A Patient Management System

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of
Bachelor in Computer Science & Engineering

by

Parna Basak
CSE 069 08017
&
Fatema Akther Mitu
CSE 069 08024

Supervised by: Samia Sultana
Lecturer

Department of Computer Science and Engineering
STAMFORD UNIVERSITY BANGLADESH

September 2023

Abstract

The Diagnostic Center Management System is a comprehensive software solution designed to streamline and optimize healthcare services in diagnostic centers. This project report presents the development process and key achievements of the system, which includes patient registration, appointment booking, invoice management, and user authentication functionalities. The system utilizes Django, a powerful web framework, and follows a modular architecture for scalability and maintainability. Through a detailed background study, the significance of diagnostic centers in healthcare services and the role of technology in the industry are explored. Lessons learned during development, including the importance of user feedback and continuous testing, are discussed. The report also outlines future enhancements, such as integrating advanced analytics, telemedicine functionalities, and a mobile application. The Diagnostic Center Management System represents a significant step towards improving healthcare operations and patient care, with the potential to positively impact the healthcare industry as a whole.

Approval

The Project Report “A Patient Management System” submitted by Parna Basak ID: CSE 069 08017, Fatema Akther Mitu ID: CSE 069 08024, to the Department of Computer Science & Engineering, Stamford University Bangladesh, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science (Hons) in Computer Science & Engineering and approved as to its style and contents.

Supervisor’s Signature and Date:

.....
Samia Sultana

Date:

Declaration

We, hereby, declare that the work presented in this Project is the outcome of the investigation performed by us under the supervision of Samia Sultana, Lecturer, Department of Computer Science & Engineering, Stamford University Bangladesh. We also declare that no part of this Project or thereof has been or is being submitted elsewhere for the award of any degree or Diploma.

Signature and Date:

.....
Parna Basak
Date:

.....
Fatema Akther Mitu
Date:

Dedicated to our parents, who have always been a source of inspiration and support throughout our academic journey.

Acknowledgments

We sincerely extend our gratitude to all those who have played a significant role in the successful completion of this project.

First and foremost, we would like to express our deepest appreciation to our project supervisor, Samia Sultana, for her exceptional guidance, unwavering support, and invaluable advice throughout the entire project. Her expert knowledge and encouragement were pivotal in our progress and learning.

Furthermore, we are grateful to our friends who offered their assistance and contributed to various aspects of the project. Their insights, ideas, and suggestions were instrumental in shaping the project and enriching its outcomes.

Lastly, we would like to convey our heartfelt thanks to our families for their unconditional support and encouragement throughout this project journey. Their constant love and understanding acted as a constant source of motivation and kept us focused on achieving our goals.

Table of Contents

List of Figures **1**

1: Introduction **2**
1.1 Project Overview . . . 2
1.2 Purpose and Scope . . . 2
1.3 Objectives . . . 3
1.4 Technologies Used . . . 3

2: Background Study **5**
2.1 Introduction . . . 5
2.2 Related Systems and Solutions . . . 5
2.2.1 Hospital Management Systems . . . 5
2.2.2 Electronic Health Records (EHR) Systems . . . 5
2.2.3 Laboratory Information Management Systems (LIMS) . . . 5
2.2.4 Appointment Booking Systems . . . 6
2.2.5 Telemedicine Platforms . . . 6
2.2.6 Mobile Applications for Healthcare . . . 6
2.3 Existing Diagnostic Center Management Systems . . . 6
2.4 Areas for Improvement . . . 7
2.5 Conclusion . . . 8

3: System Design **9**
3.1 Introduction . . . 9
3.2 MVT Architecture . . . 9
3.2.1 Model . . . 9
3.2.2 View . . . 9
3.2.3 Template . . . 10
3.3 Database Schema . . . 10
3.3.1 Patient Model . . . 10
3.3.2 Test Model . . . 10
3.3.3 Invoice Model . . . 11
3.3.4 InvoiceItem Model . . . 11
3.3.5 Payment Model . . . 11
3.3.6 Speciality Model . . . 11

3.3.7 Schedule Model 12
3.3.8 Doctor Model 12
3.3.9 Appointment Model 12
3.3.10 AppointmentPayment Model 12
3.4 Entity-Relationship Diagram (ERD) 13
3.5 Use Case Diagram 13
3.6 System Modules 13
3.6.1 Admin Dashboard 13
3.6.2 Authentication: Login, Registration, Logout 16
3.6.3 Landing Page, Doctor List, Doctor Detail, Appointment Booking . 16
3.6.4 Patient Dashboard: Create Invoice, View Invoice, View Appointments 17
3.6.5 Staff Dashboard: Dashboard, Create Patient, View Patients List, Patient Details, Create Invoice, View Invoice List, Invoice Details, View Appointments 18
3.7 Data Flow Diagram 19
3.8 Conclusion 20

4: Implementation **21**
4.1 Development Environment and Tools 21
4.1.1 Operating System 21
4.1.2 Integrated Development Environment (IDE) 21
4.1.3 Google Chrome Browser 22
4.1.4 Python Packages and Virtual Environment 22
4.2 Technology Stack 22
4.2.1 Python 3.10 22
4.2.2 Django Framework 23
4.2.3 Django Templates 23
4.2.4 SQLite Database 23
4.2.5 HTML 23
4.2.6 Tailwind CSS 23
4.3 Django Project Setup 24
4.3.1 Authentication Pages Screenshots 24
4.3.2 Landing Page Screenshots 25
4.3.3 Admin Dashboard Screenshots 27
4.3.4 Patient Dashboard Features 29
4.3.5 Patient Dashboard Screenshots 29

4.3.6 Staff Dashboard Screenshots 32
4.4 Conclusion 36

5: Conclusion **37**
5.1 Key Achievements 37
5.2 Lessons Learned 38
5.3 Future Enhancements 38
5.4 Conclusion 39

References **40**

List of Figures

3.1 Entity-Relationship Diagram (ERD) for the Diagnostic Center Management System 14
3.2 Use Case Diagram for the Diagnostic Center Management System 15
3.3 Data Flow Diagram for the Diagnostic Center Management System 19

4.1 Login Page 24
4.2 Register 25
4.3 Index Page 25
4.4 Services Page 26
4.5 Doctors List Page 26
4.6 Doctor Detail Page 27
4.7 Admin Dashboard Home Page 27
4.8 Admin Dashboard - Patient Management 28
4.9 Admin Dashboard - Doctor Management 28
4.10 Admin Dashboard - Appointment Management 29
4.11 Admin Dashboard - Invoice Management 29
4.12 Patient Dashboard - Appointment Management 30
4.13 Patient Dashboard - Invoice List 30
4.14 Patient Dashboard - View Invoice Details 31
4.15 Patient Dashboard - Create Invoice 31
4.16 Patient Dashboard - Update Personal Information 32
4.17 Staff Dashboard - Home 32
4.18 Staff Dashboard - Home Services Management 33
4.19 Staff Dashboard - Patient Management 33
4.20 Staff Dashboard - View Patient Details 34
4.21 Staff Dashboard - Create Invoice 34
4.22 Staff Dashboard - View Invoices 35
4.23 Staff Dashboard - View Invoices 35
4.24 Staff Dashboard - Appointment Management 36

1 Introduction

1.1 Project Overview

The Diagnostic Center Management System is a web-based application designed to enhance the management and operations of a diagnostic center. The system aims to empower users by providing them with the flexibility to create their invoices, enabling them to avoid queues and save time during their visits. By offering a variety of features, including home service options and efficient invoice and patient management, the system caters to different users, such as admin, patients and staff, ensuring a seamless experience for all stakeholders.

1.2 Purpose and Scope

The purpose of this project is to develop a comprehensive and user-friendly system that simplifies the management of a diagnostic center. By allowing patients to generate their invoices, the system aims to reduce administrative workload and improve overall efficiency. Additionally, the project aims to provide financial insights through various statistics, enabling administrators to make informed decisions for the center's growth.

The scope of the Diagnostic Center Management System includes the following features:

- User-Created Invoices: Patients can create and manage their invoices, reducing the dependency on manual billing processes.
- Service Options: Users can choose between in-person service at the center or opt for convenient home service.
- User Roles: The system defines three user roles - Admin, Patient, and Staff - with distinct permissions and functionalities.
- Database Management: Data is securely stored in an SQLite database, ensuring seamless data retrieval and management.
- Customized Dashboards: Admin, Staff, and Patients have separate dashboards with unique designs and customized features to meet their specific needs.

- Financial Insights: The system provides revenue tracking for the current month and comparisons with the previous month, along with other relevant statistics.

- Data Security: The system leverages Django's default security offerings to protect user data and ensure confidentiality.

1.3 Objectives

The key objectives of the Diagnostic Center Management System project are as follows:

- To develop a user-friendly web application that simplifies the management of a diagnostic center.

- To empower patients by allowing them to create their invoices and manage their appointments efficiently.

- To provide different service options, including home service, to enhance user convenience.

- Implement role-based access control to ensure secure and appropriate data access for different user categories.

- Provide comprehensive financial information and statistics to aid in decision-making for administrators.

- To use Django[1], Django Templates[2], HTML[3], and Tailwind CSS[4] to develop a robust and visually appealing system.

1.4 Technologies Used

The following technologies and tools were used in the development of the Diagnostic Center Management System:

- Django: A powerful web application framework for backend development.

- Django Templates: Used to render dynamic HTML pages and present data to users.

- HTML: Provides the structure and layout of the web pages.

- Tailwind CSS: Used to style user interfaces, ensuring a visually appealing and responsive design.

2 Background Study

2.1 Introduction

The background study provides an overview of existing systems and solutions related to diagnostic center management and explores the role of technology in healthcare. This chapter aims to identify relevant information and insights that have influenced the development of the Diagnostic Center Management System.

2.2 Related Systems and Solutions

In the rapidly evolving healthcare industry, numerous software solutions have been developed to streamline diagnostic center management. These systems focus on enhancing patient care, optimizing operational efficiency, and improving overall healthcare services. Some of the related systems and their key features are discussed below:

2.2.1 Hospital Management Systems

Hospital management systems are comprehensive software solutions that encompass various aspects of hospital and diagnostic center operations. These systems typically include modules for patient registration, appointment scheduling, inventory management, billing, and reporting.

2.2.2 Electronic Health Records (EHR) Systems

EHR systems digitize and centralize patient health records, making them easily accessible to healthcare providers. These systems enable seamless information sharing among different departments within a healthcare facility, facilitating better coordination and collaboration.

2.2.3 Laboratory Information Management Systems (LIMS)

LIMS is specialized software designed to manage laboratory workflows and data. In diagnostic centers, LIMS handles tasks such as sample tracking, test result management, quality control, and data analysis.

2.2.4 Appointment Booking Systems

Appointment booking systems offer online scheduling of doctor appointments and diagnostic tests. These systems provide patients with the flexibility to choose the appropriate time slots and reduce waiting times, leading to better patient satisfaction.

2.2.5 Telemedicine Platforms

Telemedicine platforms enable remote healthcare services, allowing patients to consult with doctors and receive medical advice from the comfort of their homes. Integrating telemedicine functionalities can expand the reach of diagnostic centers and cater to patients in distant locations.

2.2.6 Mobile Applications for Healthcare

Mobile applications have transformed healthcare services by providing on-the-go access to medical information and services. Developing a mobile application for the Diagnostic Center Management System could offer greater convenience to both patients and staff.

2.3 Existing Diagnostic Center Management Systems

Several commercial and open-source diagnostic center management systems are available in the market, each offering various features to support the day-to-day operations of diagnostic centers. Some notable systems include:

1. **LabCollector**: LabCollector[5] is a comprehensive laboratory management and sample tracking system that supports various types of laboratories, including diagnostic centers. It provides features for sample tracking, inventory management, result reporting, and data visualization. However, LabCollector may lack some specialized functionalities required for healthcare-specific workflows.

2. **ClinicMaster**: ClinicMaster[6] is a cloud-based clinic management system that caters to small and medium-sized healthcare facilities, including diagnostic centers. It offers appointment scheduling, billing, patient records management, and reporting. While ClinicMaster is user-friendly, it may lack advanced features needed for large-scale diagnostic centers.

3. **OpenMRS**: OpenMRS[7] is an open-source electronic health record (EHR) system designed for medical practices and healthcare facilities in resource-limited settings. It provides a flexible and extensible platform for capturing patient data and managing medical records. However, customizing OpenMRS to suit the specific needs of diagnostic centers may require significant development effort.

2.4 Areas for Improvement

Despite its strengths, the Diagnostic Center Management System can still benefit from continuous improvement and enhancement. Some areas worth exploring for further development include:

- **Business Intelligence and Analytics**: Integrating advanced analytics and business intelligence tools can provide diagnostic centers with valuable insights into operational efficiency, resource utilization, and financial performance.
- **Telemedicine Integration**: As telemedicine gains traction, integrating telemedicine functionalities into the system can broaden the scope of services offered by diagnostic centers, enabling remote consultations and follow-ups.
- **Machine Learning for Diagnostic Support**: Leveraging machine learning algorithms for diagnostic support can aid in improving the accuracy and speed of test result analysis, leading to faster and more precise diagnoses.
- **Mobile Application**: Developing a mobile application for the system can enhance patient engagement and accessibility, allowing patients to view test results, manage appointments, and receive notifications conveniently.
- **Data Security and Privacy Compliance**: Strengthening data security measures and ensuring compliance with healthcare privacy regulations is crucial to maintaining patient trust and confidentiality.

By addressing these areas for improvement, the Diagnostic Center Management System can continue to evolve as a cutting-edge solution, empowering diagnostic centers to deliver exceptional healthcare services and stay ahead in the dynamic healthcare industry.

2.5 Conclusion

The Background Study chapter provides an insightful overview of the Diagnostic Center Management System and its relevance in the healthcare industry. Analysis of related systems, such as hospital management systems, EHR systems, and LIMS, showcased various features and limitations. The study highlights the potential for the system to optimize diagnostic center operations and enhance patient care. It also identifies areas for improvement, including telemedicine integration, a dedicated mobile application, and machine learning for diagnostic support. By leveraging these insights, the Diagnostic Center Management System can continue to evolve and offer exceptional healthcare services, staying at the forefront of the industry and positively impacting patient outcomes.

3 System Design

3.1 Introduction

The System Design chapter presents the architectural and functional design of the Diagnostic Center Management System, utilizing the Model-View-Template (MVT) architecture, which is based on Django's Model-View-Controller (MVC) pattern. This section provides an overview of how the MVT architecture is structured and how each component contributes to the system's functionality.

3.2 MVT Architecture

The Model-View-Template (MVT)[8] architecture is a variation of the popular Model-View-Controller (MVC) design pattern, tailored specifically for web applications in Django. MVT emphasizes a clear separation of concerns, making the codebase more organized, maintainable, and scalable. In this section, we delve into the three main components of the MVT architecture: Model, View, and Template, and understand their roles and interactions in the Diagnostic Center Management System.

3.2.1 Model

The Model component represents the data structure and business logic of the system. It is responsible for defining the database schema, handling data retrieval and manipulation, and enforcing data integrity. In the Diagnostic Center Management System, the Model component is implemented using Django's 'models.py' file, where each model corresponds to a database table.

3.2.2 View

The View component manages user interactions, business logic, and the flow of data between the Model and Template components. In the Diagnostic Center Management System, views are implemented as Python functions or classes within the 'views.py' file. They receive user requests, interact with the Model to retrieve or modify data, and return appropriate responses.

3.2.3 Template

The template component is responsible for rendering the user interface, with which users interact when accessing the Diagnostic Center Management System. Templates, implemented using HTML with Django template tags, are stored in the ‘templates‘ directory.

The MVT architecture in the Diagnostic Center Management System ensures that the Model, View, and Template components work in harmony, creating a well-structured, efficient, and user-friendly web application. The clear separation of concerns allows developers to focus on specific components, leading to easier maintenance and scalability as the system evolves and grows.

3.3 Database Schema

The Database Schema of the Diagnostic Center Management System is defined by a set of Django models, representing the entities and their relationships in the system. These models are mapped to database tables, ensuring that data is stored, retrieved, and manipulated efficiently. In this section, we present an overview of the models used in the system and elaborate on their relationships.

3.3.1 Patient Model

The `Patient` model represents individuals registered within the system as patients. It holds essential patient information, including their name, phone number, address, and registration date. The model is associated with Django's built-in `User` model through a one-to-one relationship, allowing for user authentication and authorization. Each patient can have a single associated user account, enabling personalized access to the system.

3.3.2 Test Model

The `Test` model defines the various diagnostic tests offered by the diagnostic center. It includes attributes like the test name, price, and whether home service is available. Each test record in the model represents a unique diagnostic test and its associated details.

3.3.3 Invoice Model

The `Invoice` model is instrumental in managing patient invoices for diagnostic services. It maintains information related to an individual patient's invoice, such as the patient ID (linked to the `Patient` model), the total amount to be paid, the amount already paid, and the invoice creation date. The model also handles the patient's preference for home service and records the date and time for home service appointments.

The `Invoice` model has a foreign key relationship with the `Patient` model, connecting each invoice to a specific patient. It also employs one-to-many relationships with other models, such as `InvoiceItem` and `Payment`, to manage invoice items and payments associated with each invoice.

3.3.4 InvoiceItem Model

The `InvoiceItem` model represents individual items within an invoice. Each item corresponds to a specific diagnostic test (linked to the `Test` model) and includes attributes like status, report file (if available), and date-related details. The status field stores the current status of the test item, indicating whether it is pending, the specimen is collected, or the report is ready.

The `InvoiceItem` model is related to the `Invoice` model through a foreign key, allowing each item to be associated with a specific invoice.

3.3.5 Payment Model

The `Payment` model tracks the payments made by patients for their invoices. It stores details such as the payment amount, payment method, and account information. The model uses a foreign key relationship to connect each payment to a particular invoice through the `Invoice` model.

3.3.6 Speciality Model

The `Speciality` model defines various medical specializations available in the system. Each specialization is represented as a unique record in the model, containing attributes like the name of the specialization and a slug field used for URL identification.

3.3.7 Schedule Model

The `Schedule` model manages the schedules of doctors and their availability for appointments. It includes attributes like the day of the week, start time, and end time for each schedule.

3.3.8 Doctor Model

The `Doctor` model represents medical practitioners available in the diagnostic center. It holds details such as the doctor's name, workplace, degree, contact information, and biographical information. Each doctor can have a specific medical specialization associated with the `Speciality` model, enabling efficient scheduling based on their expertise.

The `Doctor` model utilizes a many-to-many relationship with the `Schedule` model, allowing doctors to have multiple schedules for different days and times.

3.3.9 Appointment Model

The `Appointment` model manages patient appointments with doctors. It includes attributes such as the patient ID, doctor ID, appointment date, time, and appointment status (pending, confirmed, completed, or cancelled). The model also has a field for notes related to the appointment.

The `Appointment` model is related to the `Patient` and `Doctor` models through foreign keys, establishing connections between patients and their scheduled doctors.

3.3.10 AppointmentPayment Model

The `AppointmentPayment` model tracks payments made by patients for their appointments. It stores details such as the payment amount, payment method, and account information.

The relationships among these models create a well-organized and efficient database schema that supports the various functionalities of the Diagnostic Center Management System. The system can handle patient registration, diagnostic tests, invoicing, doctor schedules, appointment management, and payment processing seamlessly.

The use of Django's Object-Relational Mapping (ORM) ensures data integrity, easy database interactions, and simplified querying, making it a robust and scalable solution for managing diagnostic center operations.

3.4 Entity-Relationship Diagram (ERD)

”An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system” as defined in an article[9] by Lucidchart.

3.5 Use Case Diagram

A Use Case Diagram is a graphical representation that depicts the functional requirements and interactions between actors (users or external systems) and the system under consideration. ”In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system’s users (also known as actors) and their interactions with the system” as mentioned in an article[10] by Lucidchart.

3.6 System Modules

The Diagnostic Center Management System is designed to offer a seamless experience to its users while efficiently managing diagnostic center operations. To achieve this, the system is divided into several modules, each catering to specific functionalities. In this section, we will explore and explain each of these modules in detail.

3.6.1 Admin Dashboard

The Admin Dashboard is a powerful and centralized module that provides administrators with complete control over the system’s settings, data, and user management. This module leverages Django’s built-in admin interface[11], which offers a user-friendly way to manage database records and perform CRUD (Create, Read, Update, Delete) operations on various entities. Key features of the Admin Dashboard include:

- **User Management:** Administrators can create, modify and delete user accounts, including patients, staff, and doctors, directly from the administration interface. They can also assign permissions and roles to different user groups to manage access control.

- **Database Management:** The admin interface allows easy management of database records for entities such as tests, doctors, schedules, specializations, etc. Admins can add new records, edit existing ones, and delete obsolete data.

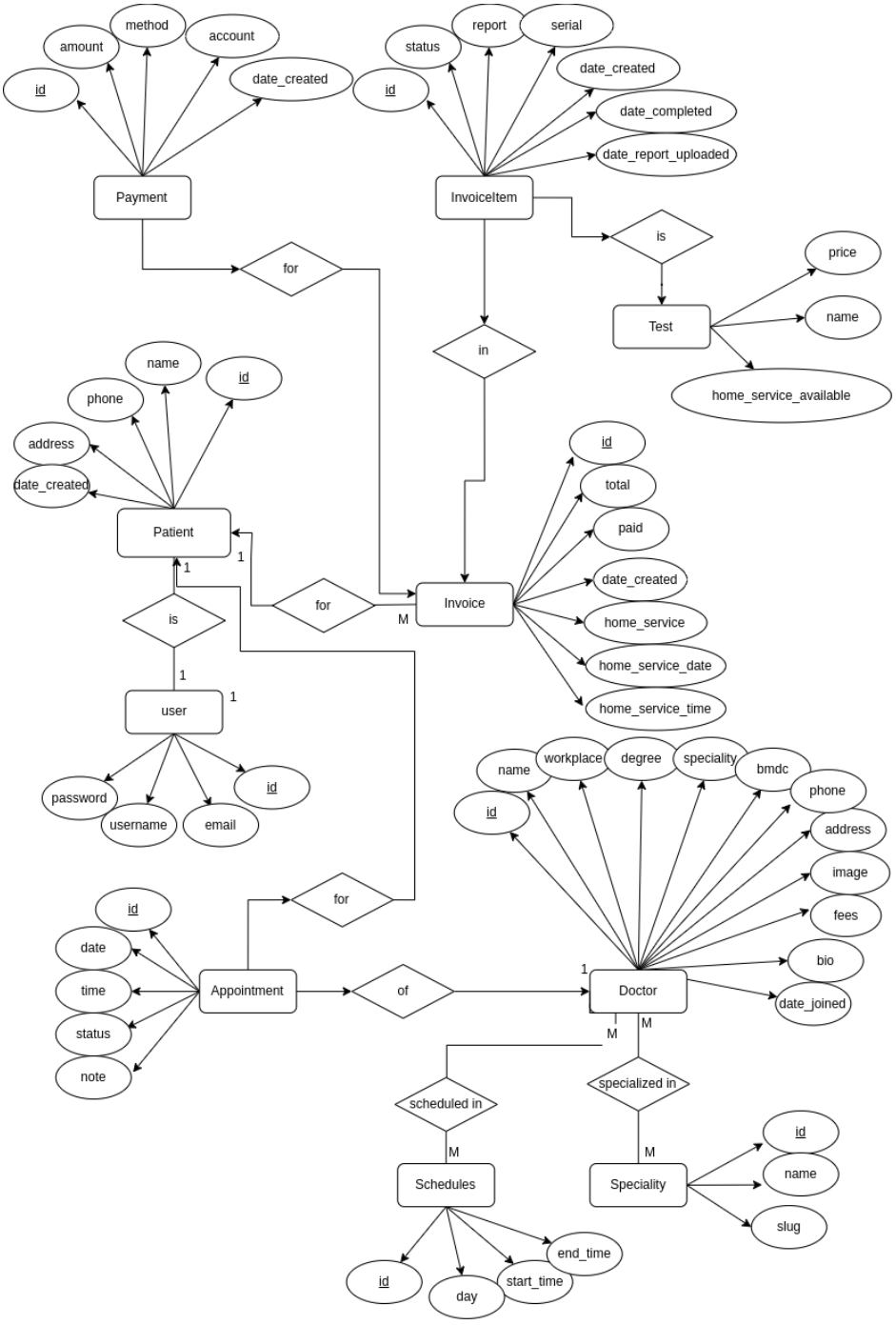

Figure 3.1: Entity-Relationship Diagram (ERD) for the Diagnostic Center Management System

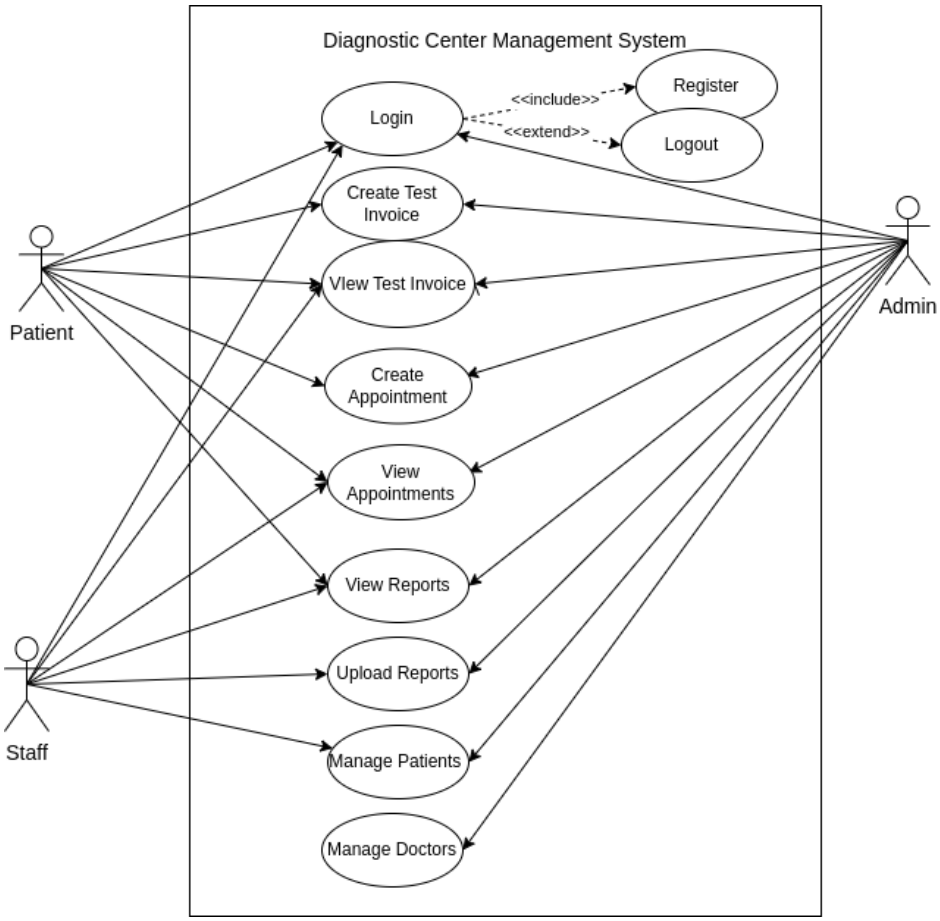

Figure 3.2: Use Case Diagram for the Diagnostic Center Management System

- **Invoice and Appointment Management:** Administrators have access to view and edit all invoices and appointments in the system. They can track payments, mark appointments as confirmed or completed, and perform other necessary actions.

The Admin Dashboard significantly simplifies administrative tasks and ensures that the system administrators have full control over the system's configuration and data management.

3.6.2 Authentication: Login, Registration, Logout

The authentication module handles user authentication and authorization for both patients and staff. It consists of three primary components:

- **User Registration:** This component allows new patients and staff to create accounts in the system. Users provide essential details such as their name, email, phone number, and password to register. The system verifies the uniqueness of the email address and provides appropriate error messages for any conflicts.

- **User Login:** Once registered, users can log in to the system using their email and password. The system authenticates the credentials against the stored data and grants access to the respective user's dashboard based on their role (patient or staff).

- **User Logout:** The system provides a logout functionality that allows users to securely log out of their accounts, ensuring the protection of sensitive data and preventing unauthorized access.

The authentication module plays a critical role in maintaining the security and privacy of the system's users, safeguarding their personal and medical information.

3.6.3 Landing Page, Doctor List, Doctor Detail, Appointment Booking

The landing page serves as an entry point for users visiting the Diagnostic Center Management System. It provides an overview of the system's offerings and directs users to various functionalities. Key components of this module include:

- **Landing Page:** The landing page is designed to be informative and visually appealing. It may include details about the diagnostic center, its services, contact information, and any promotional content to attract new users.

- **Doctor List:** This component displays a complete list of all doctors available at the diagnostic center. Users can browse the list to view basic information about each doctor, such as his name, specialization, and workplace.

- **Doctor Detail:** Clicking on a doctor's name or profile in the doctor list opens the doctor detail page. This page provides in-depth information about the selected doctor, including their degrees, contact details, work schedules, and a brief biography.

- **Appointment Booking:** Users, particularly patients, can schedule appointments with doctors through this module. The appointment booking functionality allows users to select a preferred doctor, choose a suitable date and time, and specify any additional notes or requirements.

The landing page and doctor-related functionalities enhance the user experience by providing relevant information and streamlining the appointment booking process.

3.6.4 Patient Dashboard: Create Invoice, View Invoice, View Appointments

The Patient Dashboard is a personalized module that empowers patients to manage their appointments and financial transactions within the system. Key features of the Patient Dashboard include:

- **Create Invoice:** Patients can create invoices for the diagnostic tests they wish to undergo. The system allows patients to select tests from a predefined list, specify the quantity (if applicable), and opt for home service (if available).

- **View Invoice:** Patients can view their generated invoices, which display details about the tests requested, the total amount due, the amount already paid, and the remaining balance. The view invoice page also shows the current status of the invoice (e.g., pending or paid).

- **View Appointments:** Patients can access a list of all their scheduled appointments through this module. The list includes information about the doctor, appointment date, and status (confirmed, completed, or cancelled).

The Patient Dashboard enhances patient engagement by enabling them to actively manage their diagnostic procedures and stay informed about their upcoming appointments and financial obligations.

3.6.5 Staff Dashboard: Dashboard, Create Patient, View Patients List, Patient Details, Create Invoice, View Invoice List, Invoice Details, View Appointments

The Staff Dashboard is a comprehensive module that empowers the diagnostic center's staff to efficiently manage patient records, appointments, and invoices. Key features of the Staff Dashboard include:

- **Dashboard:** The staff dashboard provides an overview of the diagnostic center's daily activities, including the number of appointments scheduled, the total revenue generated, and the pending tasks.

- **Create Patient:** Staff members can create new patient records by entering essential details, such as the patient's name, contact information, and address. This information is crucial for managing appointments and invoices associated with each patient.

- **View Patients List:** The staff dashboard offers access to a list of all registered patients. Staff members can search for specific patients, view their basic information, and access their detailed profiles.

- **Patient Details:** Clicking on a patient's name or profile in the patients list opens the patient's detail page. This page displays comprehensive information about the patient, including their previous appointments, invoices, and payment history.

- **Create Invoice:** Staff members can generate invoices for patients based on the requested diagnostic tests. They can select tests from the predefined list, specify the quantity (if applicable), and indicate if home service is required.

- **View Invoice List:** The staff dashboard provides access to a list of all generated invoices. Staff members can view details of each invoice, such as the tests included, the total amount, the amount paid, and the due balance.

- **Invoice Details:** Clicking on an invoice in the invoice list opens the invoice details page, where staff members can review the invoice in its entirety, including any attached reports and payment status.

- **View Appointments:** Similar to the patient dashboard, staff members can access a list of all scheduled appointments. This list displays details about the patients, the assigned doctors, and the appointment status.

The Staff Dashboard is a central hub for staff members, providing them with the tools and insights to manage patient records and ensure smooth diagnostic center operations effectively.

Each of these modules plays a critical role in the overall functionality and usability of the Diagnostic Center Management System, providing users with a streamlined and comprehensive experience. These modules work in tandem to improve patient care, simplify administrative tasks, and improve the efficiency of diagnostic center operations.

3.7 Data Flow Diagram

"A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles, and arrows, plus short text labels, to show data inputs, outputs, storage points, and the routes between each destination." as mentioned in an article[12] by Lucidchart.

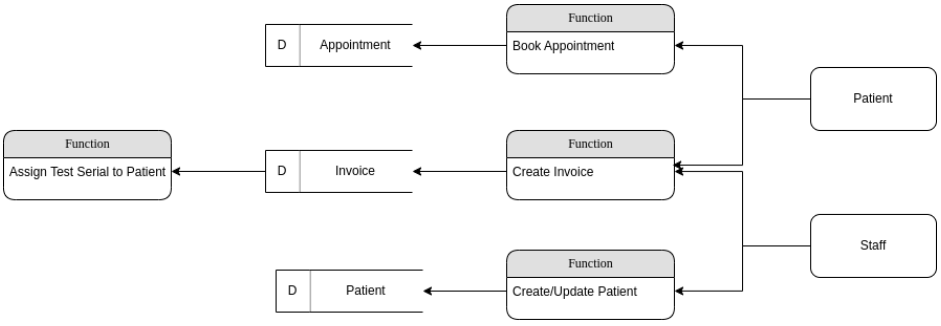

Figure 3.3: Data Flow Diagram for the Diagnostic Center Management System

3.8 Conclusion

The System Design chapter outlines a user-centric approach to the Diagnostic Center Management System, utilizing Django's MVT architecture and a robust database schema. The integration of modules empowers patients and staff with efficient appointment scheduling, invoice management, and administrative control. With a focus on usability and streamlined operations, the system aims to enhance the diagnostic center's services and provide a seamless experience for all users. The ERD and Use Case Diagram visually represent the data model and system functionalities, ensuring a clear understanding of the system's structure. Overall, the well-structured design promises to revolutionize the diagnostic center's operations and improve healthcare services for all stakeholders involved.

4 Implementation

The Implementation chapter delves into the technical aspects of building the Diagnostic Center Management System using the Django framework. This chapter provides a detailed account of the steps taken to transform the system design into a fully functional and user-friendly software solution.

In this chapter, we will discuss the technology stack chosen for the implementation, the database setup, and the Django project's configuration. Furthermore, we will explore the application structure, which ensures modularity and maintainability of the codebase. The integration of Django models with the SQLite database will be explained, as well as the implementation of views and templates for rendering dynamic HTML pages.

4.1 Development Environment and Tools

The development environment plays a crucial role in the efficient and smooth development of the Diagnostic Center Management System. This section provides an overview of the tools and software used during the implementation process.

4.1.1 Operating System

The development of the system was carried out on a Windows 10 operating system. Windows 10 is a widely used operating system known for its user-friendly interface and compatibility with various software and development tools.

4.1.2 Integrated Development Environment (IDE)

Visual Studio Code (VS Code)[13] was the integrated development environment chosen for this project. VS Code is a lightweight and feature-rich code editor developed by Microsoft. It offers excellent support for Python development and provides a wide range of extensions to enhance productivity. With its built-in Git integration and debugging capabilities, VS Code facilitated version control and efficient debugging during the development process.

4.1.3 Google Chrome Browser

During the development process, the Google Chrome browser served as the primary browser for testing the application. Google Chrome's extensive developer tools, such as the Inspector and Console, were utilized for debugging and examining front-end elements and network activities.

4.1.4 Python Packages and Virtual Environment

The project made use of various Python packages to extend Django's capabilities and streamline development. To effectively manage dependencies, a virtual environment was set up using the built-in `venv` module in Python. This ensured the isolation of project-specific packages from system-wide packages, making it easier to manage dependencies and avoid conflicts.

The combination of these tools and technologies created a robust and efficient development environment to build the Diagnostic Center Management System. The compatibility and synergy between these tools greatly contributed to the success of the implementation process.

4.2 Technology Stack

The Technology Stack used in the implementation of the Diagnostic Center Management System comprises a well-rounded combination of frameworks, languages, and tools that collectively contribute to building a robust and user-friendly web application. The primary components of the technology stack include Django, Django Templates, HTML, Tailwind CSS, and SQLite. Each of these elements plays a pivotal role in different aspects of the system's development, ensuring seamless integration and optimal performance.

4.2.1 Python 3.10

Python, being a powerful and versatile programming language, was the language of choice for building the Diagnostic Center Management System. Python 3.10, the latest stable version at the time of development, was used for its enhanced features, improved performance, and better error handling. Python's simplicity and readability allowed for faster and cleaner code implementation.

4.2.2 Django Framework

The system was developed using the Django web framework[1], which is a high-level Python web development framework known for its rapid development capabilities and adherence to the Model-View-Template (MVT) architecture. Django's built-in features, such as the ORM system, URL routing, authentication, and template engine, significantly expedited the development process. The framework's emphasis on security and scalability made it an ideal choice for a healthcare management system.

4.2.3 Django Templates

Django Templates[2] are an integral part of the Django framework, serving as the Template layer in the MVT[8] architecture. They play a crucial role in separating the presentation logic from the application's business logic. Django templates use a syntax that allows developers to embed Python code within HTML, making it easy to dynamically generate content based on data from the model layer.

4.2.4 SQLite Database

For local development and testing purposes, the project utilized the SQLite database engine[14], which is included by default[15] in Django. SQLite is a lightweight and serverless database management system, making it convenient for development and small-scale applications. Its file-based nature allowed for easy setup and configuration without the need for complex server management.

4.2.5 HTML

Hypertext Markup Language (HTML) is the standard markup language used to create the structure and content of web pages. In the Diagnostic Center Management System, HTML templates are utilized in conjunction with Django Templates to define the user interface and layout of each page.

4.2.6 Tailwind CSS

Tailwind CSS[4] is a utility-first CSS framework that helps to create a responsive and visually appealing user interface. Unlike traditional CSS frameworks that come with

predesigned components, Tailwind CSS provides a set of low-level utility classes that developers can use to build custom styles.

The utility-first approach allows for a highly efficient and flexible styling process. Developers can compose styles by combining utility classes directly in the HTML, resulting in a reduction of repetitive CSS code and a smaller overall stylesheet.

Tailwind CSS provides a wide range of utility classes for styling elements such as typography, layout, colors, borders, and spacing. This extensive set of classes facilitates rapid development and customization, ensuring consistency in the system's design.

Furthermore, Tailwind CSS's responsiveness features allow the system to adapt gracefully to different screen sizes, making it accessible across various devices.

4.3 Django Project Setup

The development of the Diagnostic Center Management System was initiated by setting up a Django project. Django provides a command-line tool, `django-admin`, to handle project management tasks.

4.3.1 Authentication Pages Screenshots

Below are some screenshots of the Authentication Pages:

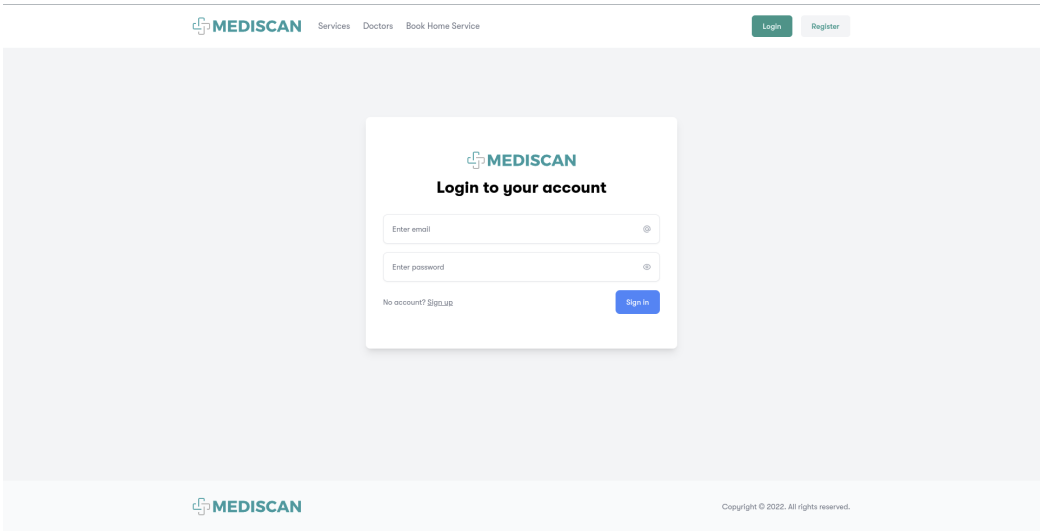

Figure 4.1: Login Page

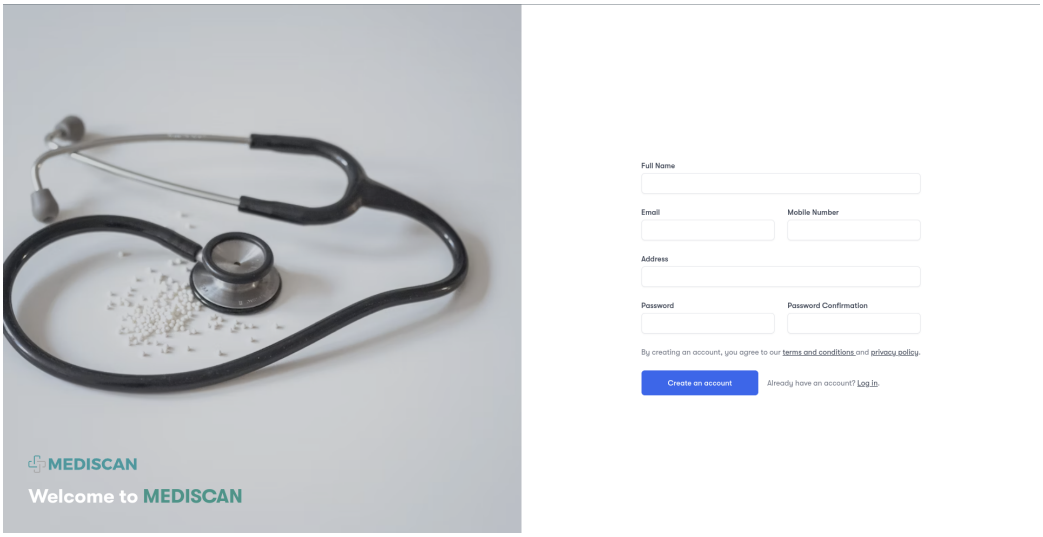

Figure 4.2: Register

4.3.2 Landing Page Screenshots

Below are some screenshots of the Landing Pages:

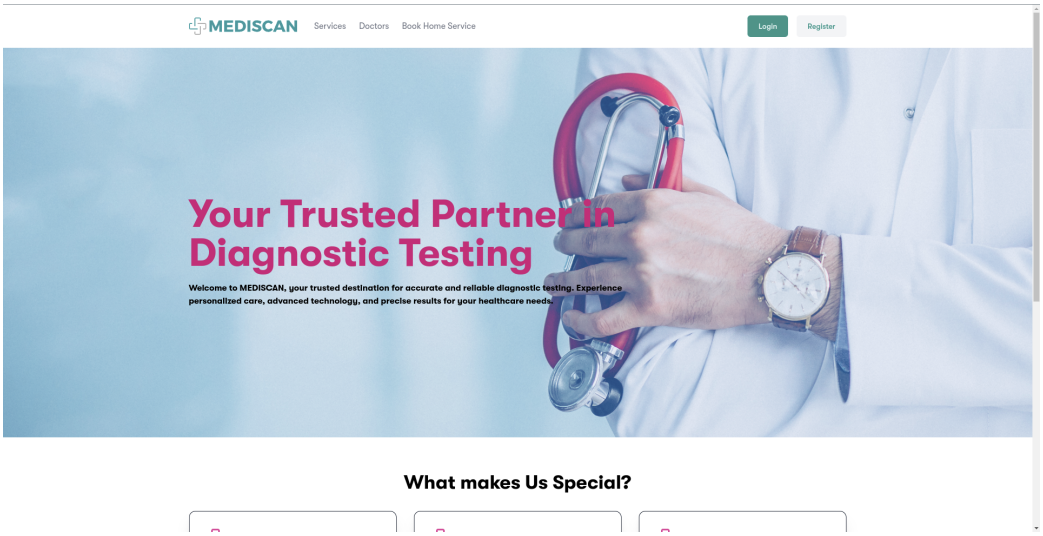

Figure 4.3: Index Page

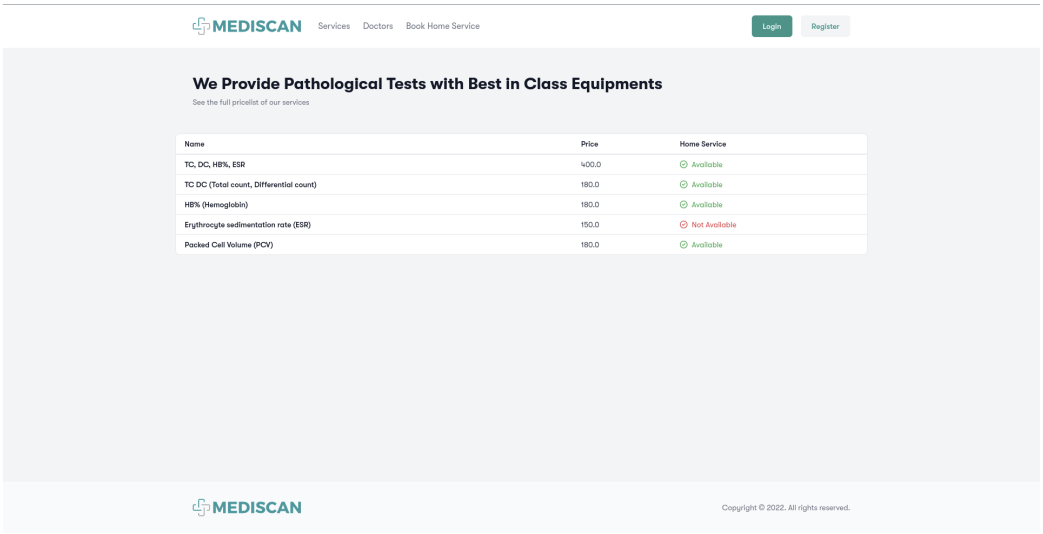

Figure 4.4: Services Page

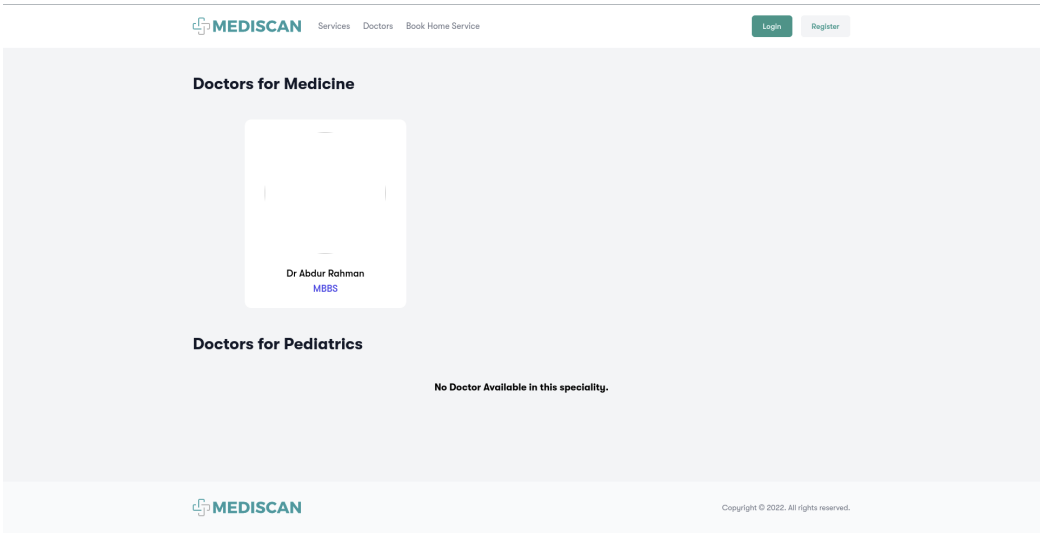

Figure 4.5: Doctors List Page

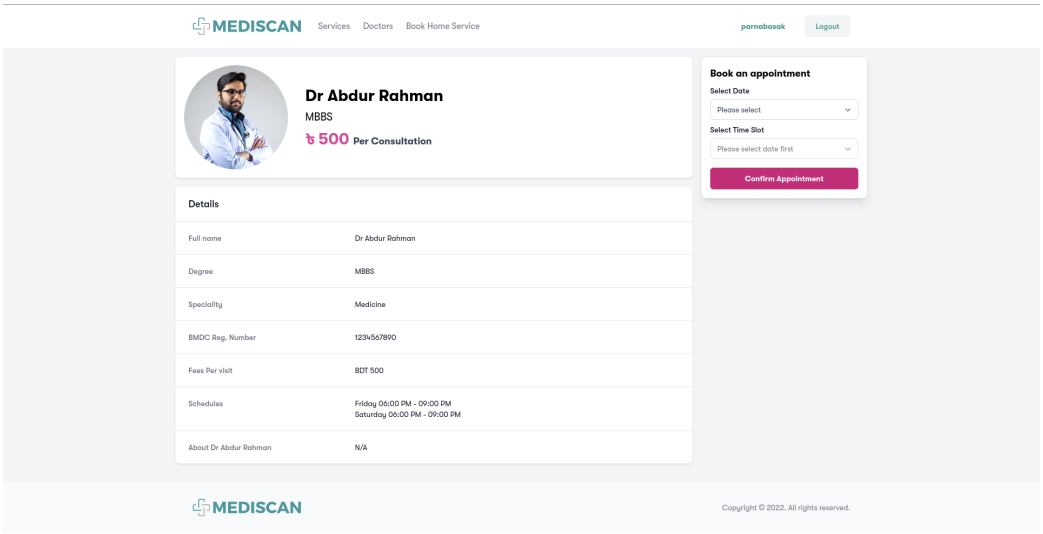

Figure 4.6: Doctor Detail Page

4.3.3 Admin Dashboard Screenshots

Below are some screenshots of the Admin Dashboard[11]:

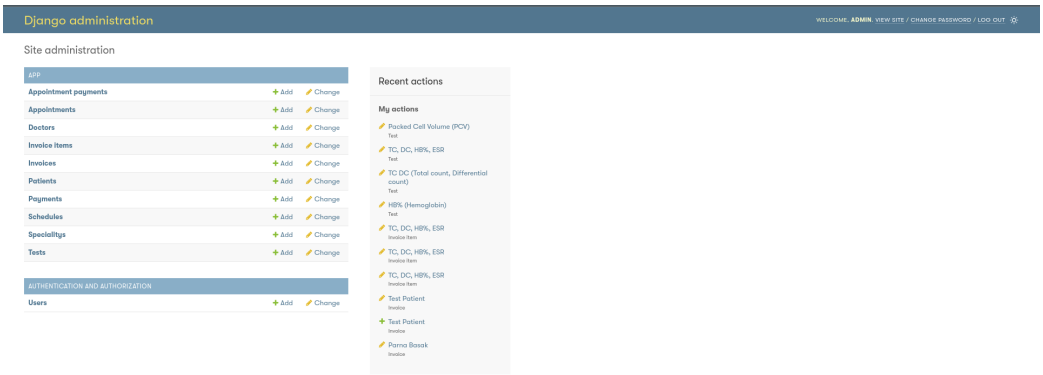

Figure 4.7: Admin Dashboard Home Page

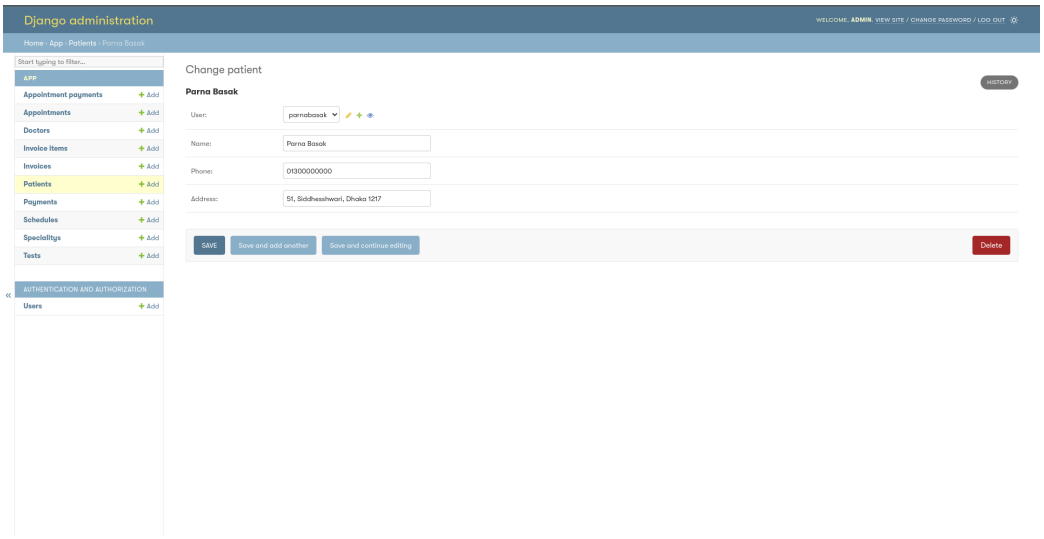

Figure 4.8: Admin Dashboard - Patient Management

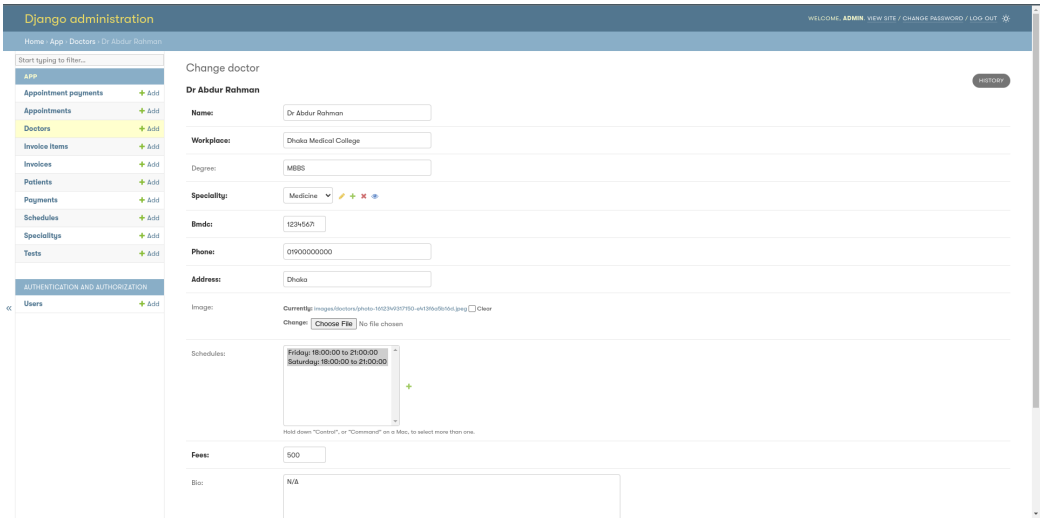

Figure 4.9: Admin Dashboard - Doctor Management

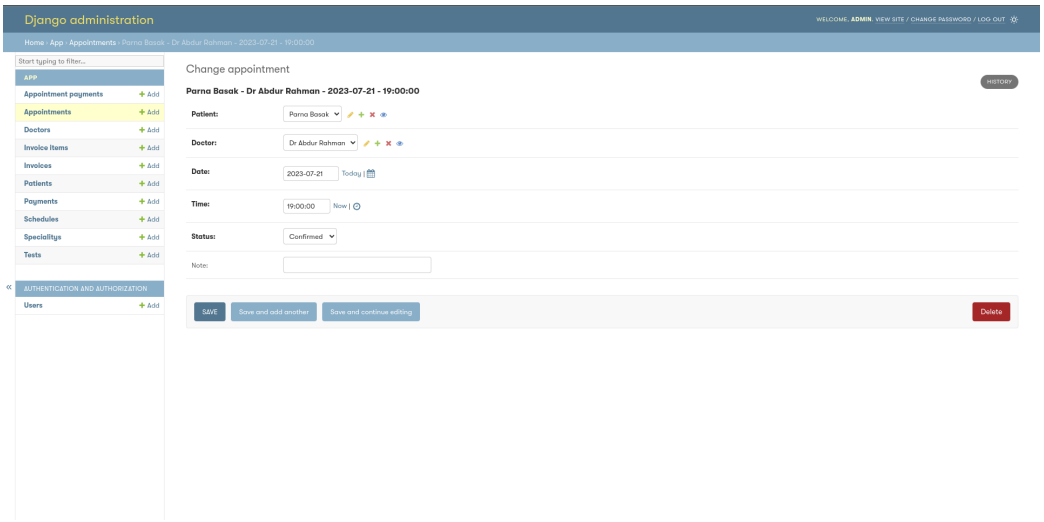

Figure 4.10: Admin Dashboard - Appointment Management

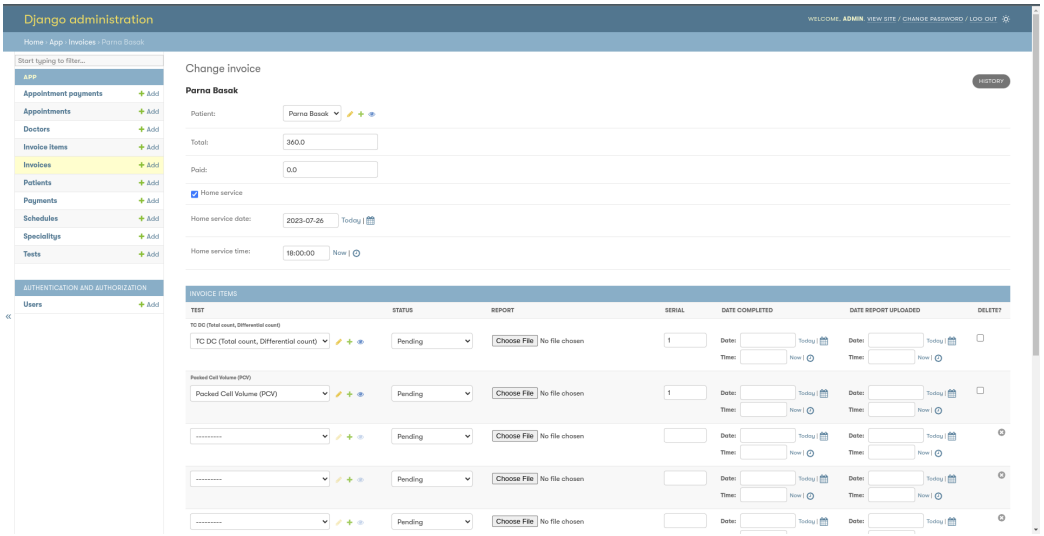

Figure 4.11: Admin Dashboard - Invoice Management

4.3.4 Patient Dashboard Features

4.3.5 Patient Dashboard Screenshots

Below are some screenshots of the Patient Dashboard:

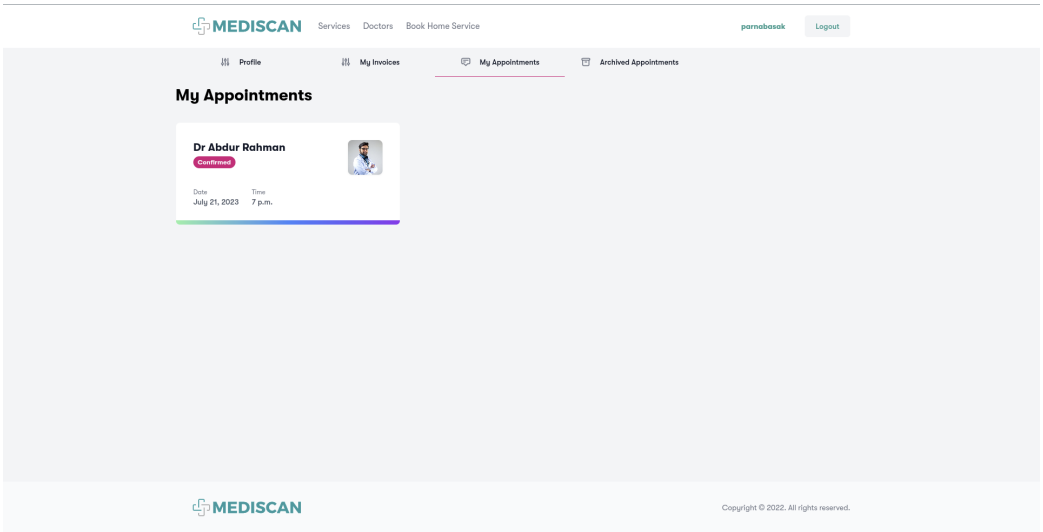

Figure 4.12: Patient Dashboard - Appointment Management

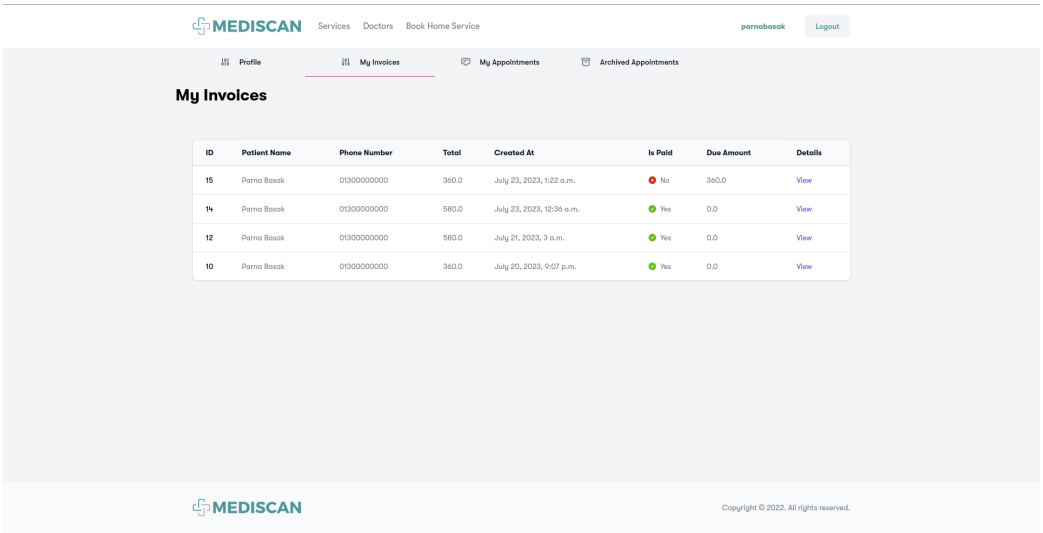

Figure 4.13: Patient Dashboard - Invoice List

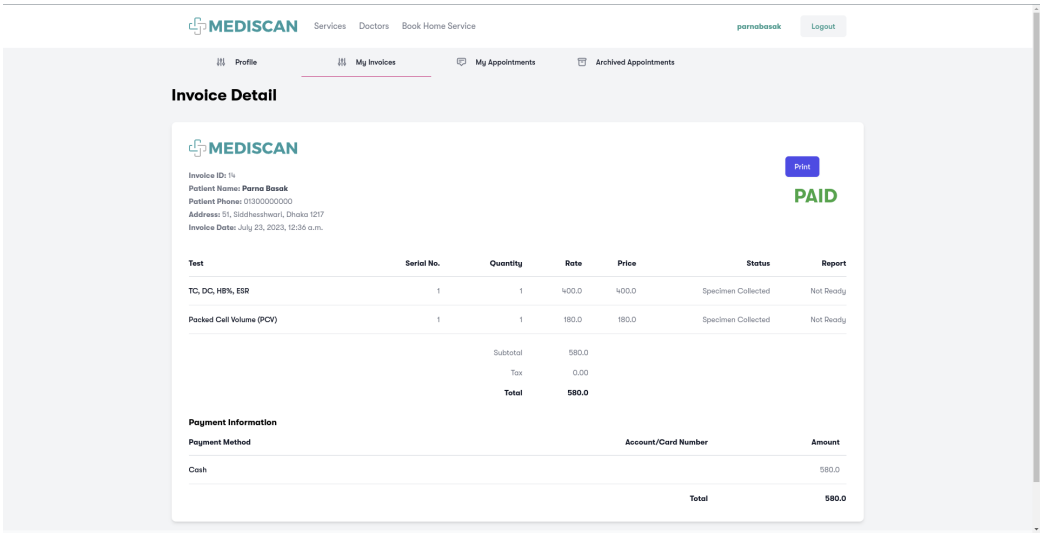

Figure 4.14: Patient Dashboard - View Invoice Details

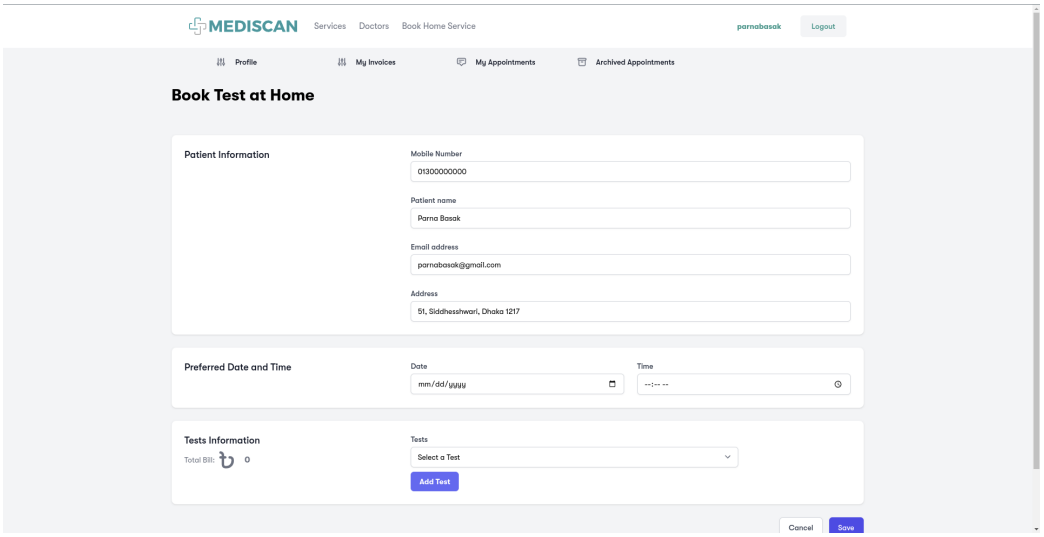

Figure 4.15: Patient Dashboard - Create Invoice

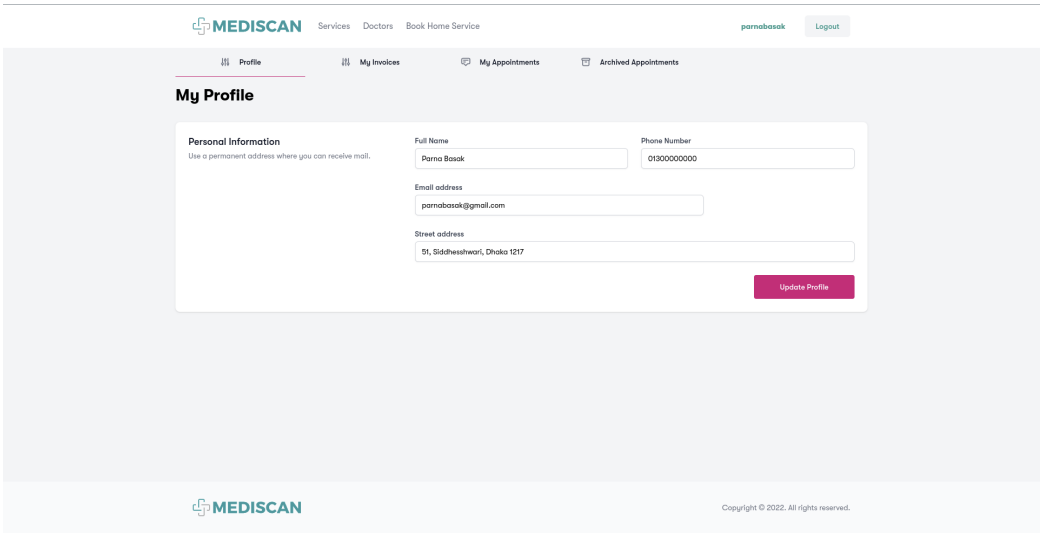

Figure 4.16: Patient Dashboard - Update Personal Information

4.3.6 Staff Dashboard Screenshots

Below are some screenshots of the Staff Dashboard:

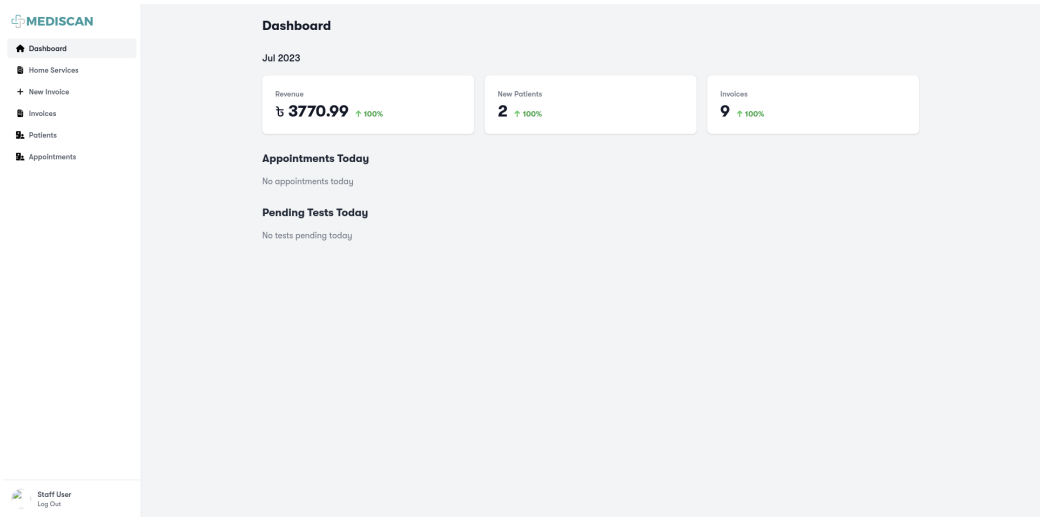

Figure 4.17: Staff Dashboard - Home

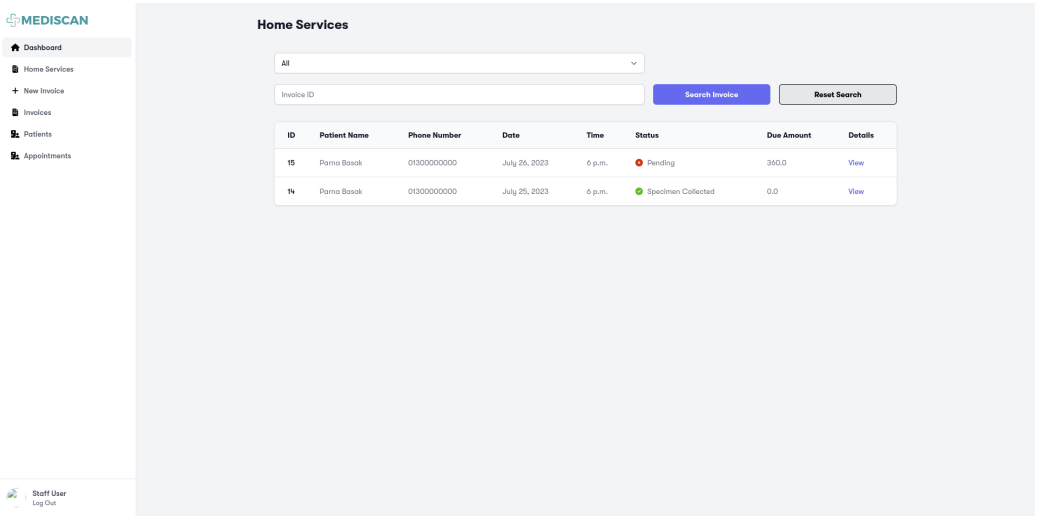

Figure 4.18: Staff Dashboard - Home Services Management

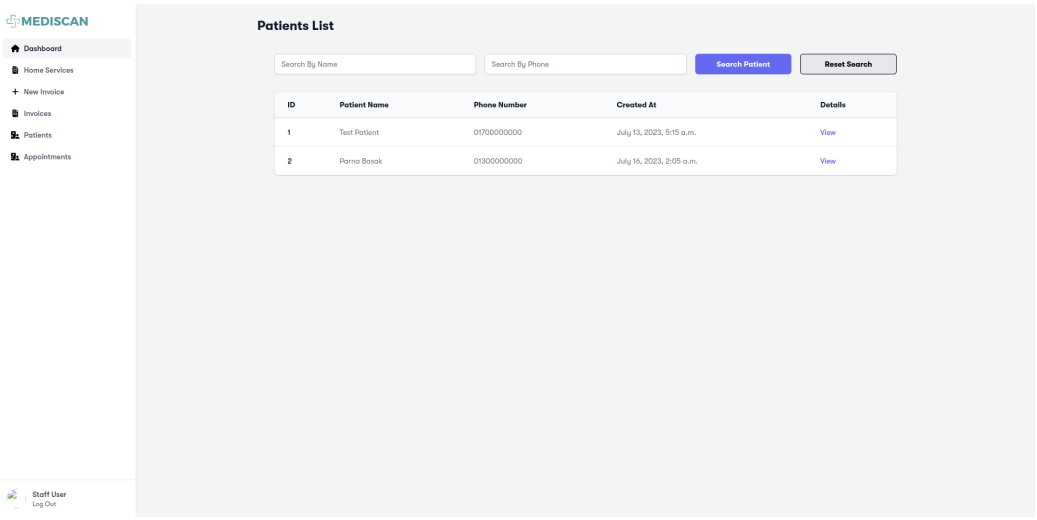

Figure 4.19: Staff Dashboard - Patient Management

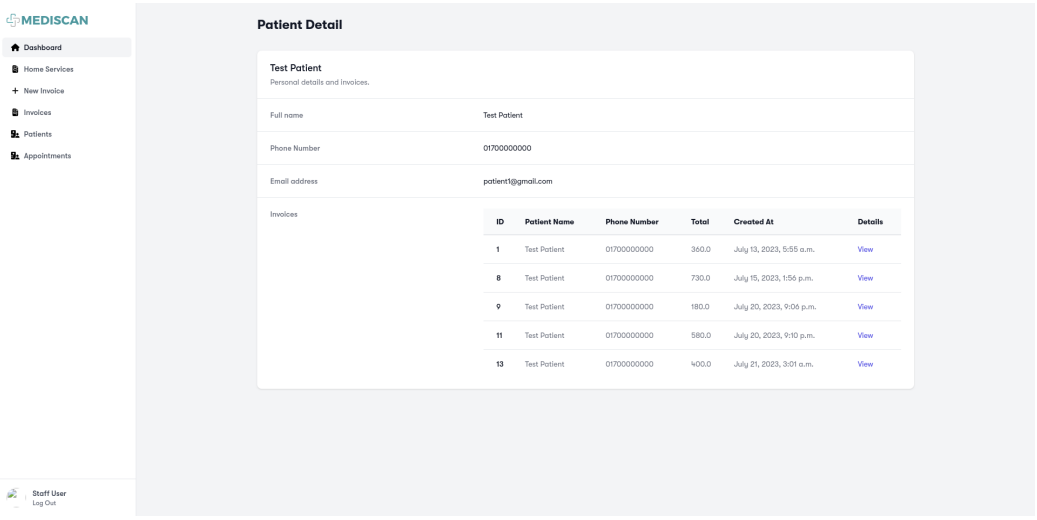

Figure 4.20: Staff Dashboard - View Patient Details

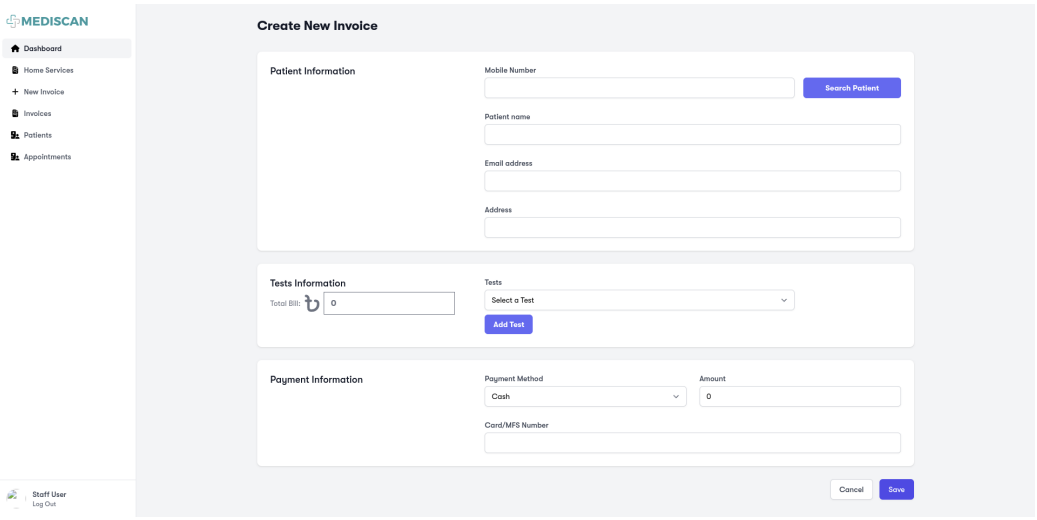

Figure 4.21: Staff Dashboard - Create Invoice

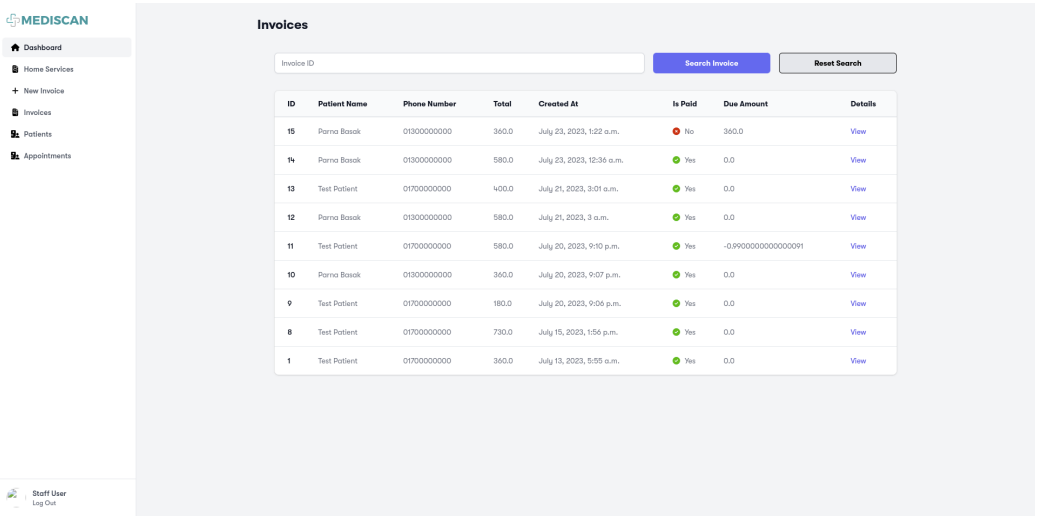

Figure 4.22: Staff Dashboard - View Invoices

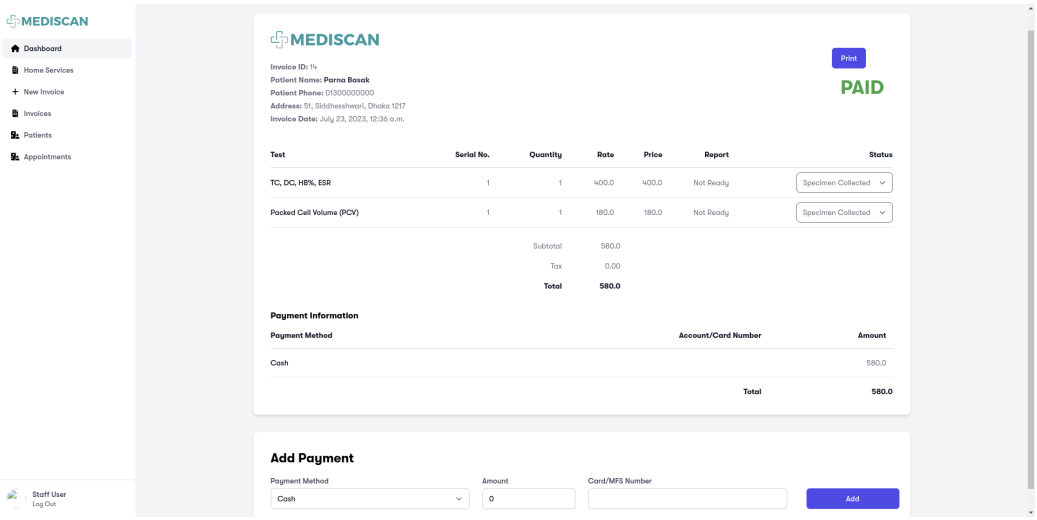

Figure 4.23: Staff Dashboard - View Invoices

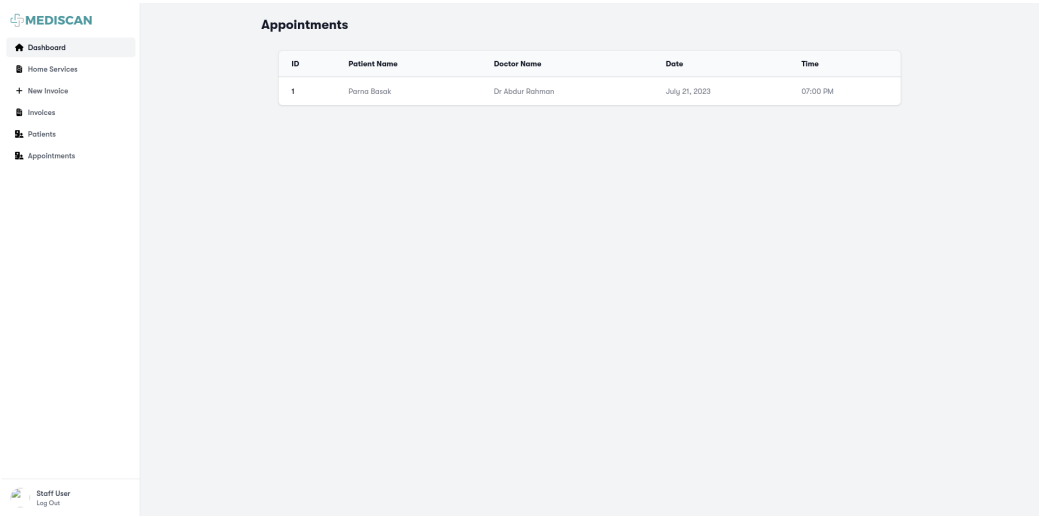

Figure 4.24: Staff Dashboard - Appointment Management

4.4 Conclusion

In conclusion, the implementation of the Diagnostic Center Management System using the Django framework and a well-chosen technology stack has been successful. The system offers a user-friendly experience for patients, staff, and administrators. The modular application structure ensures code maintainability and scalability. User authentication and role-based dashboards are efficiently implemented. Thorough testing and performance optimization have been prioritized. The system is now ready for deployment and marks a significant step towards streamlining operations and enhancing patient care in diagnostic centers.

5 Conclusion

5.1 Key Achievements

The development of the Diagnostic Center Management System has resulted in several key achievements that significantly enhance the efficiency and effectiveness of diagnostic centers. One of the main accomplishments is the successful implementation of core functionalities such as patient registration, appointment booking, and invoice management. These features were meticulously designed and rigorously tested to ensure seamless user interactions and accurate data management.

The modular architecture of the system played a crucial role in achieving a scalable and maintainable codebase. By following Django's best practices and leveraging its powerful features, developers were able to compartmentalize functionalities into separate apps, making it easier to maintain and extend the system in the future. This modularity also promotes code reusability, reducing redundancy, and streamlining development efforts.

The challenges related to database design, including defining appropriate relationships between models and ensuring data integrity, were effectively addressed. By utilizing Django's Object-Relational Mapping (ORM) system, the database integration process was simplified, allowing for efficient data storage and retrieval. The system's robust database design lays the foundation for reliable and accurate record-keeping, essential for the seamless functioning of diagnostic centers.

User authentication, a critical aspect of the system, was meticulously implemented with an emphasis on security and privacy. The system's user registration, login, and logout functionalities were designed to safeguard sensitive patient information and prevent unauthorized access. The seamless integration of Django's authentication system ensures that patients and staff can securely access their personalized dashboards, enhancing data privacy and control.

The front-end design, achieved through a combination of Django Templates and Tailwind CSS, resulted in an intuitive and visually appealing user interface. Patient, staff and administrator dashboards were thoughtfully designed to provide a user-friendly experience, allowing users to navigate the system effortlessly. The front-end's responsiveness ensures that the application is accessible across various devices, accommodating users' preferences.

5.2 Lessons Learned

The background study provided valuable information on the healthcare industry, highlighting the pivotal role of diagnostic centers in facilitating disease diagnosis and patient care. Understanding the significance of these centers reinforced the purpose of the project and highlighted the impact that a well-designed management system could have on healthcare services.

Exploring the role of technology in healthcare sheds light on the myriad advancements that have revolutionized the industry. The use of software solutions and management systems has greatly improved healthcare services, contributing to improved patient outcomes and operational efficiency. These findings underscored the importance of developing a robust and user-friendly diagnostic center management system to address the unique challenges faced by healthcare providers.

Throughout the development process, several valuable lessons were learned. Emphasizing user feedback and involving end-users in the User Acceptance Testing (UAT) phase was instrumental in identifying pain points and areas for improvement. This iterative approach enabled the team to fine-tune the system's functionalities, ensuring that it aligns seamlessly with the needs of diagnostic center staff and patients.

Continuous testing and debugging were crucial to identifying and correcting errors early in the development cycle. Rigorous testing procedures helped maintain code quality and reduce the likelihood of critical issues affecting the system's performance. The adherence to best practices for security and data privacy mitigated potential vulnerabilities, creating a robust and secure environment for data management.

5.3 Future Enhancements

While the current version of the Diagnostic Center Management System successfully fulfills its primary objectives, there are several areas for future enhancements and additional features.

The integration of advanced analytics and machine learning algorithms holds tremendous potential for the system. By analyzing historical patient data, the system could enable predictive analysis, aiding in disease diagnosis and treatment planning. Predictive modeling could assist healthcare providers in making informed decisions and recommending personalized treatment options, optimizing patient care.

Exploring telemedicine functionalities could expand the system's capabilities and accessibility. Introducing video consultations and remote patient monitoring features

could facilitate virtual healthcare services, particularly useful in scenarios where physical visits may not be possible or convenient for patients. Telemedicine could extend the reach of diagnostic centers, providing healthcare services to a broader patient population.

Developing a dedicated mobile application for the Diagnostic Center Management System could offer additional convenience and accessibility to patients and staff. A mobile app would enable users to access the system on-the-go, allowing patients to view appointments, access reports, and receive notifications. For staff, a mobile app would enhance mobility and streamline administrative tasks, leading to improved operational efficiency.

Incorporating features to support multilanguage support and localization could enhance the system's usability for a diverse user base. By catering to different languages and cultural preferences, the system could accommodate users from various regions, further improving their experience and satisfaction.

5.4 Conclusion

The successful development of the Diagnostic Center Management System marks a significant step towards transforming healthcare services and improving patient care. Through careful planning, diligent implementation, and iterative improvements based on user feedback, the system has become a valuable tool for diagnostic centers to streamline their operations and provide better healthcare services to patients.

The project's achievements, including the implementation of core functionalities, modular architecture, robust database integration, and secure user authentication, demonstrate the capabilities of modern technology in improving healthcare services. The lessons learned throughout the development process have reinforced the importance of user-centered design and continuous testing in creating efficient and user-friendly software solutions.

Looking ahead, the future enhancements

discussed offer exciting possibilities for further elevating the system's capabilities. With the integration of advanced analytics, telemedicine functionalities, and a mobile application, the Diagnostic Center Management System can continue to evolve, positively impacting the healthcare industry and the lives of patients and healthcare professionals alike. By continuously adapting and incorporating emerging technologies, the system can stay at the forefront of healthcare management, contributing to the overall well-being of the community it serves.

References

[1] Django Software Foundation, “The web framework for perfectionists with deadlines — django.” [Online]. Available: https://www.djangoproject.com/

[2] Django Soft. Foundation, “Templates - django documentation - django.” [Online]. Available: https://docs.djangoproject.com/en/4.2/topics/templates/

[3] mdn web docs, “Html: Hypertext markup language — mdn,” (Accessed on 09/15/2023). [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML

[4] Tailwind Labs, “Tailwind css - rapidly build modern websites without ever leaving your html.” [Online]. Available: https://tailwindcss.com/

[5] AgileBio, “All-in-one lab management & notebook — labcollector lims.” [Online]. Available: https://labcollector.com/

[6] Addatech, “Clinicmaster - cloud-based emr software solution, quebec-.” [Online]. Available: https://clinicmaster.com/

[7] OpenMRS, “Openmrs.org.” [Online]. Available: https://openmrs.org/

[8] N. Arora, “Django project mvt structure - geeksforgeeks.” [Online]. Available: https://www.geeksforgeeks.org/django-project-mvt-structure/

[9] Lucidchart, “Er diagram (erd) - definition & overview — lucidchart,” (Accessed on 09/15/2023). [Online]. Available: https://www.lucidchart.com/pages/er-diagrams

[10] Lucidchart, “Uml use case diagram tutorial — lucidchart,” (Accessed on 09/15/2023). [Online]. Available: https://www.lucidchart.com/pages/uml-use-case-diagram

[11] The Django Project, “The django admin site — django documentation — django,” (Accessed on 09/15/2023). [Online]. Available: https://docs.djangoproject.com/en/4.2/ref/contrib/admin/

[12] Lucidchart, “What is a data flow diagram — lucidchart,” (Accessed on 09/15/2023). [Online]. Available: https://www.lucidchart.com/pages/data-flow-diagram

[13] Microsoft, “Visual studio code - code editing. redefined.” [Online]. Available: https://code.visualstudio.com/

[14] SQLite, “Sqlite home page.” [Online]. Available: https://www.sqlite.org/index.html

[15] Django Software Foundation, “Writing your first django app, part 2 — django documentation — django.” [Online]. Available: