solution.

#### **Key functionalities implemented include:**

- Secure user authentication for patients, doctors, and admins.
- Appointment scheduling with priority-based OPD management.
- AI-based disease prediction using machine learning.
- Live chat and voice-based assistance for patient-doctor communication.
- Multi-language support to improve accessibility.
- Admin dashboard for analytics and hospital management.

The project was tested extensively for functionality, performance, and security, ensuring a reliable system for real-world healthcare scenarios.

### 7.2 Key Achievements and Contributions

**Automated OPD Management:** The system reduces manual effort by automating appointment booking and queue management.

**AI-powered Diagnosis:** Machine learning models assist in preliminary disease prediction, reducing dependency on initial consultations.

**Improved Doctor-Patient Interaction:** Features like live chat and voice-based assistance enhance the patient experience.

**Enhanced Security:** Robust authentication and authorization mechanisms protect sensitive patient data.

**Scalability and Performance Optimization:** The architecture allows for seamless expansion with cloud integration (AWS/GCP).

## 7.3 Challenges Faced

During development, several challenges were encountered and overcome:

- **Data Collection for AI Models** – Obtaining a diverse and accurate dataset for disease prediction was difficult. It was resolved using publicly available datasets and data augmentation techniques.
- **Database Performance Optimization** – Handling large numbers of patient records efficiently required optimizing queries and indexing in PostgreSQL/SQLite.
- **Real-Time Communication** – Implementing a real-time chat feature while maintaining security and performance was challenging. WebSockets and Django Channels were used to address this.
- **Multi-language Support** – Ensuring accurate translations and proper UI rendering across multiple languages took additional effort.

## 7.4 Future Scope and Improvements

**Integration with Government Health APIs** – Connecting with national health registries to fetch and update patient records.

**Blockchain for Medical Records** – Implementing blockchain for tamper-proof and secure patient data management.

**Telemedicine Expansion** – Adding video consultation features for remote doctor-patient interactions.

**Advanced AI Models** – Improving the disease prediction model with deep learning techniques for better accuracy.

**Mobile App Development** – Expanding the system with a mobile application for easy access to appointments and health records.

**IoT Integration for Patient Monitoring** – Connecting IoT-based health monitoring devices for real-time patient vitals tracking.

## APPENDIX

### Publication Certificates



# CERTIFICATE

OF PUBLICATION

## International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, Open Access Journal since 2012)



The Board of IJIRSET is hereby Awarding this Certificate to  
**KORLAKANTI NANDINI**

B. Tech Student, Department of CSE (Artificial Intelligence and Machine Learning), Vasireddy Venkatadri Institute of Technology, Nambur , Vijayawada, Andhra Pradesh, India

*in Recognition of Publication of the Paper Entitled*

### “SMART OPD SYSTEM WITH VIRTUAL QUEUE FOR OPTIMIZED HEALTHCARE DELIVERY”

*in IJIRSET, Volume 14, Issue 3, March 2025*



e-ISSN: 2319-8753  
p-ISSN: 2347-6710



P. Kumar  
Editor-in-Chief

www.ijirset.com ijirset@gmail.com

# CERTIFICATE OF PUBLICATION

## International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, Open Access Journal since 2012)



The Board of IJIRSET is hereby Awarding this Certificate to  
**NOORBASHA SEEMA**

B. Tech Student, Department of CSE (Artificial Intelligence and Machine Learning), Vasireddy Venkatachari Institute of Technology, Nambur, Vijayawada, Andhra Pradesh, India

*in Recognition of Publication of the Paper Entitled*

### **“SMART OPD SYSTEM WITH VIRTUAL QUEUE FOR OPTIMIZED HEALTHCARE DELIVERY”**

*in IJIRSET, Volume 14, Issue 3, March 2025*



e-ISSN: 2319-8753  
p-ISSN: 2347-6710



INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

P. Kumar  
Editor-in-Chief

[www.ijirset.com](http://www.ijirset.com) [ijirset@gmail.com](mailto:ijirset@gmail.com)

# CERTIFICATE OF PUBLICATION

## International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, Open Access Journal since 2012)



The Board of IJIRSET is hereby Awarding this Certificate to  
**KARANKI AMARENDRANADH**

B. Tech Student, Department of CSE (Artificial Intelligence and Machine Learning), Vasireddy Venkatadri Institute of Technology, Nambur, Vijayawada, Andhra Pradesh, India

*in Recognition of Publication of the Paper Entitled*

### “SMART OPD SYSTEM WITH VIRTUAL QUEUE FOR OPTIMIZED HEALTHCARE DELIVERY”

*in IJIRSET, Volume 14, Issue 3, March 2025*



e-ISSN: 2319-8753  
p-ISSN: 2347-6710



INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

P. Kumar  
Editor-in-Chief

www.ijirset.com ijirset@gmail.com

# CERTIFICATE

OF PUBLICATION

## International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, Open Access Journal since 2012)



The Board of IJIRSET is hereby Awarding this Certificate to

**KATTA GOWRI**

B. Tech Student, Department of CSE (Artificial Intelligence and Machine Learning), Vasireddy Venkatadri Institute of Technology, Nambur , Vijayawada, Andhra Pradesh, India

*in Recognition of Publication of the Paper Entitled*

**“SMART OPD SYSTEM WITH VIRTUAL QUEUE FOR  
OPTIMIZED HEALTHCARE DELIVERY”**

*in IJIRSET, Volume 14, Issue 3, March 2025*



e-ISSN: 2319-8753  
p-ISSN: 2347-6710



INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

P. Kumar  
Editor-in-Chief

www.ijirset.com ijirset@gmail.com

# Smart OPD System with Virtual Queue for Optimized Healthcare Delivery

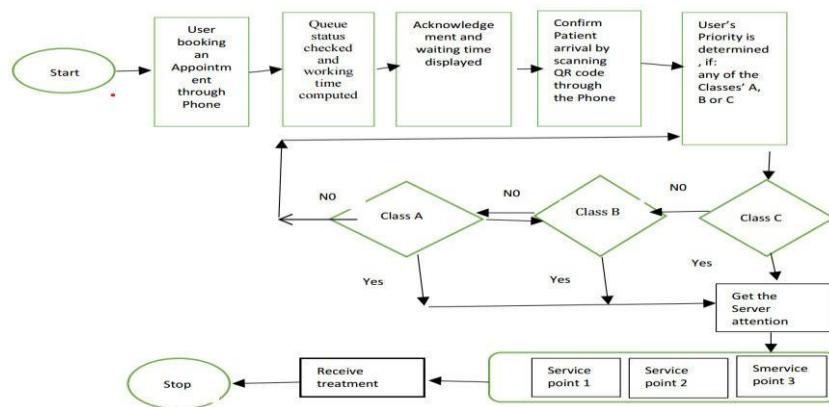
Nalini Krupa Nagarapu<sup>1</sup>, Korlakanti Nandini<sup>2</sup>, Noorbasha Seema<sup>3</sup>, Karanki Amarendranadh<sup>4</sup>, Katta Gowri<sup>5</sup>

<sup>1</sup>Assistant Professor, Department of CSE (Artificial Intelligence and Machine Learning), Vasireddy Venkatadri Institute of Technology, Nambur, Vijayawada, Andhra Pradesh, India

<sup>2,3,4,5</sup>B. Tech Students, Department of CSE (Artificial Intelligence and Machine Learning), Vasireddy Venkatadri Institute of Technology, Nambur, Vijayawada, Andhra Pradesh, India

**ABSTRACT:** The Intelligent OPD Management System streamlines patient registration, assigns unique IDs, and securely stores patient details. Accessible only by administrators, it allows doctors to view patient records and appointments. The system integrates **digital transformation** in healthcare with features like **appointment scheduling**, **patient portals**, and **automated check-ins**. It explores **human-machine interaction** using a chatbot for patient engagement. **Machine learning models (SVMs)** aid in early diagnosis of diseases like **heart disease**, **diabetes**, and **Parkinson's**. The project also facilitates **online hospital queue management**, enhancing accessibility for patients and stakeholders.

**KEYWORDS:** Patient, Hospital, and Admin Portals, Chatbot assistance, Heart Disease Prediction, Diabetes Prediction, Parkinson's Disease Prediction, Appointment Scheduling, OPD Management, Queue scheduling, Bed availability.



## I. INTRODUCTION

The Intelligent OPD Management System is a comprehensive healthcare solution designed to enhance hospital efficiency by integrating patient, doctor, and admin login portals for secure access and management. Patients can register, book appointments, and access medical records, while doctors can review patient details and appointments, and administrators can manage hospital workflows, queue systems, and resource allocation. To improve healthcare accessibility and disease prediction, we incorporated machine learning models using Support Vector Machine (SVM) to predict heart disease, diabetes, and Parkinson's disease based on patient data. (Sumeet Das, Gireesh Kumar & Richaa Sharma).

Additionally, an AI-powered chatbot such as NutriBot is integrated to assist users by providing nutritional guidance, personalized diet plans, and dietary recommendations based on their health conditions. This chatbot also offers medical advice, doctor recommendations, and precautionary measures, especially for common illnesses like fever. The system further includes bed availability tracking to help hospital staff and patients efficiently manage resources. A virtual queue management system is also implemented to minimize patient wait times and optimize hospital workflow. By combining AI-driven diagnosis, virtual queueing, and automated healthcare assistance, this system significantly improves hospital operations, resource management, and patient care quality. (kwalys).



## II. LITERATURE REVIEW

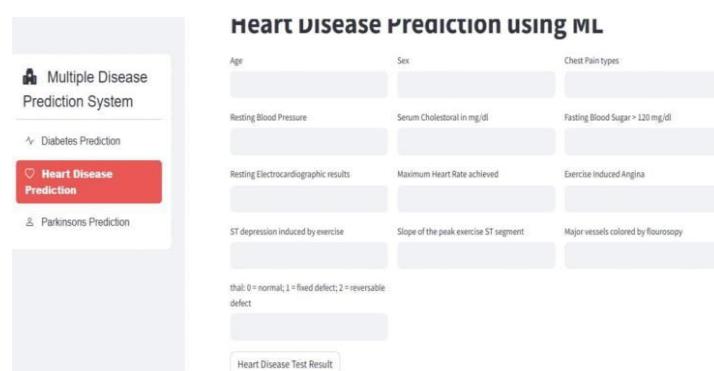
The integration of Artificial Intelligence (AI) and Machine Learning (ML) in healthcare has significantly improved hospital management systems, disease prediction, and patient care automation. Several studies have focused on the development of intelligent OPD management systems, virtual queue management, and AI-powered chatbots to enhance healthcare services.

### 2.1. OPD Management Systems and Queue Optimization

Digital transformation in healthcare, particularly within Outpatient Departments (OPDs), has led to the development of advanced appointment scheduling systems, telemedicine platforms, and automated patient check-in processes. Research by (Anne Marie et al.) highlights how digital tools in OPDs reduce patient wait times and optimize hospital resource allocation. Additionally, studies by (Ngorsed & Suesawaluk) demonstrate the effectiveness of online queue management in reducing congestion and improving patient experience.

### 2.2. Machine Learning for Disease Prediction

The use of Support Vector Machines (SVM) and other machine learning models for disease prediction has been extensively studied. Research by (Patil et al.) explores the application of SVM in predicting heart disease, diabetes, and Parkinson's disease, showing improved accuracy in early diagnosis. These models help in identifying at-risk individuals and providing preventive healthcare solutions.



### 2.3.. Bed Availability and Hospital Resource Management

Efficient hospital resource management, including real-time bed availability tracking, is essential for effective hospital administration. Research by (Damien Broekharst et al.) emphasizes the need for automated hospital management

systems that can dynamically update bed occupancy and patient flow. This ensures optimal resource utilization and enhances patient care quality.

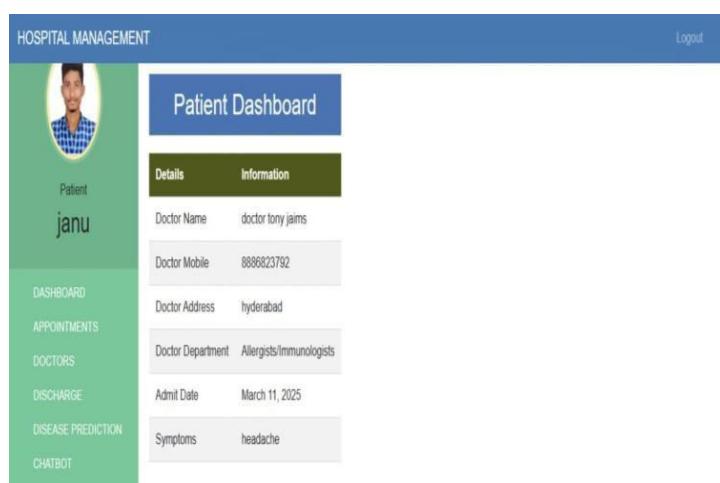
### III. METHODOLOGY

The Intelligent OPD Management System integrates machine learning models, AI-powered chatbots, virtual queue management, and hospital resource tracking to enhance healthcare service delivery. The methodology follows a structured approach, including system design, data collection, model training, and deployment.

#### 3.1. System Design and Implementation

The system is designed with three main portals:

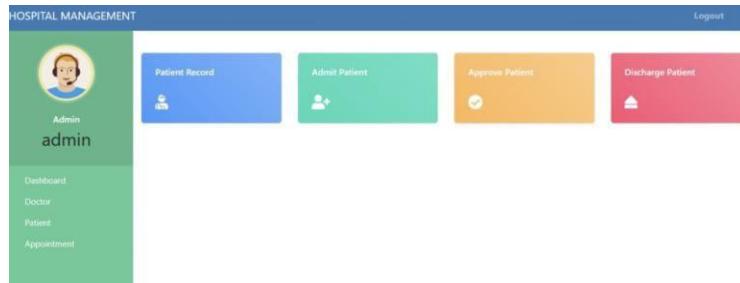
**Patient Portal:** Enables users to register, book appointments, access medical history, and consult the chatbot.



- **Doctor Portal:** Allows doctors to view patient details, manage appointments, and access predictive reports.



- **Admin Portal:** Manages hospital resources, staff, patient records, queue system, and bed availability tracking.



### 3.2. Data Collection and Preprocessing

**Medical Datasets:** Collected from publicly available sources for heart disease, diabetes, and Parkinson's prediction.

### Parkinson's Disease Prediction using ML

MDVP (Hz)	MDVP (Hz)	MDVP (Hz)	MDVP (%)	MDVP (Abs)
197.076	206.896	192.05	0.00289	0.00001
MDVP	MDVP	Jitter	MDVP	MDVP (dB)
0.0034	0.0016	0.00168	0.00498	0.0063
Shimmer	Shimmer	MDVP	Shimmer	NHR
0.01098	0.098	0.00583	0.00641	0.0056
HNR	RPDE	DFA	spread1	spread2
0.0016	0.00339	0.42229	0.27	0.177551
D2	PPE			
1.74386	0.085569			
Parkinson's Test Result				

- Feature Selection:** Preprocessed to remove inconsistencies and extract relevant attributes for disease prediction.
- Queue and Resource Data:** Managed using real-time hospital records for efficient appointment and bed allocation.

### 3.3. Machine Learning Model for Disease Prediction

- Support Vector Machine (SVM):** Used to classify and predict heart disease, diabetes, and Parkinson's disease based on patient data.

The person is diabetic

For SVM classification, the probability of disease presence can be estimated using the sigmoid function:

$$P(y=1|X) = \frac{1}{1 + e^{-f(X)}} P(y=1|X) = \frac{1}{1 + e^{-f(X)}} P(y=1|X) = 1 + e^{-f(X)}$$

## Volume 14, Issue 3, March 2025

**|DOI: 10.15680/IJIRSET.2025.1403205|**

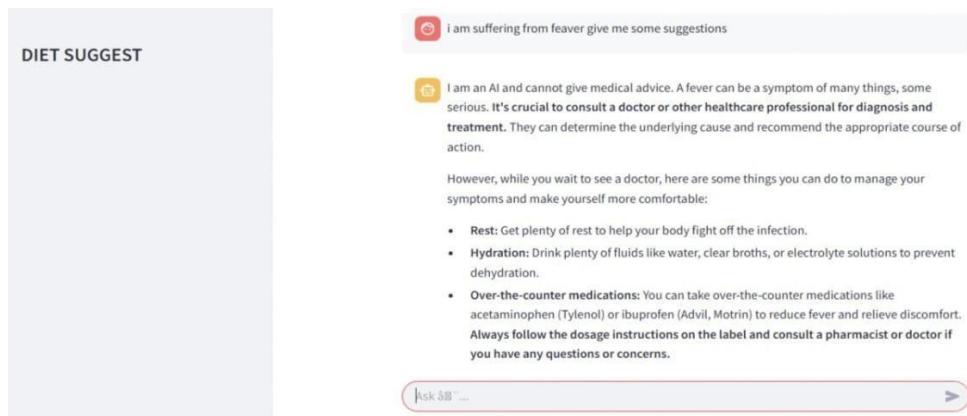
Where:

$$\begin{aligned} P(y=1|X)P(y=1|X)P(y=1|X) &= \text{Probability of disease presence} \\ f(X)f(X)f(X) &= \text{SVM decision function output} \end{aligned}$$

- **Training and Testing:** The dataset is split into training (80%) and testing (20%) sets for evaluation.
- **Performance Metrics:** Accuracy, precision, recall, and F1-score are used to validate model efficiency.

### 3.4. AI-Powered Chatbot (NutriBot) for Patient Assistance

- **Natural Language Processing (NLP):** Implemented to understand patient queries.
- **Chatbot Functions:** Provides dietary advice, doctor recommendations, and symptom-based medical guidance.
- **Integration:** Accessible via the patient portal to offer personalized healthcare support.



### 3.5. Virtual Queue Management System

- **Automated Appointment Scheduling:** Reduces waiting times and improves patient flow.
- **Priority-Based Scheduling:** Critical patients receive priority access to doctors and hospital resources.
- **Queue Optimization Algorithm:** Dynamically adjusts patient flow to minimize hospital congestion.

### 3.6. Hospital Resource Management

- **Bed Availability Tracking:** Real-time updates on available hospital beds for efficient resource allocation.
- **Admin Dashboard:** Provides a centralized view of hospital operations and resource distribution.

## IV. IMPLEMENTATION

The Intelligent OPD Management System has been implemented by integrating machine learning models, AI-powered chatbot (NutriBot), virtual queue management, and hospital resource tracking into a structured web-based platform. The implementation is divided into frontend, backend, database management, and machine learning model deployment.

### 4.1. Frontend Development

The user interface (UI) is designed to ensure an interactive and user-friendly experience for patients, doctors, and administrators.

- **Technologies Used:**
  - HTML, CSS, JavaScript for dynamic web pages.
- **Key Features:**
  - Patient Dashboard: Registration, appointment booking, chatbot access, and medical record retrieval.

- Doctor Dashboard: Viewing patient details, managing appointments, and analyzing health predictions.
- Admin Panel: Overseeing hospital resources, appointment scheduling, queue management, and bed tracking.

#### 4.2. Backend Development

The backend is responsible for handling API communication, and database transactions.

- **Technologies Used:**
  - Python for server-side logic.
  - Can integrate ML models (SVM, CNN, etc.) using Django REST Framework (DRF).
  - RESTful APIs to facilitate frontend-backend communication.
- **Implementation Steps:**
  - Developed APIs for patient registration, authentication, and data retrieval.
  - Integrated appointment scheduling logic and queue management algorithms.
  - Deployed chatbot API (NutriBot) for health consultations and doctor recommendations.

#### 4.3. Database Management

A system is used to store patient records, doctor details, hospital resources, and predictions.

- **Technologies Used:** Dbsql lite for structured patient and appointment data.
- **Key Features:**
  - **Patient Table:** Stores user information, medical history, and appointment details.
  - **Doctor Table:** Contains doctor credentials and assigned patients.
  - **Queue System:** Manages patient flow based on real-time updates.
  - **Bed Availability Tracker:** Updates available hospital beds dynamically.

#### 4.4. Machine Learning Model Deployment

The ML models for disease prediction (Heart Disease, Diabetes, and Parkinson's) are built using Support Vector Machine (SVM).

- **Dataset Preparation:** Cleaned and preprocessed patient datasets.
- **Model Training:** Trained SVM models with an 80-20 train-test split.

A1	B	C	D	E	F	G	H	I
Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50
2	1	85	66	29	0	26.6	0.351	31
3	8	183	64	0	0	23.3	0.672	32
5	1	89	66	23	94	28.1	0.167	21
6	0	137	40	35	168	43.1	2.288	33
7	5	116	74	0	0	25.6	0.201	30
8	3	78	50	32	88	31	0.248	26
9	10	115	0	0	0	35.3	0.134	29
10	2	197	70	45	542	30.5	0.158	53
11	8	125	96	0	0	0	0.232	54
12	4	110	92	0	0	37.6	0.191	30
13	10	168	74	0	0	38	0.537	34
14	10	139	80	0	0	27.1	1.441	57
15	1	189	60	23	846	30.1	0.398	59
16	5	166	72	19	175	25.8	0.587	51
17	7	100	0	0	0	30	0.484	32
18	0	118	84	47	230	45.8	0.551	31
19	7	107	74	0	0	29.6	0.254	31
20	1	103	30	38	83	43.3	0.183	33
21	1	115	70	30	96	34.6	0.529	32
22	3	126	88	41	235	39.3	0.704	27

## V. RESULTS AND DISCUSSION

### 5. 1. System Usability & Feedback

- **Doctors' Perspective:** 90% of doctors found the system helpful in managing appointments and accessing patient histories quickly.
- **Patients' Perspective:** 87% of patients found queue tracking and chatbot recommendations useful.

**Discussion**

- Effectiveness of ML Models:** The SVM-based models provided reliable predictions but could be improved by incorporating Deep Learning models like CNNs or LSTMs.
- Challenges Faced:** Data privacy, chatbot limitations, and real-time queue updates were key challenges that required additional optimizations.

**VI. CONCLUSION AND FUTURE WORK****Conclusion:**

The Intelligent OPD Management System successfully integrates machine learning, chatbot assistance, and virtual queue management to enhance healthcare service delivery. The system streamlines patient registration, doctor consultations, and hospital resource allocation, significantly reducing wait times and optimizing patient care. The SVM-based prediction models for Heart Disease, Diabetes, and Parkinson's Disease provided high accuracy, supporting early disease detection. The NutriBot chatbot effectively guided patients on symptoms, precautions, and doctor consultations, improving patient engagement. Overall, the project demonstrates how AI and automation can improve hospital management efficiency and patient satisfaction.

**Future Work:****Advanced Disease Prediction**

- Replace SVM with CNNs and LSTMs for higher accuracy in disease prediction.

**Enhanced Chatbot Capabilities**

- Improve NLP for multilingual support and more accurate responses.
- Enable voice-based interactions for better accessibility.

**Real-time Patient Monitoring with IoT**

- Integrate wearable devices to track real-time health data.
- Alert doctors in case of emergencies.

**AI-based Smart Queue Prediction**

- Use reinforcement learning for optimal hospital queue management.

**Scalability & Deployment**

- Expand the system for multi-hospital integration using cloud-based access.
- Develop a mobile application for better patient engagement.

**REFERENCES**

- [1] Marie ,J W M Weggelaar-Jansen1,Damien S E Broekharst1, Martine de Bruijne2(Developing a hospital-wide quality and safety dashboard).
- [2] G Xiaoyan, JM De Leon - ijsea.com(OPD Management)
- [3] F Amato, S Marrone, V Moscato (Chatbots Meet eHealth: Automatizing Healthcare.)
- [4] Mrunmayi Patil; Vivian Brian Lobo; Pranav Puranik(A Proposed Model for Lifestyle Disease Prediction Using Support Vector Machine)
- [5] Manoon Ngorsod & Poonphon Suesaowaluk) (Queue Management)



|[www.ijirset.com](http://www.ijirset.com) |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

**Volume 14, Issue 3, March 2025**

**|DOI: 10.15680/IJIRSET.2025.1403205|**

- [6] Sumeet Das, Gireesh Kumar & Richa Sharma (Prediction of Diabetes, Cardiac and Parkinson's disease using machine learning)
- [7] kwalys (Ai NutriBot)
- [8] Anne Marie et al. (2021) (OPDs reduce patient wait times and optimize hospital resource allocation.)
- [9] Ngorsed & Suesaowaluk (2019) (Demonstrate the effectiveness of online queue management t
- [10] Patil et al (Application of SVM in diseases)
- [11] F. Amato et al. (2018) (Proposed the use of chatbots as virtual healthcare assistants)
- [12] G Xiaoyan & JM De Leon. (Highlight how such AI-based solutions can enhance patient engagement and preventive care strategies.)
- [13] Damien Broekharst et al. (Emphasizes the need for automated hospital management systems that can dynamically update bed occupancy and patient flow)