A Mini-Project Report on

HOSPITAL MANAGEMENT SYSTEM

Submitted by

VAISHNAVI (U03NM21T029064)

J. BHAVYA PRIYA (U03NM21T029017)

V SEM, B. TECH (CSE)

Under the guidance of

Dr. Champa H N
Professor,
Department of Computer Science,
UVCE

Department of Computer Science and Engineering

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING
K.R. Circle, Bangalore – 560001

Jun-2024
BANGALORE UNIVERSITY

BANGALORE UNIVERSITY

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K.R. Circle, Bangalore – 560001

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that **VAISHNAVI** of V Semester, B.Tech, Computer Science and Engineering, bearing the register number **U03NM21T029064** has submitted the DBMS Mini-Project Report on **"HOSPITAL MANAGEMENT SYSTEM"**, in partial fulfilment for the DBMS Lab, prescribed by the Bangalore University for the academic year 2023-24.

Mr. Irshad Khan	**Dr. Champa H N**	**Dr. Thriveni J.**
Research Scholar,	Professor,	Professor & Chairperson,
Dept. of CSE,	Dept. of CSE,	Dept. of CSE,
UVCE	UVCE	UVCE

Examiners:

1. …… 2. ...…….

ACKNOWLEDGEMENT

I take this opportunity to thank our institution **University of Visvesvaraya College of Engineering** for having given me an opportunity to carry out this project.

I would like to thank **Dr. Paul Vizhian S., Principal, UVCE**, for providing us all the facilities to work on this project. I am indebted to him for being my pillar of strength and inspiration.

I wish to place my grateful thanks to **Dr. Thriveni J., Professor and Chairperson, Department of Computer Science and Engineering, UVCE,** who helped me to make my project a great success.

I am grateful to **Dr. Champa H N, Professor, Department of Computer Science and Engineering**, **UVCE,** for her valuable suggestions and support, which has sustained me throughout the course of the project.

I am thankful to **Mr. Irshad Khan, Research Scholar, Department of Computer Science and Engineering, UVCE,** for his constant support during the course of the project.

I express my sincere thanks to all teaching and non-teaching staff, Department of Computer Science and Engineering, UVCE for all the facilities that they have provided me for successfully completing this project.

I'm incredibly grateful to my teammate **J.Bhavya Priya** for their exceptional support on our project.

I also thank my parents and friends for their continuous support and encouragement.

VAISHNAVI
(U03NM21T029064)

ABSTRACT

The Hospital Management System (HMS) is a comprehensive software solution designed to streamline hospital operations and administrative functions. This system integrates critical components such as patient management, doctor management, appointment scheduling, discharge details, room management, and departmental administration, all within a unified platform. The HMS features an intuitive interface that allows patients to book, reschedule, or cancel appointments online, reducing the need for in-person visits and minimizing wait times. Doctors benefit from a personalized dashboard where they can access their appointments, manage appointments, and view their patient records. The system effectively handles patient discharge processes by recording discharge summaries. Room management within the HMS tracks room availability, occupancy, and maintenance schedules. Departmental administration is streamlined, facilitating communication between departments. Built on a robust relational database management system (RDBMS), HMS ensures data integrity, security, and efficient processing. It supports seamless data exchange with other healthcare systems and employs role-based access to safeguard sensitive information.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This chapter will discuss the various features and aim of this application.

1.1 Introduction to Hospital Management System

This chapter will introduce the Hospital Management System and how efficiently and coherently the hospital data is saved.

1.2 Objective

The objective of this project is to develop a Hospital Management System (HMS) as a web application that can effectively manage hospital operations. The system aims to simplify the process of tracking patients, doctors, appointments, discharge details, rooms, and departments. Additionally, the system is designed to provide an easy-to-use interface for different user roles, making it simple to input, manage, and retrieve data. The end goal is to create a powerful tool that enhances operational efficiency and provides valuable insights for hospital staff and administrators.

The system will include three user models: ‘Admin’, ‘Doctor’, and ‘Patient’. This structure ensures strong access control over database queries by assigning specific permissions to each type of user.

The user interface aims to provide the following views for users:

- **Admin Dashboard**: Accessible only to ‘Admin’ users, this dashboard allows for the efficient creation, reading, updating, and deletion of any data in the database. Admins can manage staff, assign rooms, oversee departmental functions, and handle patient and doctor records.
- **Doctor Dashboard**: Available to ‘Doctor’ users, this dashboard enables doctors to view their schedules, manage appointments, access patient records, and update medical information. It facilitates efficient patient care and record management.
- **Patient Dashboard**: Designed for ‘Patient’ users, this dashboard allows patients to book, reschedule, or cancel appointments, view their medical

records, and access discharge details. This interface aims to improve patient experience by minimizing the need for in-person visits and reducing wait times.

The system also includes views for:

- **Appointment Management**: Displays all scheduled appointments and allows for easy management of these bookings.
- **Patient Management**: Shows detailed information about patients, including assigned doctor and appointment.
- **Room Management**: Tracks room availability, occupancy, and maintenance schedules.
- **Department Management**: Organizes and manages communication to the department head.

By implementing the HMS, hospitals can achieve significant improvements in managing daily operations, and enhancing patient care. This integrated solution addresses the immediate needs of hospital staff and supports the long-term strategic goals of the healthcare facility.

1.3 Functionality

Admin

- Signup their account. Then Login (No approval Required).
- Can register/view/approve/reject/delete doctor (approve those doctor who applied for job in their hospital).
- Can admit/view/approve/reject/discharge patient (discharge patient when treatment is done).
- Can Generate/Download Invoice pdf (Generate Invoice according to medicine cost, room charge, doctor charge and other charge).
- Can view/book/approve Appointment (approve those appointments which is requested by patient).

Doctor

- Apply for job in hospital. Then Login (Approval required by hospital admin, Then only doctor can login).
- Can only view their patient details (symptoms, name, mobile) assigned to that doctor by admin.
- Can view their discharged(by admin) patient list.
- Can view their Appointments, booked by admin.
- Can delete their Appointment, when doctor attended their appointment.

Patient

- Create account for admit in hospital. Then Login (Approval required by hospital admin, Then only patient can login).
- Can view assigned doctor's details like (specialization, mobile, address).
- Can view their booked appointment status (pending/confirmed by admin).
- Can book appointments. (approval required by admin)
- Can view/download Invoice pdf (Only when that patient is discharged by admin).

Room

- Add the patient the room
- See the room

Department

- List of all department with department heads

1.4 Database Management System

DBMS is a collection of programs that enables users to create and maintain a database The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications. It also provides protection and the security to the database. In case of multiple users, it also maintain the data consistency.

A Relational database is a database that has a collection of tables of data items, all of which is formally described and organized according to the relational model. Data in a single table represents a relation, from which the name of the database type comes.

Relation does not contain the duplicate tuples and the tuples of a relation have no specific order. In typical solutions, tables may have additionally defined relationships with each other. In the relational model, each table schema must identify a column or group of columns, called the primary key, to uniquely identify each row. A relationship can then be established between each row in the table and a row in another table by creating a foreign key, a column or group of columns in one table that points to the primary key of another table.

1.4.1 Characteristics of Database Management Systems

- Self-describing nature.
- Keeps a tight control on data redundancy.
- Enforces user defined rules to ensure that integrity of table data.
- Provides insulation between Programs and data, Data abstraction.
- Supports multiple views of the data.
- Helps sharing of data and Multi-user transaction processing.

1.4.2 Advantages of DBMS

- Controlling the redundancy.
- Restricting unauthorized access.
- Providing persistent storage for program objects.
- Providing storage structures for efficient query processing.
- Providing multiple users interfaces
- Representing complex relationships among data.
- Enforcing integrity constraints.

1.5 PostgreSQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 35 years of active development on the core platform.

PostgreSQL is known for its strict adherence to SQL standards and provides a rich set of features, including advanced data types, window functions, common table expressions (CTEs), and full support for ACID (Atomicity, Consistency, Isolation, Durability) transactions.

PostgreSQL offers a wide range of built-in data types, including arrays, JSON, JSONB (binary JSON), geometric types, and custom user-defined types. This makes PostgreSQL more suitable for handling complex data structures and diverse data types.

Generally known for its reliability, data integrity, and advanced features rather than raw performance. While PostgreSQL's performance has improved significantly over the years, it may not always match the speed of MySQL in certain scenarios.

It is known for its strong and passionate community of developers and users, who contribute to its continuous development and improvement. PostgreSQL also has a thriving ecosystem of extensions, plugins, and tools, including PostGIS for geographic information systems and TimescaleDB for time-series data.

1. CREATE

This command is used to create a table or view by giving it a name and specifying its attributes and constraints. The attributes are specified first, and each attribute is given a name, a data type to specify its domain values, and any attribute constraints such as NOT NULL.

Syntax: CREATE TABLE <TNAME> (ATR1 TYP1 CONST1, ATR2 TYP2 CONST2,…)

2. ALTER

The definition of a base table can be altered by ALTER command which is a Schema Evolution command. The possible ALTER TABLE includes adding or dropping a column (attribute), changing a column definition, and adding or dropping table constraints.

Example: ALTER TABLE STUDENT ADD NAME VARCHAR (12)

3. DROP

If a whole schema is not needed any more, the DROP SCHEMA command can be used. There are two drop behaviour options: CASCADE and RESTRICT.

CASCADE option is used to remove the database schema and all its tables, domains and other elements.

If the RESTRICT option is chosen in place of CASCADE, the schema is dropped only it has no elements in it; otherwise, the DROP command will not be executed.

Syntax: DROP TABLE STUDENT CASCADE

1.5.1 Statements in SQL:

Following are the important statements used in SQL.

1. SELECT - Used to retrieve the information from the relation.
2. INSERT - Used to insert the new values to the relation.
3. DELETE - Used to delete one or more existing tuples from the relation.
4. UPDATE - Used to update already existing values in the relation.

1.5.2 Aggregate Functions in SQL:

Following aggregate functions are provided by the SQL.

1) COUNT - Returns number of tuples.
2) SUM - Returns sum of entries in a column.
3) MAX - Returns Maximum value from an entire column.
4) MIN - Returns Minimum value from an entire column.
5) AVG - Returns Average of all the entries in a column.

1.5.3 Constraints in SQL:

Following constraints are provided by the SQL.

1) NOT NULL - Column should contain some value.
2) PRIMARY KEY - Should not allow duplicate and null values to a column.
3) UNIQUE - Each value of a column should be unique.

1.5.4 Data Types in SQL:

Any object's type of data can be specified using an attribute called SQL Data Type. In SQL, a related data type exists for every column, variable, and expression. When making your tables, you can utilise these data kinds. Based on your needs, you can select a data type for a table column.

Numeric data types:

INT(size) - A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295.

INTEGER(size) - Equal to INT(size).

SMALLINT(size) - A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535.

FLOAT*(*size, d*)* - A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter.

DOUBLE*(*size, d*)* - A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter.

DECIMAL*(*size, d*)* - An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter

Character-String data types:

CHAR(size) - A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1.

VARCHAR(size) - A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535.

CHAR VARYING(n) – similar to VARCHAR(size).

Bit-String data types:

BIT(size) - A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1.

BIT VARYING(n) - similar to BIT(size) where n is the maximum number of bits. The default for n ,the length of a character string or bit string is 1.

BINARY(size) - Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1

Boolean data types:

BOOL - Zero is considered as false, nonzero values are considered as true.

BOOLEAN - Equal to BOOL.

Date and Time data types:

DATE - A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'.

DATETIME(fsp) - A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.

TIME(fsp) - A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'.

TIMESTAMP(fsp) - TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss.

.

CHAPTER 2

LITERATURE REVIEW

This chapter focuses on the already existing systems for the management of Hospitals and establish the software requirements for the project.

2.1 Survey of existing system

Transitioning from traditional hospital management systems to modern Hospital Management Systems (HMS) using web applications marks a significant leap in efficiency, accessibility, and data management within healthcare facilities. Traditional systems rely on manual processes, such as paper-based files and Excel spreadsheets, leading to inefficiencies, errors, and limited accessibility. In contrast, HMS utilizing web applications automate various processes, including patient registration and appointment scheduling, streamlining workflows and reducing errors. Remote accessibility allows authorized users to securely access patient records from anywhere with internet connectivity, enhancing collaboration and patient care. Robust security measures ensure data protection and compliance with healthcare regulations, addressing concerns about privacy and confidentiality. Scalability and integration capabilities enable hospitals to adapt to evolving needs and seamlessly integrate with other systems, such as Electronic Health Records (EHR) and medical imaging systems. Overall, the transition to HMS using web applications revolutionizes hospital management, improving operational efficiency, patient care, and overall healthcare outcomes.

Disadvantages

- More human power.
- Repetition of same procedure.
- Low security.
- Data redundancy.
- Difficulty to handle.
- Difficulty to update data.
- Record keeping is difficult.

Developed System

In this project, we utilize the Django web framework to develop the back-end of the Hospital Management System, along with PostgreSQL as the database management system. Django is a high-level Python web framework known for its simplicity, scalability, and security features. It follows the model-view-template (MVT) architectural pattern, providing a structured approach to building web applications.

PostgreSQL, on the other hand, serves as the reliable and powerful relational database management system for storing and managing hospital data. With its advanced features, data integrity mechanisms, and support for complex queries, PostgreSQL ensures the efficient and secure storage of critical information within the Hospital Management System.

Together, Django and PostgreSQL form a robust foundation for developing a comprehensive Hospital Management System. Django's built-in features, such as authentication, session management, and URL routing, streamline the development process and enhance the system's security. Meanwhile, PostgreSQL's scalability, performance optimizations, and support for advanced data types empower the system to handle the complexities of hospital operations effectively.

By leveraging Django and PostgreSQL, the Hospital Management System offers a user-friendly and efficient solution for managing various aspects of hospital operations, including doctor management, patient records, appointments, and administrative tasks.

2.2 SOFTWARE REQUIREMENT

Operating System: Windows/Linux/MacOS

Frontend: HTML, CSS, Bootstrap

Backend: Django web-framework (Python), PostgreSQL (database management system)

Web-Server: Django's default server

2.2.1 Frontend:

The front end is an interface between the user and the back end. The front and back ends may be distributed amongst one or more systems. In network computing, *front end* can refer to any hardware that optimizes or protects network traffic. It is called application front-end hardware because it is placed on the network's outward-facing front end or boundary. Network traffic passes through the front-end hardware before entering the network. In compilers, the front end translates a computer programming source code into an intermediate representation, and the back end works with the intermediate representation to produce code in a computer output language.

1. HTML

HTML or **Hypertext Mark-up Language** is the standard mark-up language used to create web pages. HTML is written in the form of HTML elements consisting of *tags* enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent *empty elements* and so are unpaired, for example <img>. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*).

2. CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a mark-up language. While most often used to style web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS

is a cornerstone specification of the web and almost all web pages' use CSS style sheets to describe their presentation.

3. Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

2.2.2 Backend:

1. Django

Django is a high-level Python web framework renowned for its simplicity, scalability, and versatility. It follows the "batteries-included" philosophy, providing developers with a comprehensive set of tools and libraries to build web applications rapidly. Django follows the model-view-template (MVT) architectural pattern, which organizes code into three components: models for data management, views for business logic, and templates for user interface design.

Key features of Django include its built-in authentication system, admin interface, and powerful ORM (Object-Relational Mapping) for database interaction. The framework emphasizes security best practices, such as protection against common web vulnerabilities like SQL injection and cross-site scripting (XSS). Django also offers robust internationalization and localization support, making it suitable for building multilingual applications.

Django's pragmatic design and extensive documentation make it accessible to developers of all skill levels. Its vibrant community actively contributes to the ecosystem by creating reusable apps, plugins, and extensions. Additionally, Django's scalability and performance optimizations make it suitable for handling high-traffic websites and complex web applications.

Overall, Django empowers developers to focus on building feature-rich web applications quickly while adhering to best practices in web development. Its flexibility, reliability, and ecosystem of libraries make it a popular choice for startups, large enterprises, and developers worldwide.

2. PostgreSQL

PostgreSQL is a powerful open-source relational database management system (RDBMS) renowned for its reliability, performance, and advanced features. Developed by a global community of contributors, PostgreSQL is known for its adherence to SQL standards and its extensibility, making it suitable for a wide range of applications, from small-scale projects to large enterprise solutions.

Key features of PostgreSQL include its robust ACID (Atomicity, Consistency, Isolation, Durability) compliance, which ensures data integrity and reliability, even in high-concurrency environments. PostgreSQL offers a wide array of data types, including advanced ones like arrays, JSON, and geometric types, enabling developers to model complex data structures effectively.

PostgreSQL's support for transactions, views, triggers, and stored procedures facilitates sophisticated data manipulation and business logic implementation. Its advanced indexing capabilities and query optimization techniques contribute to efficient data retrieval and processing.

Furthermore, PostgreSQL prioritizes security, offering features such as role-based access control (RBAC), encryption, and authentication mechanisms to protect sensitive data. The database system's extensibility allows developers to create custom functions, data types, and extensions to address specific requirements.

With its strong community support, comprehensive documentation, and active development, PostgreSQL continues to evolve, incorporating new features and enhancements to meet the changing needs of modern applications. As a result, PostgreSQL remains a popular choice for businesses and developers seeking a reliable and feature-rich database solution.

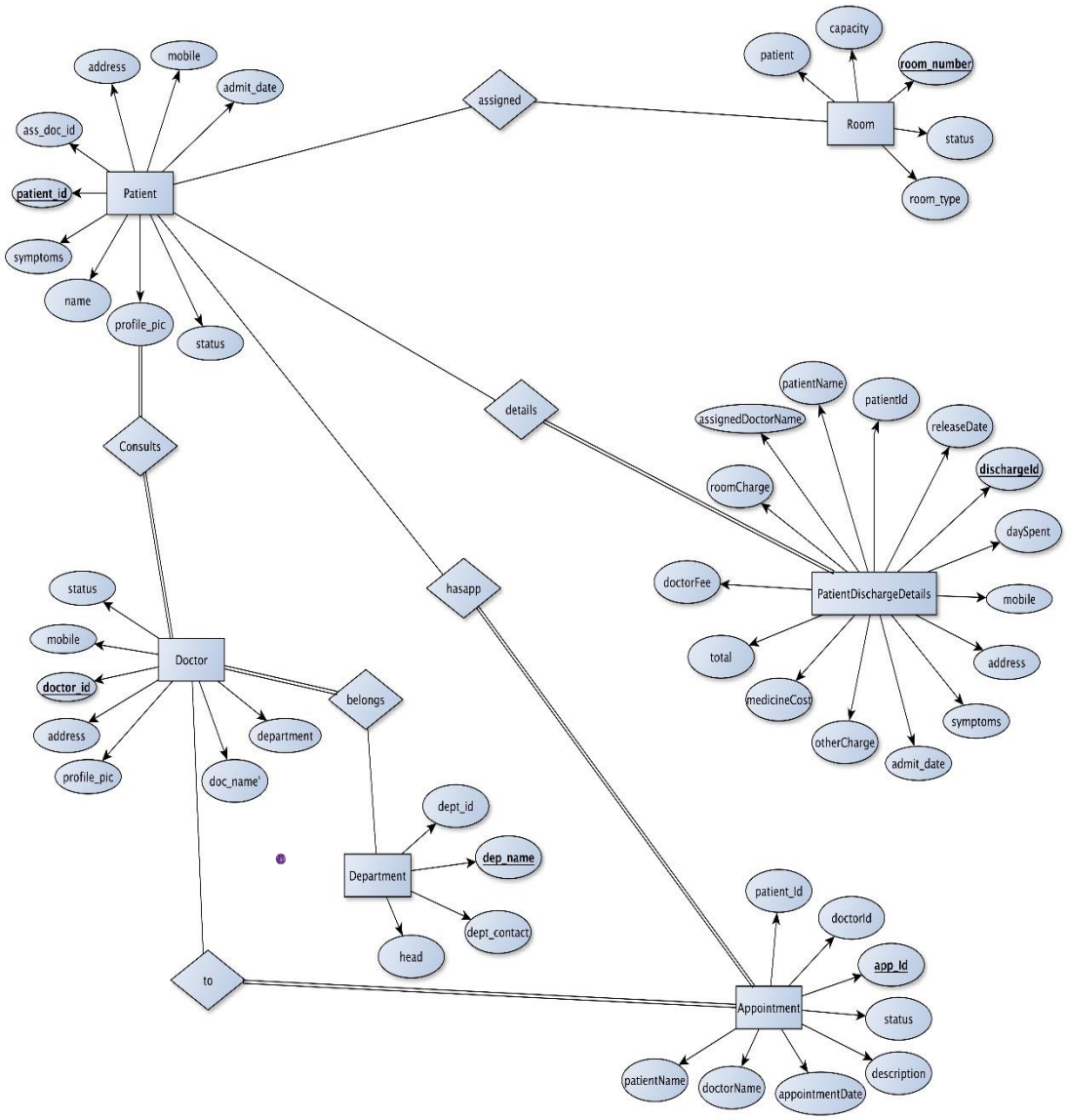
CHAPTER 3

PROPOSED WORK

This chapter will discuss the proposed work for the Hospital management system.

3.1 Entity Relationship (ER) model:

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system's entities and the relationships between those entities.



Entities and their attributes:

In this project, we will be using several entities to maintain the Hospitals. These entities include Patient, Doctor, Appointment, Department, PatientDischargeDetails and Room. Each of these entities plays a crucial role in the hospital management process.

Doctor : The primary entity in the Hospital Management System, representing healthcare professionals responsible for diagnosing and treating patients. Each doctor entry includes essential details such as name, specialization, contact information, and appointment availability.

Patient : Represents individuals seeking medical care within the hospital. Patient records include personal information, medical history, current ailments, and book appointment prior. These records facilitate efficient patient management and continuity of care.

Appointment : Records the scheduling of consultations and treatments between patients and doctors. Each appointment entry contains details such as the patient's name, doctor's name, appointment date and time, and reason for the visit.

Department : Represents distinct areas within the hospital specializing in specific medical services or treatments. Department entries include department names, head id, and contact information.

Patient Discharge Details : Records information related to patient discharges from the hospital. Each discharge entry contains details such as discharge date, reason for discharge, post-discharge instructions, and follow-up appointments.

Room : Represents physical spaces within the hospital used for patient care, consultations, and procedures. Room entries include room numbers, descriptions, capacities, and current occupancy status.

All these entities will be stored in the PostgreSQL database and will be connected to each other through relationships. This allows for easy retrieval and analysis of information, as well as the ability to generate schedules, standings, and statistics. By effectively managing these entities, the Hospital Management System will be able to streamline the recording process of all patient , doctor and patient

3.2 Relational model:

The following is the relational model for the proposed management system:

DOCTOR

Doctor						
Doctor_Id	address	Profile_pic	doc_name	mobile	status	department

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
id	integer			✓	✓
address	character varying	40		✓	
mobile	character varying	20			
department	character varying	50		✓	
status	boolean			✓	
user_id	integer			✓	
profile_pic	character varying	100			
full_name	character varying	100		✓	

APPOINTMENT

Appointment							
app_id	doctor_Id	doctorName	appointmentDate	description	status	Patientname	patientId

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
id	integer			✓	✓
patientid	integer				
doctorid	integer				
appointmentdate	date				
description	text			✓	
status	boolean			✓	
doctorname	character varying	40			
patientname	character varying	40			
appointmenttime	time without time zo...				

PATIENT

Patient								
patient_id	symptoms	name	profile_pic	status	ass_doc_id	address	mobile	admit_date

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
id	integer			✓	✓
address	character varying	40		✓	
mobile	character varying	20		✓	
symptoms	character varying	100		✓	
assigneddoctorid	integer				
status	boolean			✓	
user_id	integer			✓	
admitdate	date			✓	
profile_pic	character varying	100			
full_name	character varying	100		✓	

ROOM

Room				
room_number	capacity	room_type	status	patient

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
room_number	character varying	10		✓	✓
room_type	character varying	50		✓	
status	character varying	20		✓	
capacity	integer				
patient_id	integer				

PATIENTDISCHARGEDETAILS

PatientDischargeDetails							
dischargeId	daySpent	mobile	address	symptoms	Admit_date	otherCharge	medicinecost
total	doctorFee	releaseDate	patientId	patientname	assignedDoctorName		roomCharge

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
id	integer			✓	✓
patientid	integer				
patientname	character varying	40		✓	
assigneddoctornar	character varying	40		✓	
address	character varying	40		✓	
mobile	character varying	20			
symptoms	character varying	100			
admitdate	date			✓	

releasedate	date				
dayspent	integer				
roomcharge	integer				
medicinecost	integer				
doctorfee	integer				
othercharge	integer				
total	integer				

DEPARTMENT

Department			
Dep_name	dep_id	dept_contact	head

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
dept_id	character varying	30			
dept_name	character varying	100			
dept_contact	character varying	20			
head_id	integer				

3. 3 Normalization

The following is a normalization report for the Hospital Management System project:

1st Normal Form (1NF): All the entities in the system such as Patient, Doctor, Room,Departments,Appointments and PatientDiscahrgeDetails are in 1NF, as each record has a unique primary key and each field contains atomic (indivisible) values.

2nd Normal Form (2NF): All the entities in the system are in 2NF, as all non-primary key fields are functionally dependent on the primary key. This means that each field in the table is dependent on the primary key and not on any other non-primary key field.

3rd Normal Form (3NF): All the entities in the system are in 3NF, as there are no transitive dependencies between non-primary key fields. This means that all non-primary key fields are directly dependent on the primary key, and there are no

dependencies between non-primary key fields. For example, The Doctor entity is represented in 3NF as follows:

Doctor Table (Primary Key: doctor_id):

- Contains attributes directly related to the doctor entity, such as doctor_id, Name, Specialization and Contact_Number.
- The doctor_id serves as the primary key, uniquely identifying each doctor record.
- Other attributes, such as Name, Specialization, Contact_Number, , are functionally dependent on the Doctor_ID, ensuring data integrity and eliminating redundant information.

Department Table (Primary Key: department_id):

o Represents the departments where doctors are assigned, such as Dermatologists or Cardiologists.

o Contains attributes like department_id (primary key), Department_Name, and Description.

o The department_id serves as the primary key, uniquely identifying each department record.

o The Doctor entity is linked to the Department entity through a foreign key (e.g., Department_ID) in the Doctor table, establishing a one-to-many relationship between doctors and departments.

This structure adheres to 3NF principles by eliminating transitive dependencies and organizing data in separate tables based on their functional dependencies. It ensures data consistency, reduces redundancy, and facilitates efficient querying and maintenance of doctor-related information within the Hospital Management System

CHAPTER 4

RESULT

The Hospital Management System represents a transformative solution in healthcare administration, revolutionizing the way medical facilities operate and deliver care. This comprehensive system offers a centralized platform for managing every aspect of hospital operations, from patient records and appointments to staff(doctor) scheduling and departmental resources.

Through intuitive user interfaces and advanced functionalities, the system empowers healthcare professionals to streamline workflows and enhance patient experiences. With features such as real-time data access, automated appointment scheduling, and secure communication channels, the Hospital Management System promotes efficiency, accuracy, and collaboration across all levels of the healthcare organization.

By leveraging cutting-edge technologies and adhering to industry standards, this system sets a new benchmark for hospital management, ensuring seamless coordination, improved decision-making, and ultimately, better patient outcomes.

4.1 Screenshots

The following is a series of screenshots of the developed application.

- **Figure 5.1:** Admin Dashboard Page
- **Figure 5.2:** Doctor Record Through Admin Page
- **Figure 5.3:** Add Patient page
- **Figure 5.4:** Patient Discharge Report page
- **Figure 5.5:** Appointment report page
- **Figure 5.6:** Room Management page

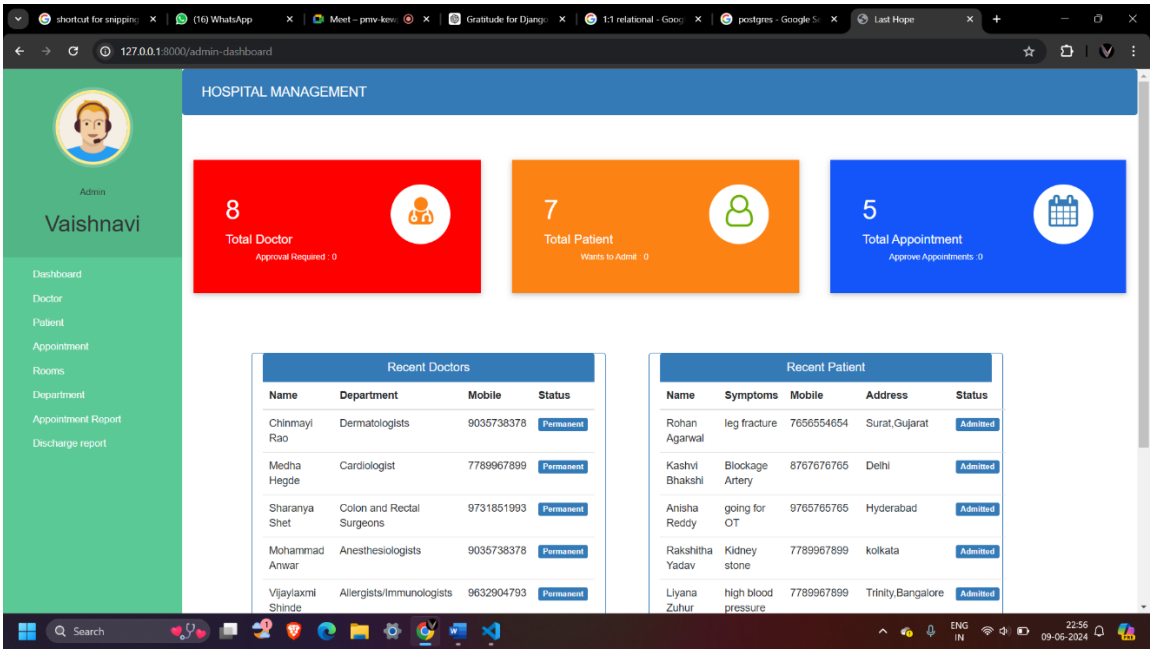


Figure 5.1: Admin Dashboard Page

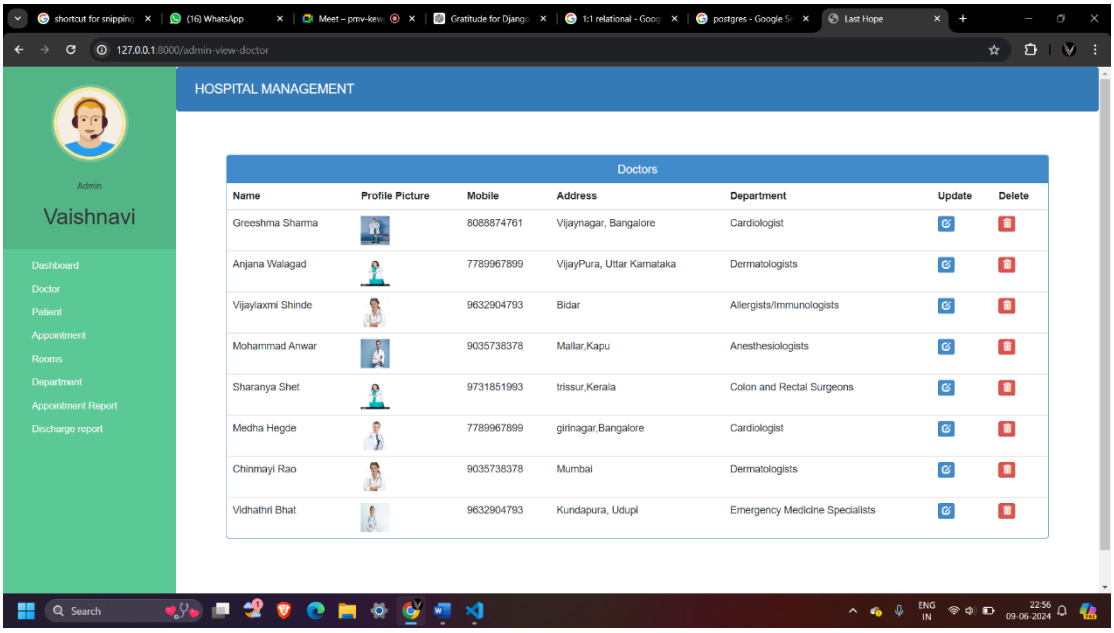


Figure 5.2: Doctor Record Through Admin Page

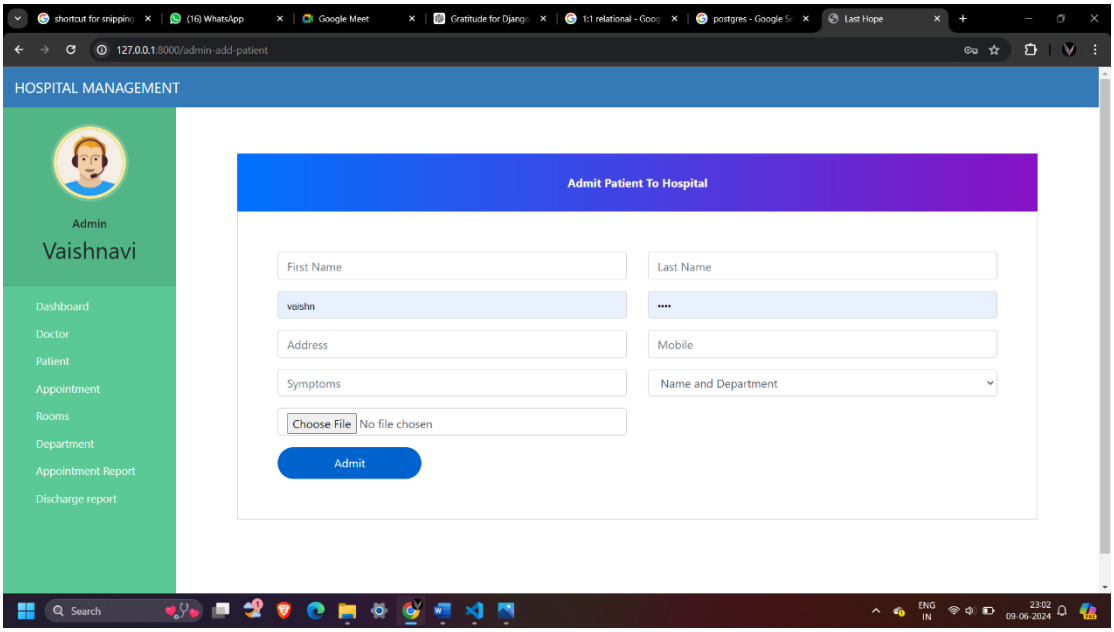


Figure 5.3: Add Patient page

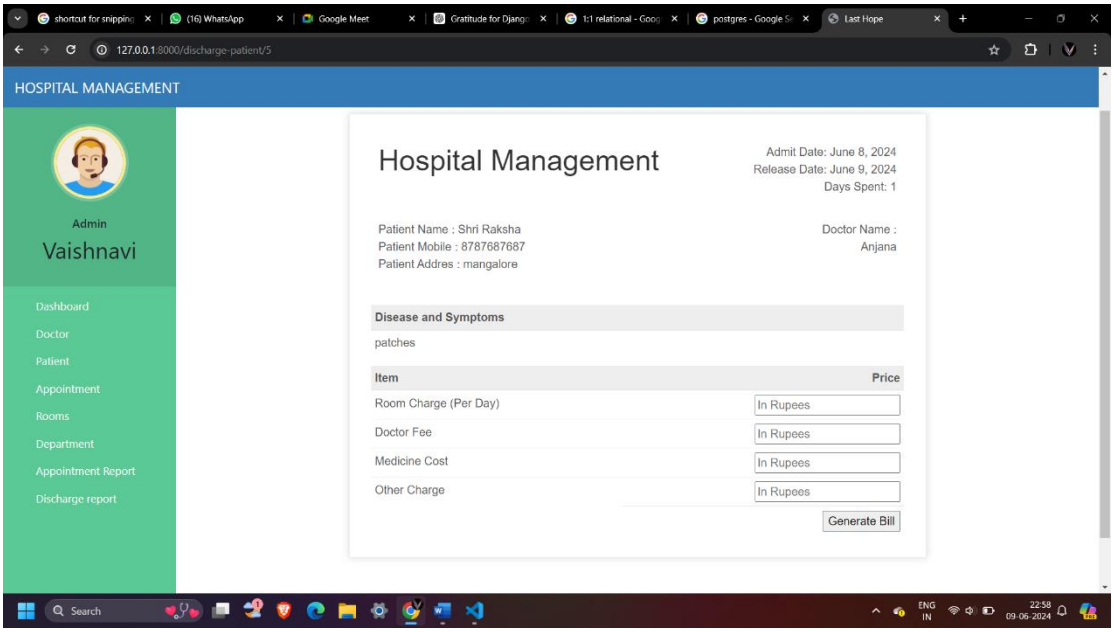


Figure 5.4: Patient Discharge Report

Figure 5.5:View Case page

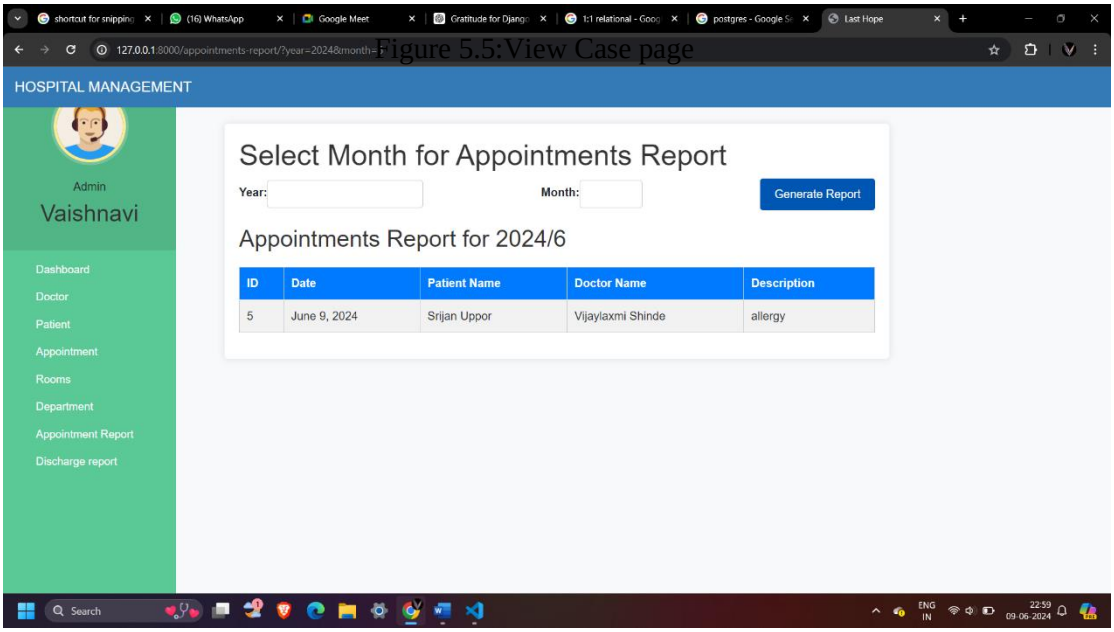


Figure 5.5: Appointment report page

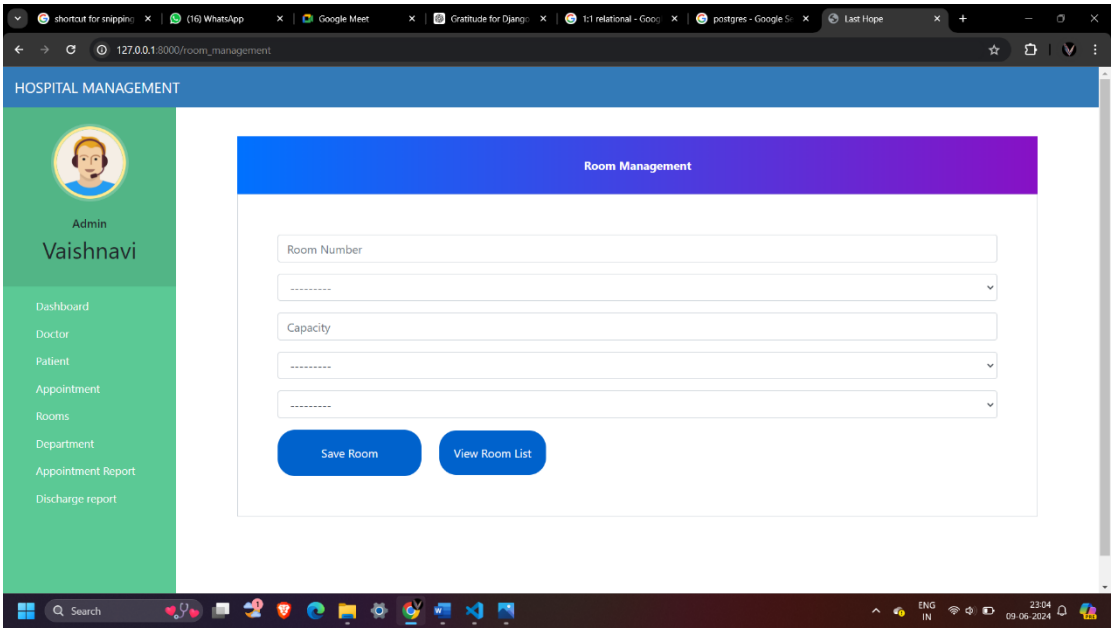


Figure 5.6: Room Management page

Figure 5.1 is the admin dashboard that has all that has all the details, authentication of doctor, patient, room and appointment .

Figure 5.2 is the admin doctor record page which has the record of all the doctor data in the hospital.

Figure 5.3 is the add patient page where the user adds the patient to the hospital record during patient admission.

Figure 5.4 is the patient discharge report which has all the data of the patient ,his symptoms, billing, assigned doctor and generates a pdf to download.

Figure 5.5 is the appointment report page where by entering the month and the year the appointment taken in that month is displayed.

Figure 5.6 is the room management page where the rooms are added.

CONCLUSION

The Hospital Management System stands as a pivotal advancement in healthcare technology, offering a comprehensive solution to streamline and optimize hospital operations. Through the integration of robust features, intuitive interfaces, and advanced functionalities, this system revolutionizes the way healthcare facilities manage patient care, administrative tasks, and resource allocation. By leveraging technologies such as Django and PostgreSQL, the system ensures data integrity, security, and scalability, while also promoting efficiency and collaboration among healthcare professionals. The implementation of this system not only enhances the quality of patient care but also improves overall operational efficiency, reduces administrative burdens, and enables better decision-making. As healthcare continues to evolve, the Hospital Management System serves as a cornerstone for modernizing hospital administration and delivering exceptional healthcare services to patients worldwide.

BIBLIOGRAPHY

[1] Ramez Elmasri and Shamkant B Navathe, *Fundamental of Database Systems,* 6th ed. Addison Wesley, 2009

[2] https://docs.djangoproject.com/en/5.0/

[3] The Hospital Management System paper by K.Nishanthan, S.Mathyvathana, R.Priyanthi , A.Thusara

[4] https://www.postgresql.org/docs/