ColonCare
[Colon cancer Centre]

Mini Project Report

Submitted by

Devika Murali

Reg. No.: AJC19MCA-I021

In Partial fulfillment for the Award of the Degree of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**COLONCARE**" is the bona fide work of **DEVIKA MURALI (Regno: AJC19MCA-I021)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

T J Jobin   **Meera Rose Mathew**

Internal Guide   **Coordinator**

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

DECLARATION

I hereby declare that the project report **"COLONCARE"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: **DEVIKA MURALI**

KANJIRAPPALLY   **Reg: AJC19MCA-I021**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Meera Rose Mathew** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Mr. T J Jobin** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

DEVIKA MURALI

ABSTRACT

.

ColonCare, your partner in comprehensive colon health and cancer care is like a super useful website that makes scheduling appointments at a colon cancer center really easy. This web-based system has been meticulously crafted to cater to the distinct needs of administrators, doctors, and patients, providing a seamless and efficient experience for all stakeholders. Administrators, armed with their unique login credentials, gain access to a suite of essential tools to oversee and enhance the center's operations. They can manage doctor specializations with ease, ensuring that patients are matched with healthcare professionals possessing the appropriate expertise. Additionally, administrators can approve doctor registrations efficiently, further streamlining the onboarding process for medical practitioners. Equally crucial, they have access to appointment data and patient lists, granting them the ability to monitor and optimize the center's appointment booking flow.

For patients, the website offers an intuitive and user-friendly interface that simplifies the healthcare journey. Patients can register effortlessly, create appointments with their preferred doctors, and access detailed profiles of healthcare providers to make informed decisions about their care. Furthermore, they can view their medical history on the platform, streamlining the sharing of vital health information with their chosen physicians. The system even allows patients to upload their past medical records, enhancing the completeness of their healthcare profiles. In addition to these functionalities, patients can manage their profiles and passwords conveniently, ensuring a secure and personalized experience.

Doctors, too, benefit from this comprehensive website. With their specialized login access, they can effortlessly review patient lists, manage appointments, and update their professional profiles to reflect their expertise and availability accurately. The platform simplifies appointment management for doctors, giving them the tools, they need to approve or reject appointment requests swiftly.

In essence, the "**ColonCare**" represents a user-centric approach to appointment booking and management, enhancing the experiences of both patients and healthcare providers within the colon cancer center while optimizing the overall efficiency of the healthcare delivery process.

CONTENT

List of Abbreviation

- IDE - Integrated Development Environment
- HTML - Hyper Text Markup Language
- CSS - Cascading Style Sheet
- SQL - Structured Query Language
- UML - Unified Modeling Language
- JS - JavaScript
- AJAX - Asynchronous JavaScript and XML Environment
- AI - Artificial Intelligence
- URL - Uniform Resource Locator
- PY - Python

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

ColonCare is a web-based platform that serves as a valuable resource for individuals seeking comprehensive colon health and cancer care. This user-centric system has been thoughtfully developed to cater to the specific needs of three key user groups: administrators, doctors, and patients, offering tailored functionalities to each. Administrators benefit from unique login credentials and access to essential tools that empower them to efficiently manage doctor specializations, streamline the onboarding of medical practitioners, and oversee the center's appointment booking process through real-time access to appointment data and patient lists. Patients, on the other hand, experience a user-friendly interface that simplifies the healthcare journey, allowing for easy registration, appointment scheduling with preferred doctors, informed decision-making via detailed doctor profiles, and the convenience of accessing and uploading their medical history. Patients can also manage their profiles and passwords securely. Likewise, doctors have specialized login access, enabling them to review patient lists, manage appointments, and update their professional profiles accurately, making the appointment management process swift and efficient. In essence, **ColonCare** embodies a user-centric approach to appointment booking and management, enhancing the experiences of patients and healthcare providers within the colon cancer center while optimizing the overall efficiency of healthcare delivery.

1.2 PROJECT SPECIFICATION

Administrators:
1. **Login and Dashboard:** Administrators have unique login credentials to access the system, where they are greeted with a comprehensive dashboard.

2. **Manage Doctor Specializations:** Administrators can efficiently organize and update doctor specializations, ensuring patients are matched with the right healthcare professionals.

3. **Doctor Registration Approval:** Administrators can swiftly approve or reject doctor registrations, streamlining the onboarding process for new medical practitioners.

4. **Appointment and Patient Data:** They have access to crucial appointment data and patient lists, allowing them to monitor and optimize the appointment booking flow.

Patients:
1. **User-Friendly Registration:** Patients can easily register on the website, creating their accounts for personalized access.

2. **Appointment Scheduling:** Patients have the convenience of booking appointments with their preferred doctors through an intuitive interface.

3. **Doctor Profiles:** They can access detailed profiles of healthcare providers, helping them make informed decisions about their care.

4. **Medical History Access:** Patients can view their medical history on the platform, facilitating the sharing of vital health information with their chosen physicians.

5. **Medical Record Upload:** The system allows patients to upload their past medical records, enhancing the completeness of their healthcare profiles.

6. **Profile Management:** Patients can manage their profiles and passwords for a secure and personalized experience.

Doctors:
1. **Doctor Dashboard:** Doctors have specialized login access, leading them to a dashboard tailored to their needs.

2. **Patient List Review:** They can effortlessly review their patient lists, ensuring they are well-prepared for appointments.

3. **Appointment Management:** Doctors can manage their appointments efficiently, including approving or rejecting appointment requests as needed.

4. **Profile Updates:** Doctors can update their professional profiles to accurately reflect their expertise and availability.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

In the realm of information technology and systems analysis, system study is an essential step that aims to thoroughly comprehend, analyse, and evaluate an existing or prospective system. It is a crucial stage in creating, enhancing, or putting into practise systems that are tailored to the demands of a given organisation.

In a system study, system analyst usually looks closely at the current system, its goals, and the different parts and procedures that make it up. Finding flaws or inefficiencies in the current system, specifying specifications for a new system, and putting forth ideas to solve organisational problems are the main objectives of system studies.

During this phase, information is gathered and analysed, stakeholders are interviewed, business processes are examined, and the structure and functionality of the system are documented. It could also include determining if the suggested changes are feasible, projecting expenses, and analysing any dangers.

The outcomes of a system study serve as a foundation for decision-making in organizations, helping them make informed choices about whether to proceed with system development or improvement. By understanding the intricacies of the existing system and the needs of the stakeholders, system study plays a pivotal role in aligning technology solutions with business objectives, ultimately leading to more effective and efficient systems.

2.2 EXISTING SYSTEM

The existing system at ColonCare, prior to the introduction of the web-based platform, was characterized by a more conventional and fragmented approach to appointment booking and management within the colon cancer center. Administrators had to rely on manual processes, such as phone calls and paperwork, to manage doctor specializations, onboard medical practitioners, and oversee appointment scheduling. This manual approach often resulted in delays and administrative overhead. Patients faced challenges in terms of appointment scheduling and accessing their medical history, as these processes were not streamlined, making the healthcare journey less user-friendly. Doctors, too, had to manage appointments and professional profiles through more traditional means,

which could be time-consuming and prone to errors. In essence, the existing system lacked the user-centric approach that the new web-based platform provides, resulting in less efficient healthcare delivery and potentially impacting the overall patient experience. The transition to the ColonCare web-based platform represents a significant advancement in addressing these shortcomings and enhancing the efficiency of colon health and cancer care services.

2.2.1 NATURAL SYSTEM STUDIED

The healthcare facility managed and scheduled appointments using antiquated manual procedures. The way that information and patient care moved in this arrangement was more traditional and dispersed, which reflected the way things naturally occur. Calls and papers were used to handle administrative activities such as practitioner onboarding, doctor specialisation, and appointment scheduling; this was similar to the organic, decentralised structure of a natural system. This strategy did not, however, come without consequences, as it frequently resulted in administrative inefficiencies, delays, and difficulties for both patients and medical staff. Beyond what a natural system could provide, the move to a web-based platform signified a substantial shift towards a more structured and user-centric approach, improving the effectiveness and calibre of colon health and cancer care services.

2.2.2 DESIGNED SYSTEM STUDIED

The newly designed system at ColonCare represents a comprehensive and user-centric approach to healthcare management, catering to the needs of administrators, patients, and doctors alike. For administrators, the system offers a secure login and an intuitive dashboard for efficient management of doctor specializations and seamless approval of new practitioner registrations. This streamlines the process of matching patients with the right healthcare professionals, ultimately enhancing the patient experience. Patients, on the other hand, benefit from a user-friendly registration process, intuitive appointment scheduling, and access to detailed doctor profiles, empowering them to make informed decisions about their care. The system goes a step further by providing access to medical history and allowing patients to upload past records, creating a holistic healthcare profile. Additionally, patients can manage their profiles for a personalized and secure experience. For doctors, the system offers a specialized dashboard, facilitating easy patient list review and efficient appointment management. They can also update their professional profiles, ensuring accuracy in their expertise and availability information. Overall, this thoughtfully designed system significantly improves the efficiency and quality of colon health and cancer care services, offering a seamless and user-friendly experience for all stakeholders involved.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Manual Processes**: The reliance on manual processes, including phone calls and paperwork, for various administrative tasks was a significant drawback. This approach was time-consuming and prone to human errors, leading to inefficiencies in managing doctor specializations, practitioner onboarding, and appointment scheduling.

- **Administrative Delays**: Due to the manual nature of the existing system, administrative delays were common. Approving doctor registrations, making updates to medical practitioner profiles, and handling appointment bookings all took longer than necessary, potentially causing delays in patient care.

- **Limited Patient Access**: Patients faced challenges in scheduling appointments and accessing their medical history. These processes were not streamlined, making it difficult for patients to manage their healthcare journey and view their vital health information conveniently.

- **Inefficient Doctor Management**: Doctors, too, had to contend with the limitations of the existing system. Managing appointments and their professional profiles through traditional means was not only time-consuming but also created a risk of errors. This lack of efficiency could impact the doctors' ability to provide timely and accurate care.

- **Lack of User-Centric Approach**: The overarching drawback of the existing system was the absence of a user-centric approach. The system did not prioritize the needs and convenience of patients, administrators, and doctors. This lack of user-friendliness had the potential to impact the overall patient experience and hinder efficient healthcare delivery.

2.4 PROPOSED SYSTEM

The proposed system at ColonCare revolutionizes healthcare management with a user-centric approach. It empowers administrators with streamlined doctor specialization management and quick practitioner approvals. Patients benefit from easy registration, intuitive appointment scheduling, and access to comprehensive doctor profiles and medical history. Patients can also upload their medical records, ensuring complete healthcare profiles. Doctors enjoy efficient patient list management and appointment handling, along with the ability to update their professional profiles. Overall, this system significantly enhances the efficiency and quality of colon health and cancer care services, offering a seamless and user-friendly experience for all

stakeholders involved.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- User-Centric Approach: The system prioritizes the needs of administrators, patients, and doctors, enhancing the overall experience and satisfaction of all stakeholders.

- Efficient Administration: Administrators benefit from streamlined doctor specialization management and quick practitioner approvals, reducing administrative overhead and ensuring optimal resource allocation.

- Patient Empowerment: Patients enjoy a user-friendly registration process, intuitive appointment scheduling, and access to comprehensive doctor profiles and their own medical history. This empowers them to make informed decisions about their care.

- Medical Record Accessibility: Patients can easily upload their medical records, leading to complete healthcare profiles and enabling seamless sharing of vital health information with healthcare providers.

- Doctor Efficiency: Doctors can efficiently manage their patient lists and appointments, leading to better preparation and scheduling. They can also keep their professional profiles up to date, ensuring the accuracy of their expertise and availability information.

- Improved Healthcare Quality: Overall, the system significantly enhances the efficiency and quality of colon health and cancer care services, reducing delays, administrative errors, and inefficiencies, ultimately leading to better patient care and experiences.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

This examination is a fundamental step in determining whether a project will achieve the organization's objectives considering the resources, time invested in it. It assists the developer in assessing the potential benefits and long-term possibilities of the project. To ascertain the feasibility and worthiness of further analysis, a feasibility study must be conducted for the proposed system. The feasibility study evaluates how the proposed system would impact the organization, its ability to meet customer demands, and the efficient use of resources. Consequently, a feasibility study is typically conducted before proceeding with the development of a new application. The assessment carefully considers various factors, including technical, financial, and operational viability, as outlined in the feasibility study document.

3.1.1 Economical Feasibility
The "ColonCare" project makes economic sense because while there are costs to create and run the website, we can make money from it. People will pay to use it or see ads on it, so it's a good investment that can help us financially in the long run. So, it's a smart choice from an economic standpoint.

3.1.2 Technical Feasibility

The "ColonCare" project is technically doable because we have the right technology and skilled people. It can work well with other healthcare systems, and we can keep patients' information safe. The website will be easy to use, even on phones, and we'll make sure it keeps working even if there are problems. We'll also update it regularly and help if people have technical issues.

3.1.3 Behavioral Feasibility

The "ColonCare" project is operationally feasible because it fits well with how the colon cancer center already works, and it won't cause too much disruption. It's like adding a helpful tool that everyone can use without major changes in how things are done. Plus, we can easily follow the rules and laws related to healthcare. So, it's practical and makes sense for our operations.

3.1.4 Feasibility Study Questionnaire

1. Project Overview?

ColonCare is a web-based platform that serves as a valuable resource for individuals seeking comprehensive colon health and cancer care. This user-centric system has been thoughtfully developed to cater to the specific needs of three key user groups: administrators, doctors, and patients, offering tailored functionalities to each. Administrators benefit from unique login credentials and access to essential tools that empower them to efficiently manage doctor specializations, streamline the onboarding of medical practitioners, and oversee the center's appointment booking process through real-time access to appointment data and patient lists. Patients, on the other hand, experience a user-friendly interface that simplifies the healthcare journey, allowing for easy registration, appointment scheduling with preferred doctors, informed decision-making via detailed doctor profiles, and the convenience of accessing and uploading their medical history. Patients can also manage their profiles and passwords securely. Likewise, doctors have specialized login access, enabling them to review patient lists, manage appointments, and update their professional profiles accurately, making the appointment management process swift and efficient. In essence, **ColonCare** embodies a user-centric approach to appointment booking and management, enhancing the experiences of patients and healthcare providers within the colon cancer center while optimizing the overall efficiency of healthcare delivery.

2. To what extend the system is proposed for?

ColonCare is a user-centric web-based platform designed to optimize colon health and cancer care within a dedicated center. It caters to administrators, doctors, and patients, offering specialized tools and functionalities to each group. Administrators can efficiently manage the center's operations, doctors can streamline appointment management, and patients can easily access care and medical information, all aimed at enhancing the overall efficiency of healthcare delivery within the center.

3. Specify the Viewers/Public which is to be involved in the System?

Users

4. List the Modules included in your System?

Admin, Doctors, Patients

5. Identify the users in your project?

Admin, Doctors, Patients

6. Who owns the system?

Administrator

7. System is related to which firm/industry/organization?

Healthcare Industry

8. Details of person that you have contacted for data collection?

Dr. Boben Thomas
Consultant Medical Oncologist

9. Questionnaire to collect details about the project? (min 10 questions, include descriptive answers, attach additional docs (e.g. Bill receipts, certificate models), if any?)

1. What specific medical services, treatments, and therapies are provided?
ANS: Our center offers a range of medical services, including colonoscopies, surgery, chemotherapy, and radiation therapy, tailored to the needs of patients.

2. How does the cancer center make sure that patients feel well taken care of and comfortable during their entire cancer treatment process?
ANS: Our center ensures patient comfort through personalized support, a compassionate approach, and a comfortable environment, including counseling and pain management services.

3. Are there support services such as counseling, nutrition guidance, and pain management available for patients and their families?
ANS: Yes, the cancer center provides support services including counseling, nutrition guidance, and pain management to assist both patients and their families throughout their cancer journey.

4. Are there any specific steps or guidelines patients need to follow when booking appointments online?
ANS: Patients need to follow a straightforward online booking process, entering their information and selecting their preferred date and time for appointments.

5. How are patient testing results kept secure and private during the delivery process?
ANS: Patient testing results are kept secure and private through encrypted

electronic transmission and access controls, ensuring that only authorized individuals can view and handle the data.

6. Are there specific eligibility criteria or age recommendations for certain screening services?
ANS: Yes, certain screening services may have specific eligibility criteria or recommended age ranges based on medical guidelines and individual risk factors.

7. Are there options for patients to book urgent or expedited screening appointments?
ANS: Yes, the center offers options for patients to book urgent or expedited screening appointments to accommodate their specific medical needs and timelines.

8. Are patients provided with information on how to take their medications correctly, including dosage instructions, potential side effects, and interactions?
ANS: Yes, patients receive comprehensive information on medication usage, including dosage instructions, potential side effects, and possible interactions, to ensure safe and effective use.

9. What types of endoscopy procedures are offered at the facility?
ANS: The facility offers a range of endoscopy procedures, including upper endoscopy, colonoscopy, bronchoscopy, and laparoscopy, among others, to diagnose and treat various medical conditions.

10. What is the process for communicating endoscopy results to patients?
ANS: Endoscopy results are typically communicated to patients through a follow-up appointment with the healthcare provider who performed the procedure, where they discuss the findings and provide a summary report.

3.1 SYSTEM SPECIFICATION
3.2.1 Hardware Specification

Processor   -   12th Gen Intel(R)

RAM   -   8.00 GB

Hard disk   -

3.2.2 Software Specification

Front End   -   HTML5, Bootstrap, CSS

Back End   -   Django Framework, Python

Database   -   SQLite

Client on PC   -   Windows 7 and above.

Technologies used -   JS, HTML5, AJAX, J Query, Python, CSS, Django, SQLite

3.3 SOFTWARE DESCRIPTION

3.3.1Django Framework:

Django is a versatile and powerful open-source web framework for Python, known for its "batteries-included" approach, which provides a comprehensive set of tools and features to expedite web application development. Following the Model-View-Template (MVT) architectural pattern, Django enforces a clean separation of concerns and promotes the "Don't Repeat Yourself" (DRY) principle. It offers an array of built-in security measures, user authentication, and an efficient admin interface for site management. With a supportive community and extensive documentation, Django is a go-to choice for developers seeking to build scalable and secure web applications, from simple websites to complex, high-traffic platforms. Its adaptability and robust ecosystem of reusable packages make it an attractive option for a wide range of projects.

3.3.2 SQLite:

SQLite is a lightweight and self-contained relational database management system, making it a popular choice for applications where simplicity, speed, and portability are paramount. Its serverless nature allows the entire database to reside in a single file, simplifying deployment and management. It's an ACID-compliant system, ensuring data integrity, and supports a subset of SQL, including essential features like transactions, indexes, views, and triggers. SQLite's minimal memory and storage footprint make it ideal for resource-constrained environments and embedded systems, while its cross-platform compatibility extends its versatility across various operating systems and mobile platforms. The open-source nature of SQLite and its public domain license make it free for any use, and its active community and extensive documentation provide valuable resources for developers seeking a compact and efficient database solution.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

Any designed system or product's development starts with the design phase. An efficient system depends on well-executed design, which is a creative process. It entails utilizing a variety of approaches and concepts to define a process or system in enough depth to allow for its actual execution. Regardless of the development model chosen, the design phase is critical in software engineering. It strives to produce the architectural detail needed to build a system or product and serves as the technical backbone of the software engineering process. This program has through a thorough design phase that optimizes every aspect of effectiveness, performance, and accuracy. A user-oriented document is converted into a document for programmers or database employees during the design process.

4.2 UML DIAGRAM

Unified Modeling Language (UML) is a standardized modeling language used in software engineering for visualizing, designing, and documenting software systems. UML diagrams are graphical representations of different aspects of a software system or a process, and they play a crucial role in understanding, communicating, and documenting software systems.
Here are some key UML diagrams:

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation in the Unified Modeling Language (UML) that illustrates the functional requirements and interactions of a system from the perspective of its users, known as "actors." Actors are depicted outside the system boundary, while use cases, representing specific functionalities or system responses, are enclosed within it. The associations between actors and use cases are represented by arrows, indicating the interactions and relationships. Use case diagrams are indispensable for requirements analysis, system design, communication among stakeholders, test case definition, and documentation, making them a vital tool in software development for modeling and understanding the system's behavior and its user interactions.
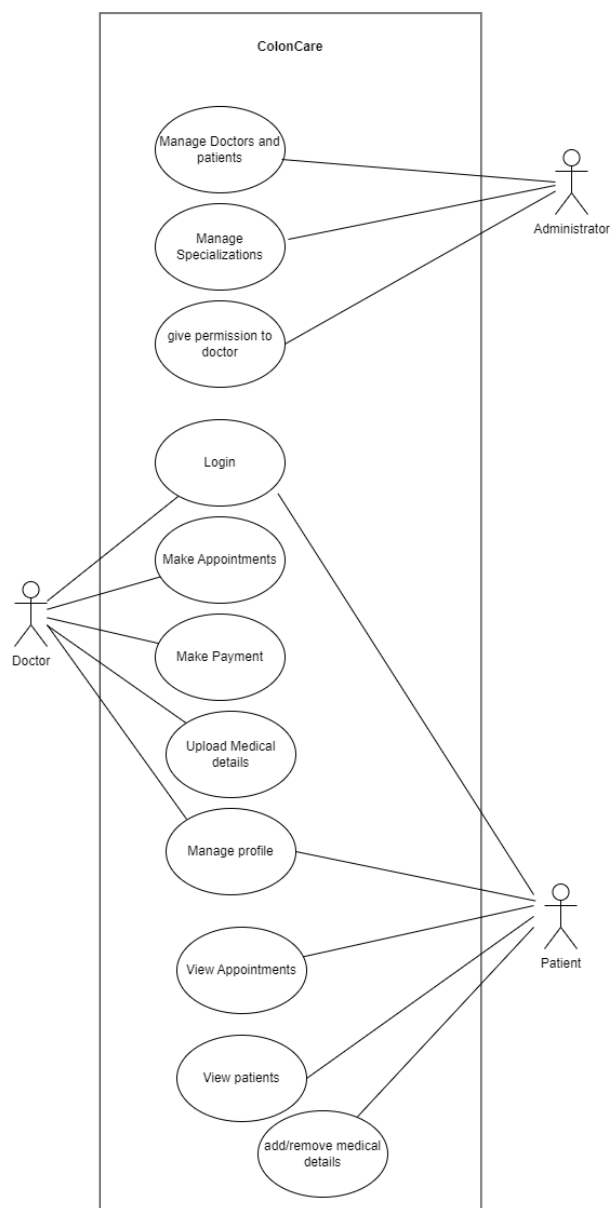
4.2.1   SEQUENCE DIAGRAM

A sequence diagram, a widely used type of Unified Modeling Language (UML) diagram in software engineering, offers a visual representation of the dynamic interactions and communication between different components, objects, or actors within a system over time. Lifelines represent these elements, and messages between lifelines illustrate the order and flow of interactions, whether synchronous or asynchronous. Activation bars show when a lifeline is actively processing a message, and return messages convey responses or acknowledgments. Sequence diagrams are pivotal for system design, aiding in architecture decisions, communication among stakeholders, behavior modeling, testing, and documentation. They provide a dynamic blueprint of how a system functions, making them a crucial tool for understanding and visualizing the runtime behavior of software systems.
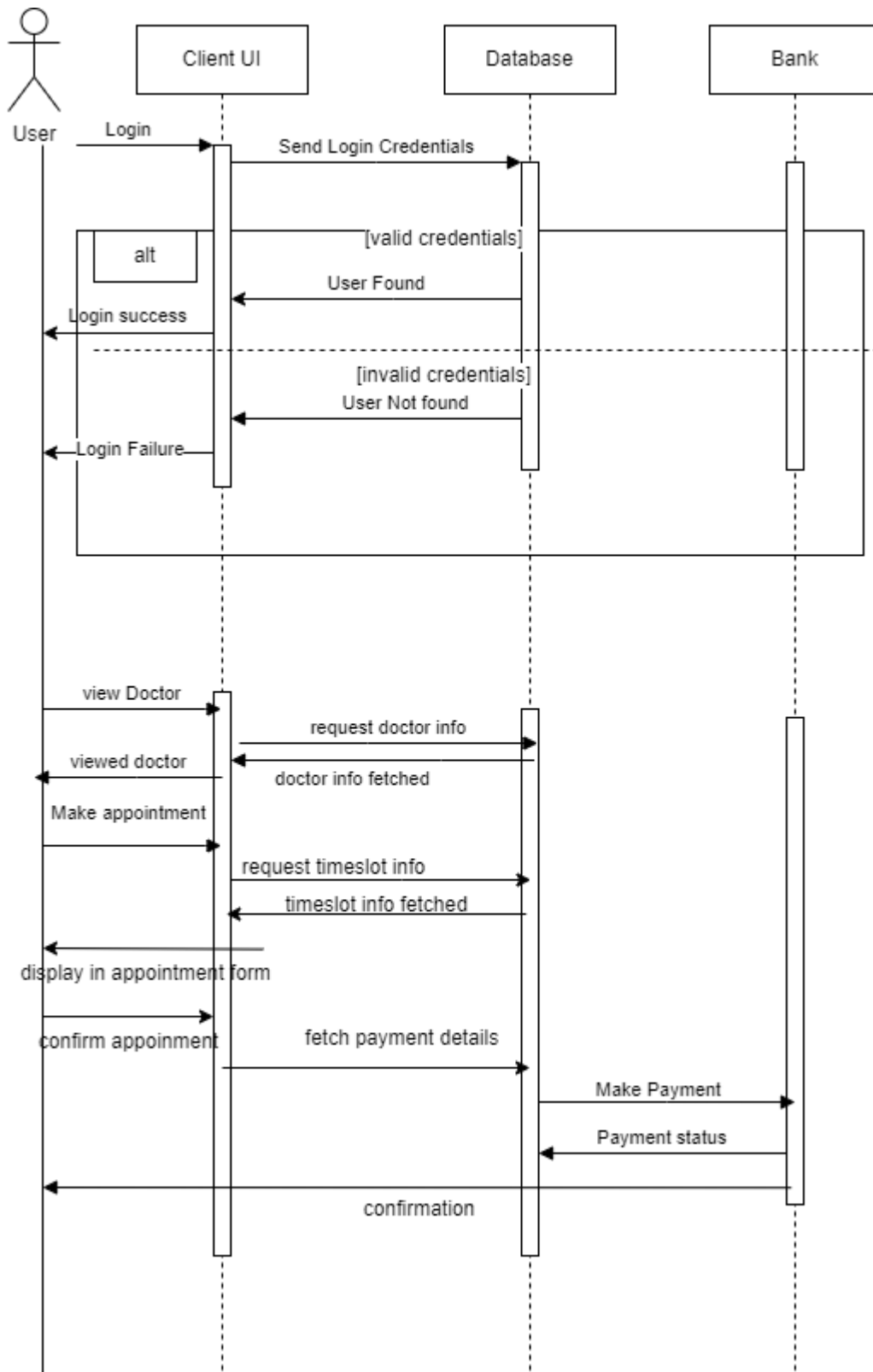
4.2.2 State Chart Diagram

A state chart diagram, also known as a state machine diagram, is a type of Unified Modeling Language (UML) diagram used in software engineering to model the dynamic behavior of an object, component, or system. State chart diagrams represent an object's life cycle and the transitions between different states, making them particularly useful for modeling complex systems or objects that exhibit distinct behaviors in response to events or changes in conditions. Here are key elements and concepts associated with state chart diagrams:

1. **States**: States represent the distinct conditions or modes in which an object or system can exist. Each state typically has a name and is visually represented as a rounded rectangle. Objects can transition between these states in response to various events.

2. **Transitions**: Transitions are depicted as arrows and represent the movement of an object from one state to another in response to events or conditions. Transitions can be triggered by events, actions, or guard conditions, and they indicate how an object's state changes.

3. **Events**: Events are external stimuli or occurrences that trigger transitions between states. They can be explicit, such as a button click, or implicit, such as a timer reaching a specific point. Events are represented as labels on transition arrows.

4. **Actions**: Actions represent the activities or behaviors that occur during a state transition. They can be used to describe what happens when an object enters or exits a state or during a transition between states.

5. **Initial and Final States**: An initial state denotes the starting point of an object's life cycle, while a final state represents the end or termination of the object's life cycle. These states help clarify the beginning and ending points of a state chart.
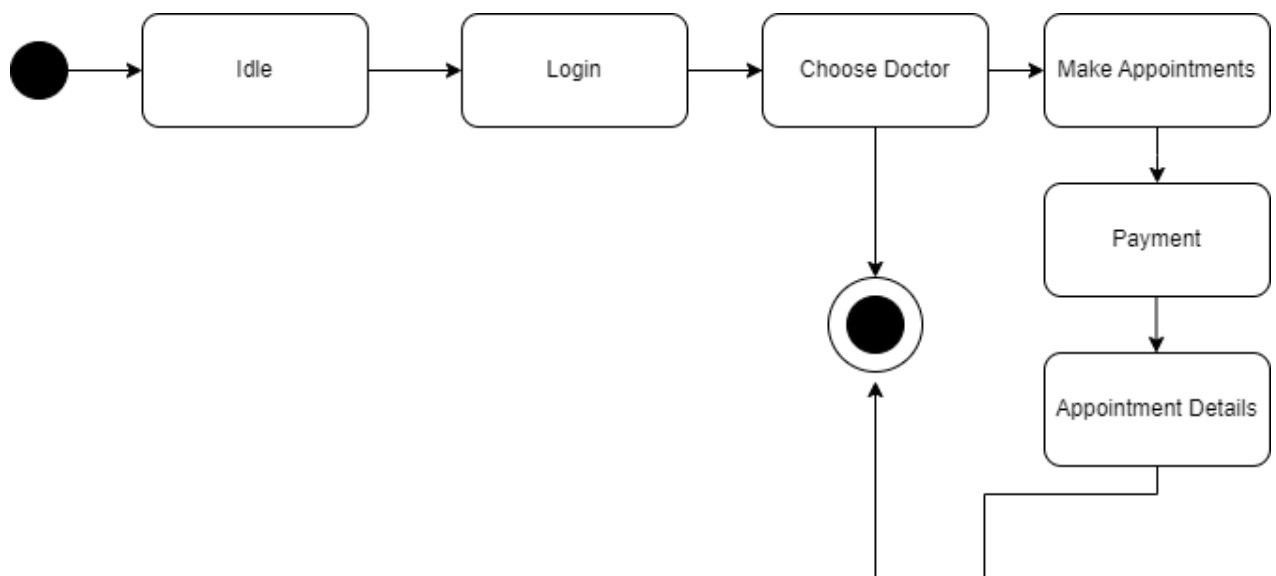
4.2.2 Activity Diagram

In the Unified Modeling Language (UML), an activity diagram is a graphical representation used to model the dynamic aspects of a system, including the flow of activities and actions within a process. Activity diagrams are primarily employed for process modeling, workflow modeling, and business process modeling. Here are key elements and concepts associated with activity diagrams:

Activities: Activities are the fundamental building blocks in an activity diagram. They represent a specific task, action, or operation that is performed within a process. Activities can range from simple actions like calculations to more complex processes.

Actions: Actions are specific steps within an activity that describe the work being performed. They can be as simple as a single operation or more complex, representing a sequence of activities.

Control Flow: Control flow lines (arrows) connect activities and actions to define the order in which they are executed. These lines show the sequence of activities and the direction in which the process flows.

Decision Nodes: Decision nodes or decision points allow branching in the flow of the process. They are often associated with conditions that determine which path the process will follow based on certain criteria.

Merge Nodes: Merge nodes are used to join the flow of control from multiple branches back into a single path. They indicate that multiple paths converge.

Fork Nodes: Fork nodes represent the splitting of control flow into multiple concurrent paths, indicating that multiple activities can be executed in parallel.

Join Nodes: Join nodes, like merge nodes, indicate the convergence of multiple control flows. They are used to show that separate paths are coming together.

Start and End Nodes: Start nodes represent the beginning of a process, while end nodes represent the conclusion or termination point.
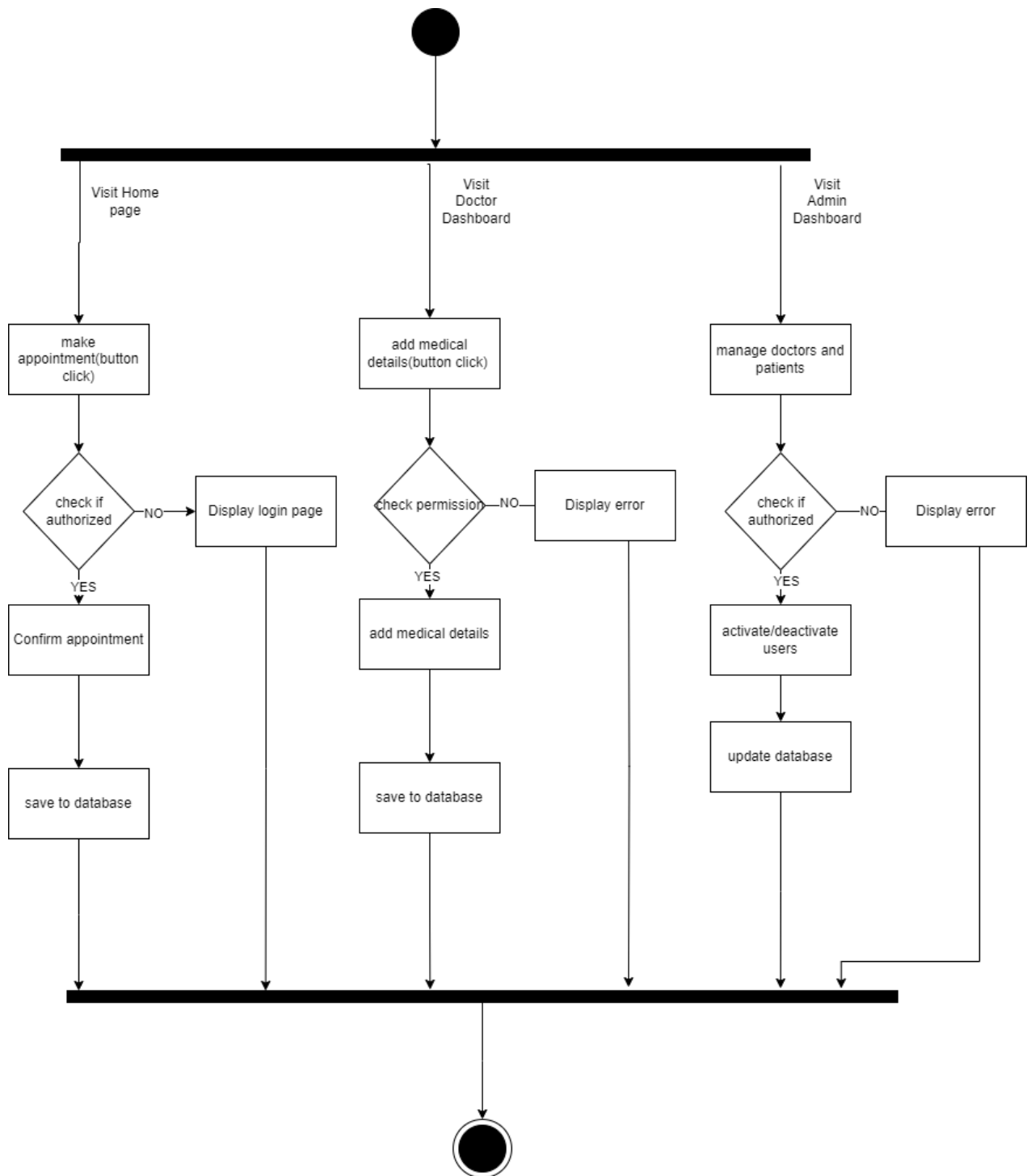
4.2.3   Class Diagram

In software engineering and system design, a class diagram is a fundamental component of the Unified Modeling Language (UML) used to depict the structure of a system or software application. Class diagrams provide a visual representation of the static aspects of a system, illustrating the classes, their attributes, methods, and the relationships between them. Here are key elements and concepts associated with class diagrams:

Class: A class is a blueprint or template for creating objects in object-oriented programming. It represents a concept, entity, or component in the system. Each class is typically depicted as a rectangle with three compartments: the top compartment contains the class name, the middle compartment lists the class attributes (data members or fields), and the bottom compartment displays the class methods (functions or operations).

Attributes: Attributes represent the properties or characteristics of a class. These can be data members, fields, or properties that store information relevant to the class.

Methods: Methods represent the behaviors or functions associated with a class. They define the operations that can be performed on the class or its attributes.

Association: Association lines connect two classes to illustrate a relationship between them. Associations show that instances of one class are linked to instances of another class. Multiplicity notations (e.g., "1", "0..", "1..") indicate how many objects can participate in the association.

Aggregation and Composition: Aggregation and composition represent types of associations where one class is part of another class. Aggregation typically represents a "whole-part" relationship, while composition signifies a stronger relationship, implying that the whole controls the existence of the part.

Inheritance (Generalization): Inheritance lines with a solid arrowhead indicate that one class is derived from another. This signifies a "is-a" relationship, indicating that the derived class inherits attributes and methods from the parent class.

Dependency: A dashed line with an arrowhead indicates a dependency relationship between two

classes, signifying that one class uses or depends on another class. Dependencies are weaker connections compared to associations.

4.2.4 Object Diagram

Explanation, Diagram

4.2.5 Component Diagram

A component diagram in UML illustrates how various components are interconnected to create larger components or software systems. It is an effective tool for representing the structure of complex systems with multiple components. By using component diagrams, developers can easily visualize the internal structure of a software system and understand how different components work together to accomplish a specific task. Its key components include:

● Component: A modular and encapsulated unit of functionality in a system that offers interfaces to interact with other components. It is represented as a rectangle with the component name inside.

● Interface: A contract between a component and its environment or other components, specifying a set of methods that can be used by other components. It is represented as a circle with the interface name inside.

● Port: A point of interaction between a component and its environment or other components. It is represented as a small square on the boundary of a component.

● Connector: A link between two components that enables communication or data exchange. It is represented as a line with optional adornments and labels.

● Dependency: A relationship between two components where one component depends on another for its implementation or functionality. It is represented as a dashed line with an arrowhead pointing from the dependent component to the independent component.

● Association: A relationship between two components that represents a connection or link. It is represented as a line connecting two components with optional directionality, multiplicity, and role names.

● Provided/Required Interface: A provided interface is an interface that a component offers to other components, while a required interface is an interface that a component needs from other components to function properly. These are represented by lollipops and half-circles respectively. Component diagrams are useful for modeling the architecture of a software system, and can help identify potential issues and improvements in the design. They can also be used to communicate the structure and behavior of a system to stakeholders, such as developers and project managers.
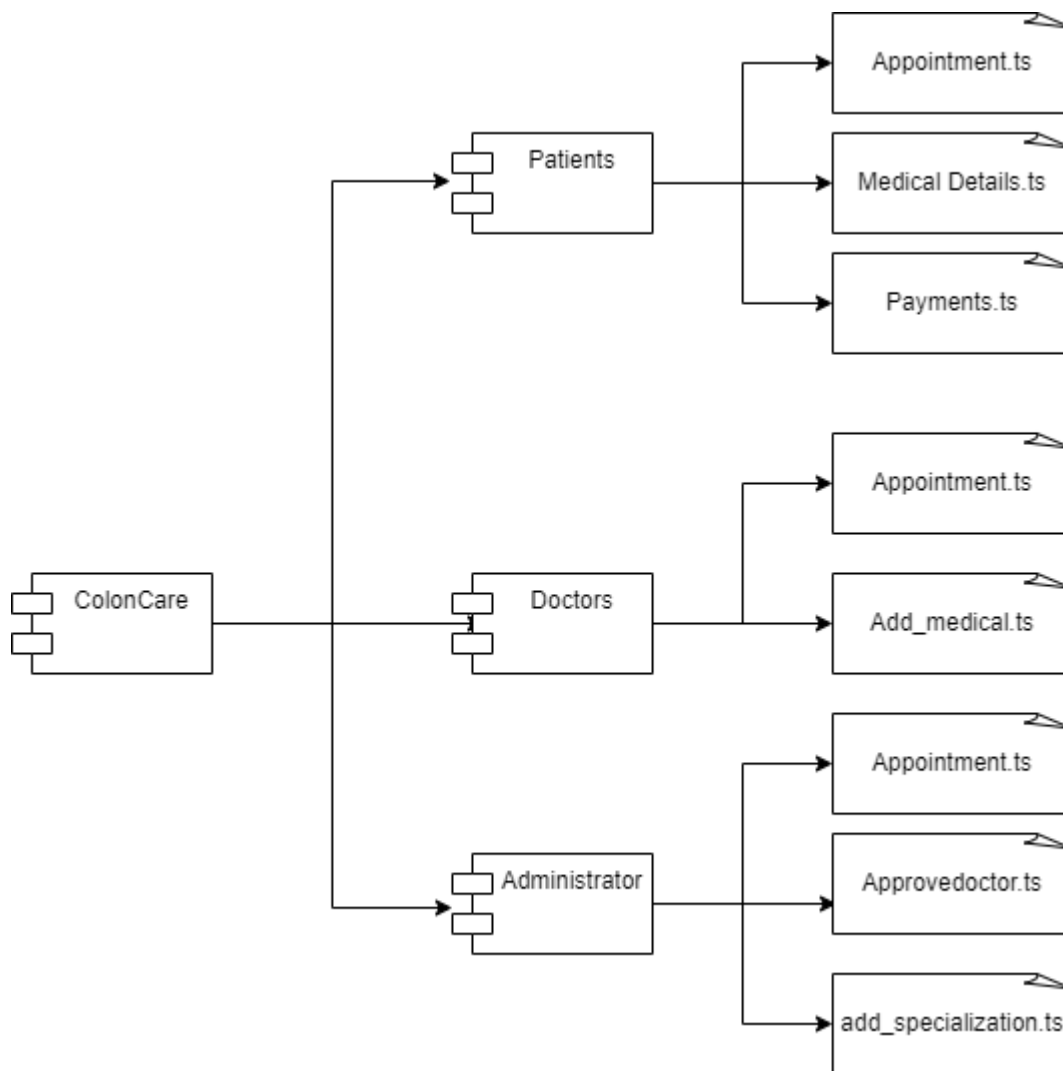
4.2.8 Deployment Diagram

A deployment diagram is a type of UML diagram that focuses on the physical hardware used to deploy software. It provides a static view of a system's deployment and involves nodes and their relationships. The deployment diagram maps the software architecture to the physical system architecture, showing how the software will be executed on nodes. Communication paths are used to illustrate the relationships between the nodes. Unlike other UML diagram types, which focus on the logical components of a system, the deployment diagram emphasizes the hardware topology. The key components of a deployment diagram are:

● Node - A node is a physical or virtual machine on which a component or artifact is deployed. It is represented by a box with the node's name inside.

● Component - A component is a software element that performs a specific function or provides a specific service. It is represented by a rectangle with the component's name inside.

 ● Artifact - An artifact is a physical piece of data that is used or produced by a component. It is represented by a rectangle with the artifact's name inside.

● Deployment Specification - A deployment specification describes how a component or artifact is deployed on a node. It includes information about the location, version, and configuration parameters of the component or artifact.

 ● Association - An association is a relationship between a node and a component or artifact that represents a deployment dependency. It is represented by a line connecting the two components with optional directionality, multiplicity, and role names.

● Communication Path - A communication path represents the connection between nodes, such as a

network connection or communication channel. It is represented by a line with optional labels and adornments.

Deployment diagrams help in visualizing the physical architecture of a system and identifying any potential issues or bottlenecks in the deployment process. They also aid in planning the deployment strategy and optimizing the use of hardware resources.
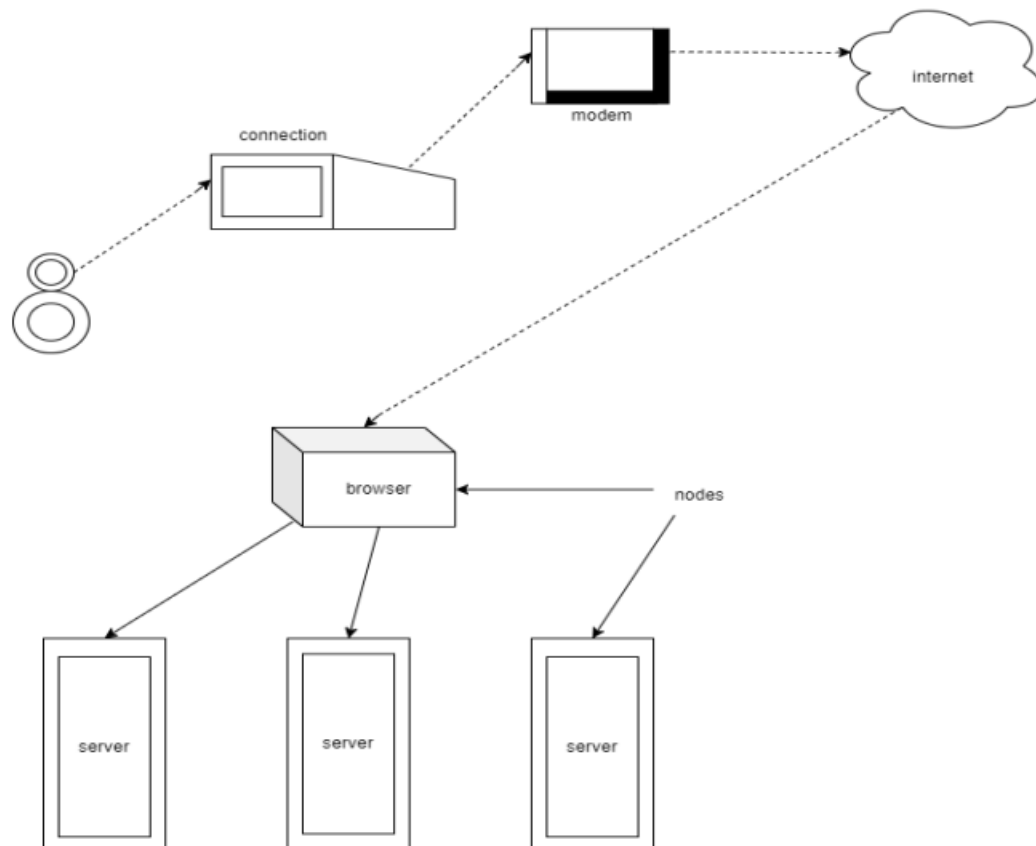
4.2.9 Collaboration Diagram
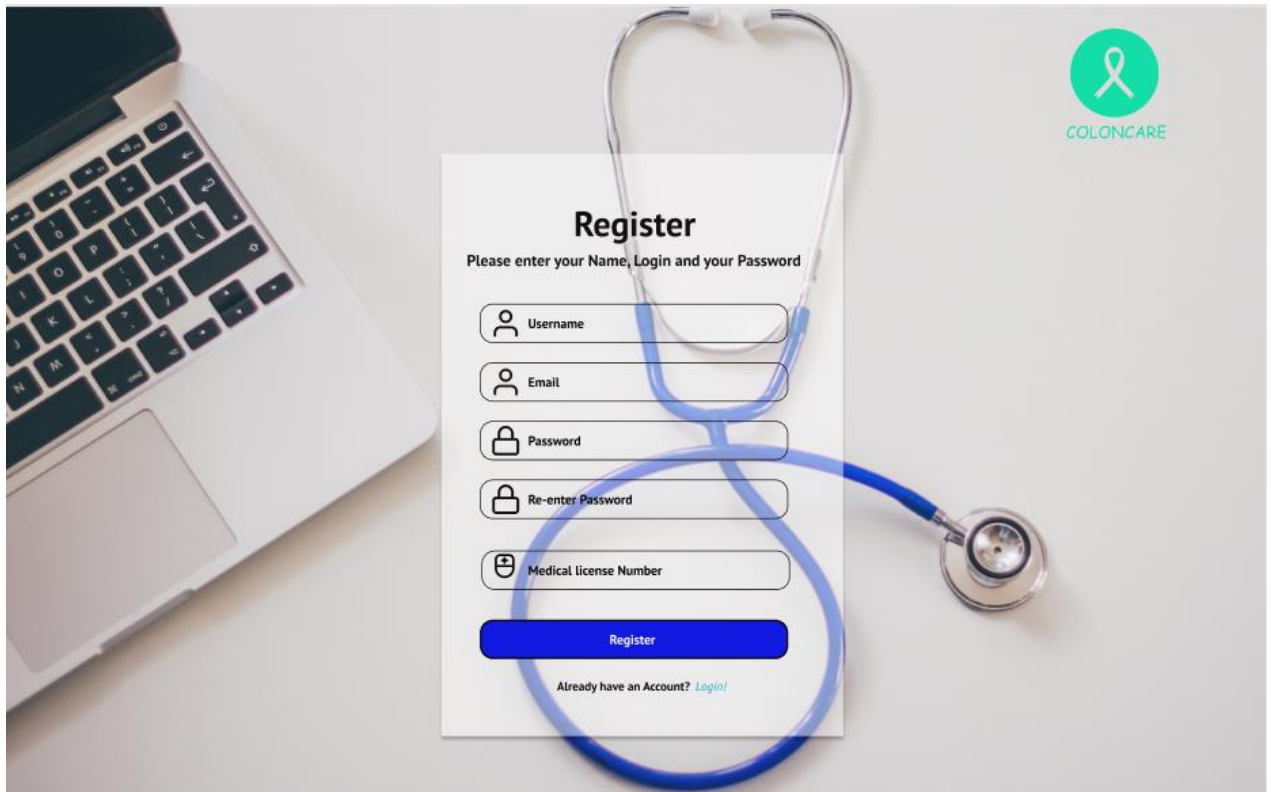
A Unified Modeling Language (UML) collaboration diagram, also known as a communication diagram, is a graphical representation used in software engineering and systems design to depict the interactions and relationships among various objects or components within a system or software application. In a collaboration diagram, objects are represented as rectangles or ovals, and lines with arrows depict the messages or communications exchanged between these objects, illustrating the flow of information and control during a specific scenario or use case. This diagram type is particularly valuable for visualizing and understanding the dynamic behavior and runtime interactions of a system, making it an essential tool for software developers and system architects in the design and documentation phases of the software development process.

1.Login
2.manage_specialization
3.get_patient details
4.get_appointment details

Admin → Server

1.authentication
2.specialization
3.patient details
4.appointment details

1.acknowlegement
2.acknowlegement category
3.acknowlegement patient details
4.acknowlegement appointment details

Database
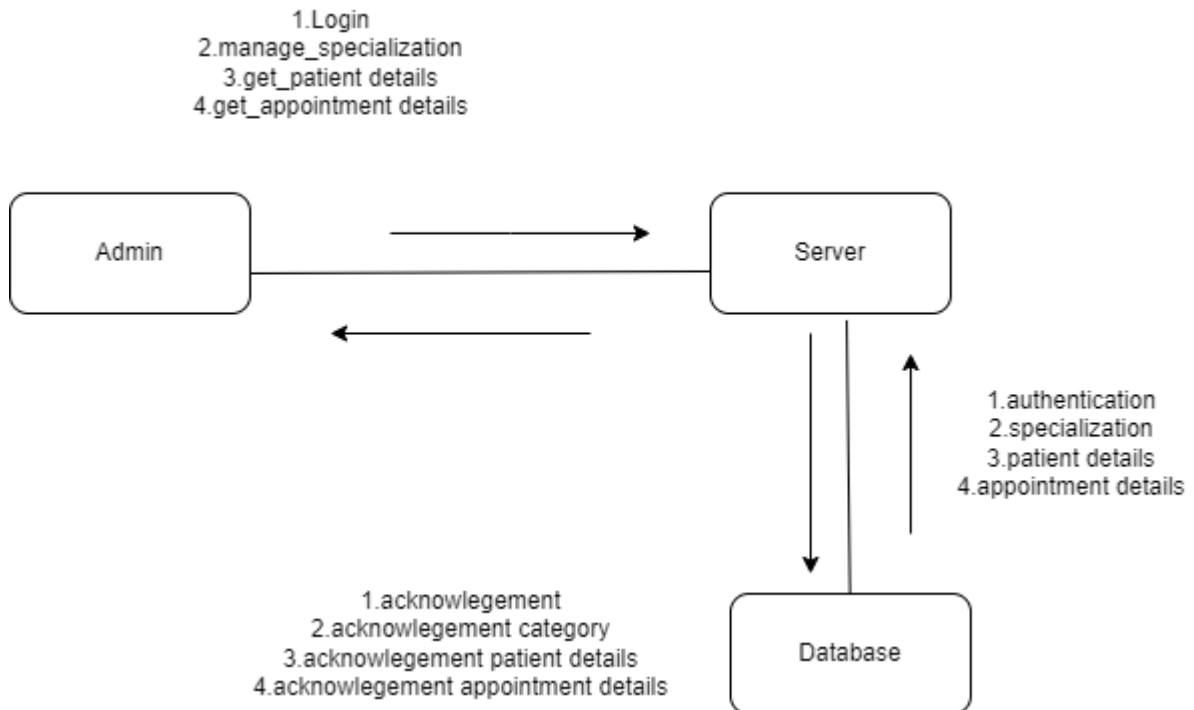
4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Login Form

Form Name: Register Form

HOME PAGE

4.3   DATABASE DESIGN

A database is an organized collection of information that's organized to enable easy accessibility, administration, and overhauls. The security of information could be a essential objective of any database. The database design process comprises of two stages. In the first stage, user requirements are gathered to create a database that meets those requirements as clearly as possible. This is known as information-level design and is carried out independently of any DBMS. In the second stage, the design is converted from an information-level design to a specific DBMS design that will be used to construct the system. This stage is known as physical-level design, where the characteristics of the specific DBMS are considered. Alongside system design, there is also database design, which aims to achieve two main goals: data integrity and data independence.

4.4.1 Relational Database Management System (RDBMS)

A relational database management system (RDBMS) is a popular type of database that organizes data into tables to facilitate relationships with other stored data sets. Tables can contain vast amounts of data, ranging from hundreds to millions of rows, each of which are referred to as records. In formal relational model language, a row is called a tuple, a column heading is an attribute, and the table is a relation. A relational database consists of multiple tables, each with its own name. Each row in a table represents a set of related values. In a relational database, relationships are already established between tables to ensure the integrity of both referential and entity relationships. A domain D is a group of atomic values, and a common way to define a domain is by choosing a data type from which the domain's data values are derived. It is helpful to give the domain a name to make it easier to understand the values it contains. Each value in a relation is atomic and cannot be further divided. In a relational database, table relationships are established using keys, with primary key and foreign key being the two most important ones. Entity integrity and referential integrity relationships can be established with these keys. Entity integrity ensures that no primary key can have null values, while referential integrity ensures that each distinct foreign key value must have a matching primary key value in the same domain. Additionally, there are other types of keys such as super keys and candidate keys.

4.4.2 Normalization

The simplest possible grouping of data is used to put them together so that future changes can be made with little influence on the data structures. The formal process of normalizing data structures in a way that reduces duplication and fosters integrity. Using the normalization technique, superfluous fields are removed and a huge table is divided into several smaller ones. Anomalies in insertion, deletion, and updating are also prevented by using it. Keys and relationships are two notions used in the standard form of data modelling. A row in a table is uniquely identified by a key. Primary keys and foreign keys are two different kinds of keys. Primary key is an element, or set of components, in a table that serves as a means of distinguishing between records from the same table. A column in a table known as a foreign key is used to uniquely identify records from other tables. Up to the third normal form, all tables have been normalized. Normalization is a process in database design that aims to organize data into proper tables and columns, making it easily correlated to the data by the user. This process eliminates data redundancy that can be a burden on computer resources. The main steps involved in normalization include: • Normalizing the data • Choosing appropriate names for tables and columns • Choosing the correct names for the data By following these steps, a developer can create a more efficient and organized database that is easier to manage and maintain.

First Normal Form

The First Normal Form (1NF) requires that each attribute in a table must contain only atomic or indivisible values. It prohibits the use of nested relations or relations within relations as attribute values within tuples. To satisfy 1NF, data must be moved into separate tables where the data is of similar type in each table, and each table should have a primary key or foreign key as required by the project. This process eliminates repeating groups of data and creates new relations for each nonatomic attribute or nested relation. A relation is in 1NF only if it satisfies the constraints that contain the primary key only

Second Normal Form

Second normal form (2NF) is a rule in database normalization that states that non-key attributes should not be functionally dependent on only the part of the primary key in a relation that has a composite primary key. In other words, each non-key attribute should depend on the entire primary key, not just a part of it. To achieve this, we need to decompose the table and create new relationships for each subkey along with their dependent attributes. It is important to maintain the relationship with the original primary key and all attributes that are fully functionally dependent on it. A relation is said to be in 2NF only if it satisfies all the 1NF conditions for the primary key and every non-primary key attribute of the relation is fully dependent only on the primary key

Third Normal Form

Third normal form (3NF) requires that a relation have no non-key attribute that is functionally determined by another non-key attribute or set of non-key attributes. This means that there should be no transitive dependency on the primary key. To achieve 3NF, we decompose the relation and set up a new relation that includes non-key attributes that functionally determine other non-key attributes. This helps eliminate any dependencies that do not just rely on the primary key. A relation is considered a relation in 3NF if it satisfies the conditions of 2NF and, moreover, the non-key attributes of the relation are not dependent on any other non-key attribute

4.4.3 Sanitization

Data sanitization is the process of removing any illegal characters or values from data. In web applications, sanitizing user input is a common task to prevent security vulnerabilities. PHP provides a built-in filter extension that can be used to sanitize and validate various types of external input such as email addresses, URLs, IP addresses, and more. These filters are designed to make data sanitization easier and faster. For example, the PHP filter extension has a function that can remove all characters except letters, digits, and certain special characters (!#$%&'*+-=?_`{|}~@.[]), as specified by a flag. Web applications often receive external input from various sources, including user input from forms, cookies, web services data, server variables, and database query results. It is important to sanitize all external input to ensure that it is safe and does not contain any malicious code or values.

4.4.4 Indexing

An index is a database structure that enhances the speed of table operations. Indexes can be created on one or more columns to facilitate quick lookups and efficient ordering of records. When creating an index, it's important to consider which columns will be used in SQL queries and to create one or more indexes on those columns. In practice, indexes are a type of table that store a primary key or index field and a pointer to each record in the actual table. Indexes are invisible to users and are only used by the database search engine to quickly locate records. The CREATE INDEX statement is used to create indexes in tables. When tables have indexes, the INSERT and UPDATE statements take longer because the database needs to insert or update the index values as well. However, the SELECT statements become faster on those tables because the index allows the database to locate records more quickly

4.5 TABLE DESIGN

1.Tbl_User

Eg.Primary key: **loginid**

Eg.Foreign key: **loginid** references table **Tbl_users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	User_id	INT(10)	PRIMARY KEY	Auto-generated unique identifier for each user
2	username	Charfield(32)	UNIQUE	A unique username for the user
3	password	Charfield(32)	NOT_NULL	The hashed password for the user
4	First_name	Charfield(32)	NOT_NULL	The user's first name
5	Last_name	Charfield(32)	NOT_NULL	The user's last name
6	email	Emailfield	UNIQUE	The user's email address
7	Date_joined	Datetime field	NOT_NULL	The date and time when the user account was created
8	Is_active	Boolean field	NOT_NULL	Indicates whether the user account is active
9	Is_staff	Boolean field	NOT_NULL	Indicates whether the user has staff/admin privileges
10	Is_superuser	Boolean field	NOT_NULL	Indicates whether the user has superuser/admin privileges

2.tbl_Userprofile
Primary key: profile_id
Foreign key: user_id references table tbl_User

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	profile_id	INT(10)	PRIMARY KEY	Auto-generated unique identifier for each user
2	User_id	Charfield(32)	FOREIGN KEY	Reference to User
3	name	Charfield(32)	NOT_NULL	The Patients name
4	email	Emailfield	UNIQUE	The Patients email address
5	Profile-pic	ImageField	NOT_NULL	The Patients profile pic
6	Phone_number	CharField (10)	NOT_NULL	The Patients phone number
7	address	TextField	NOT_NULL	The Patients address

8	age	INT (10)	NOT_NULL	The Patients age
9	place	CharField (10)	NOT_NULL	The Patients place
10	state	CharField (10)	NOT_NULL	The Patients state
11	country	CharField (10)	NOT_NULL	The Patients country
12	Blood-group	CharField (10)	NOT_NULL	The Patients blood group
13	gender	CharField (10)	NOT_NULL	The Patients gender
14	Reset_password	CharField (10)	NOT_NULL	A field for reset password

3.tbl_doctor

Primary key: doctor_id
Foreign key: user_id references table tbl_User

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	doctor_id	INT(10)	PRIMARY KEY	Auto-generated unique identifier for each user
2	User_id	Charfield(32)	FOREIGN KEY	Reference to User
3	name	Charfield(32)	NOT_NULL	The doctors name
4	email	Emailfield	UNIQUE	The doctors email address
5	Profile-pic	ImageField	NOT_NULL	The doctors profile pic
6	Phone_number	CharField (10)	NOT_NULL	The doctors phone number
7	gender	CharField (10)	NOT_NULL	Doctors gender
8	dob	DateField	NOT_NULL	Doctors date of birth
9	Add1	TextField	NOT_NULL	Doctors address
10	Add2	TextField	NOT_NULL	Doctors address
11	City	CharField (10)	NOT_NULL	Doctors city
12	State	CharField (10)	NOT_NULL	Doctors state
13	Country	CharField (10)	NOT_NULL	Doctors country
14	Postalcode	INT(10)	NOT_NULL	Doctors postalcode
15	Services	CharField (10)	NOT_NULL	Doctors services

| 16 | Specialist | CharField (10) | NOT_NULL | Doctors specialization |
| 17 | Reset password | CharField (10) | NOT_NULL | A field for reset password |

4.tbl_certification

Primary key: certification_id
Foreign key: user_id references table tbl_doctor

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	certification_id	INT(10)	PRIMARY KEY	Auto-generated unique identifier for each user
2	certification_image	INT (10)	FOREIGN KEY	Credential verification through image.
3	name	Charfield(32)	NOT_NULL	The doctors name
4	email	Emailfield	UNIQUE	The doctors email address
5	expiry_date_to	DateField	NOT_NULL	Expiration date or deadline.
6	is_approved	CharField (10)	NOT_NULL	Approval status indicator.

5.tbl_specialization

Primary key: specialization_id
Foreign key: user_id references table tbl_doctor

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	specialization_id	INT(10)	PRIMARY KEY	Identification of specific expertise
2	Specialization_name	CharField (32)	NOT_NULL	Name of specific expertise or focus.

6.tbl_appointment

Primary key: Full_name
Foreign key: user_id references table tbl_User

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Full_name	CharField (32)	PRIMARY KEY	Auto-generated unique identifier for each user
2	age	INT(10)	NOT_NULL	Patients age
3	gender	CharField (32)	NOT_NULL	Patients gender
4	Phone	INT(10)	NOT_NULL	Patients phone

5	email	CharField (32)	NOT_NULL	Patients email
6	doctor	CharField (32)	NOT_NULL	Doctors name
7	symptoms	CharField (32)	NOT_NULL	Patients symptoms
8	date	DateField	NOT_NULL	Date of appointment
9	time	INT(10)	NOT_NULL	Time of appointment

7.tbl_payment

Primary key: User_id
Foreign key: user_id references table tbl_User

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	User_id	INT (10)	FOREIGN KEY	Auto-generated unique identifier for each user
2	Payment_amount	INT(10)	NOT_NULL	Payment amount
3	Payment_status	CharField (32)	NOT_NULL	Payment status

8.tbl_patient_info

Primary key: User_id
Foreign key: user_id references table tbl_User

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	User_id	INT (10)	FOREIGN KEY	Auto-generated unique identifier for each user
2	Diagnosis date	INT(10)	NOT_NULL	Date of official diagnosis.
3	Cancer stage	CharField (32)	NOT_NULL	Cancer severity classification.
4	Biopsy results	TextField	NOT_NULL	Tissue examination findings.
5	Oncologist name	CharField (32)	NOT_NULL	Cancer specialist's name.
6	Allergy name	CharField (32)	NOT_NULL	Specific allergen identification.
7	Allergy severity	CharField (32)	NOT_NULL	Allergic reaction intensity.
8	Dosage	CharField (32)	NOT_NULL	Prescribed medication amount.
9	Frequency	CharField (32)	NOT_NULL	Treatment administration schedule.
10	Start date	DateField	NOT_NULL	Treatment initiation date.

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing involves executing a software program in a controlled manner to determine if it behaves as intended, often using verification and validation methods. Validation involves evaluating a product to ensure it complies with specifications, while verification can involve reviews, analyses, inspections, and walkthroughs. Static analysis examines the software's source code to identify issues, while dynamic analysis examines its behavior during runtime to gather information like execution traces, timing profiles, and test coverage details. Testing involves a series of planned and systematic activities that start with individual modules and progress to the integration of the entire computer-based system. The objectives of testing include identifying errors and bugs in the software, ensuring that the software functions according to its specifications, and verifying that it meets performance requirements. Testing can be performed to assess correctness, implementation efficiency, and computational complexity. A successful test is one that detects an undiscovered error, and a good test case has a high probability of uncovering such errors. Testing is crucial to achieving system testing objectives and can involve various techniques such as functional testing, performance testing, and security testing.

5.2 TEST PLAN

A test plan is a document that outlines the required steps to complete various testing methodologies. It provides guidance on the activities that need to be performed during testing. Software developers create computer programs, documentation, and associated data structures. They are responsible for testing each component of the program to ensure it meets the intended purpose. To address issues with self-evaluation, an independent test group (ITG) is often established. Testing objectives should be stated in quantifiable language, such as mean time to failure, cost to find and fix defects, remaining defect density or frequency of occurrence, and test work-hours per regression test. The different levels of testing include:

- Unit testing

- Integration testing

- Data validation testing

- Output testing

5.2.1   Unit Testing

Unit testing is a software testing technique that focuses on verifying individual components or modules of the software design. The purpose of unit testing is to test the smallest unit of software design and ensure that it performs as intended. Unit testing is typically white-box focused, and multiple components can be tested simultaneously. The component-level design description is used as a guide during testing to identify critical control paths and potential faults within the

module's perimeter. During unit testing, the modular interface is tested to ensure that data enters and exits the software unit under test properly. The local data structure is inspected to ensure that data temporarily stored retains its integrity during each step of an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once, and all error handling paths are tested to ensure that the software can handle errors correctly. Before any other testing can take place, it is essential to test data flow over a module interface. If data cannot enter and exit the system properly, all other tests are irrelevant. Another crucial duty during unit testing is the selective examination of execution pathways to anticipate potential errors and ensure that error handling paths are set up to reroute or halt work when an error occurs. Finally, boundary testing is conducted to ensure that the software operates correctly at its limits. In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. Any issues with the internal logic of the modules were fixed, and each module was tested and run separately after coding. Unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome.

5.2.2 Integration Testing

Integration testing is a systematic approach that involves creating the program structure while simultaneously conducting tests to identify interface issues. The objective is to construct a program structure based on the design that uses unit-tested components. The entire program is then tested. Correcting errors in integration testing can be challenging due to the size of the overall program, which makes it difficult to isolate the causes of the errors. As soon as one set of errors is fixed, new ones may arise, and the process may continue in an apparently endless cycle.

Once unit testing is complete for all modules in the system, they are integrated to check for any interface inconsistencies. Any discrepancies in program structures are resolved, and a unique program structure is developed.

5.2.3 Validation Testing or System Testing

The final stage of the testing process involves testing the entire software system as a whole, including all forms, code, modules, and class modules. This is commonly referred to as system testing or black box testing. The focus of black box testing is on testing the functional requirements of the software. A software engineer can use this approach to create input conditions that will fully test each program requirement. The main types of errors targeted by black box testing include incorrect or missing functions, interface errors, errors in data

structure or external data access, performance errors, initialization errors, and termination errors.

5.2.4   Output Testing or User Acceptance Testing

User acceptance testing is performed to ensure that the system meets the business requirements and user needs. It is important to involve the end users during the development process to ensure that the software aligns with their needs and expectations. During user acceptance testing, the input and output screen designs are tested with different types of test data. The preparation of test data is critical to ensure comprehensive testing of the system. Any errors identified during testing are addressed and corrected, and the corrections are noted for future reference

5.2.5 Automation Testing

Automation testing is a software testing approach that employs specialized automated testing software tools to execute a suite of test cases. Its primary purpose is to verify that the software or equipment operates precisely as intended. Automation testing identifies defects, bugs, and other issues that may arise during product development. While some types of testing, such as functional or regression testing, can be performed manually, there are numerous benefits to automating the process. Automation testing can be executed at any time of day and uses scripted sequences to evaluate the software. The results are reported, and this information can be compared to previous test runs. Automation developers typically write code in programming languages such as C#, JavaScript, and Ruby

5.2.6   Selenium Testing

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors. In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It allows for the creation of executable specifications in a humanreadable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the

requirements and helps ensure that the developed software meets the intended business goals. Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format

Example:

Test Case 1: User Login

Code

```python
from django.test import TestCase
from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'

    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        coloncare=driver.find_element(By.CSS_SELECTOR,'.header-login')
        coloncare.click()
        time.sleep(2)
```

Eg.Screenshot

```
DevTools listening on ws://127.0.0.1:61103/devtools/browser/eefc387e-2a80-41a5-bca8-99c5d1cb6e25
Login successfull
.
----------------------------------------------------------------
Ran 1 test in 12.329s

OK
Destroying test database for alias 'default'...
```

Eg.Test Report

<table>
<tr><td colspan="2">Test Case 1</td></tr>
<tr><td colspan="2">Project Name:Coloncare</td></tr>
<tr><td colspan="2" align="center">Login Test Case</td></tr>
<tr><td>Test Case ID: Test_1</td><td>Test Designed By: Devika Murali</td></tr>
<tr><td>Test Priority
(Low/Medium/High): High</td><td>Test Designed Date: 06/10/2023</td></tr>
<tr><td>Module Name: Login Screen</td><td>Test Executed By: Mr T.J Jobin</td></tr>
<tr><td>Test Title: Doctor Login</td><td>Test Execution Date: 06/10/2023</td></tr>
<tr><td colspan="2">Description: Verify login with valid email and password</td></tr>
<tr><td colspan="2">Pre-Condition: User has valid username and password</td></tr>
</table>

Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid Email	Email: hari@gmail.com	User should be able to login	User Logged in and navigated to User Dashboard	Pass
3	Provide Valid Password	Password: Hari@12 3			
4	Click on Login button				

Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database

Test Case 2: View Appointments

Code

```python
from django.test import TestCase
from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'

    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        coloncare=driver.find_element(By.CSS_SELECTOR,'.header-login')
        coloncare.click()
        time.sleep(2)


        coloncare=driver.find_element(By.CSS_SELECTOR,'.floating')
        coloncare.send_keys("hari@gmail.com")
        time.sleep(2)
        coloncare=driver.find_element(By.ID,'passwordlog')
        coloncare.send_keys("Hari@123")
        time.sleep(2)
        coloncare=driver.find_element(By.CSS_SELECTOR,'#submit')
        coloncare.click()
        time.sleep(2)
        coloncare=driver.find_element(By.ID,'app')
        coloncare.click()
        time.sleep(2)
        print("Viewed appointments")
```

Screenshot

```
DevTools listening on ws://127.0.0.1:61390/devtools/browser/f87d44fd-c943-463f-bc20-37650a9a3b9a
Viewed appointments
.
----------------------------------------------------------------
Ran 1 test in 22.662s

OK
Destroying test database for alias 'default'...
```

Test report

Test Case 2					
Project Name:Coloncare					
View Appointments **Test Case**					
Test Case ID: Test_2			**Test Designed By**: Devika Murali		
Test Priority **(Low/Medium/High): High**			**Test Designed Date:** 12/10/2023		
Module Name: Appointment Screen			**Test Executed By: Mr T.J Jobin**		
Test Title: Doctor Login			**Test Execution Date:** 12/10/2023		
Description: Display scheduled appointments.					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to doctors dashboard		Dashboard should be displayed	Login page displayed	Pass
2	Navigation to appointments page		Appointment page should be displayed	Appointment page displayed	Pass
3	View appointments				
Post-Condition: Access dashboard and view appointments.					

Test Case 3: View Patients

Code

```python
from django.test import TestCase
from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'

    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        coloncare=driver.find_element(By.CSS_SELECTOR,'.header-login')
        coloncare.click()
        time.sleep(2)


        coloncare=driver.find_element(By.CSS_SELECTOR,'.floating')
        coloncare.send_keys("hari@gmail.com")
        time.sleep(2)
        coloncare=driver.find_element(By.ID,'passwordlog')
        coloncare.send_keys("Hari@123")
        time.sleep(2)
        coloncare=driver.find_element(By.CSS_SELECTOR,'#submit')
        coloncare.click()
        time.sleep(2)
        coloncare=driver.find_element(By.ID,'app')
        coloncare.click()
        time.sleep(2)
        print("Viewed appointments")
        coloncare=driver.find_element(By.ID,'home')
        coloncare.click()
        time.sleep(2)
```

```
coloncare=driver.find_element(By.ID,'patient')
coloncare.click()
time.sleep(2)
print("Viewed Patients")
```

Screenshot

```
DevTools listening on ws://127.0.0.1:62135/devtools/browser/1cf73041-82ed-48ed-9512-2768a70dfafd
Viewed appointments
Viewed Patients
.
----------------------------------------------------------------
Ran 1 test in 35.029s

OK
Destroying test database for alias 'default'...
```

Test report

Test Case 3					
Project Name: Coloncare					
View Patients Test Case					
Test Case ID: Test_3		**Test Designed By**: Devika Murali			
Test Priority (Low/Medium/High): High		**Test Designed Date:** 12/10/2023			
Module Name: Patients Screen		**Test Executed By: Mr T.J Jobin**			
Test Title: Doctor Login		**Test Execution Date:** 12/10/2023			
Description: Display scheduled appointments.					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to doctors dashboard		Dashboard should be displayed	Login page displayed	Pass
2	Navigation to patients page		Patients page should be displayed	Patients page displayed	Pass
3	View patients				
Post-Condition: Access doctor's dashboard, view patients.					

Test Case 4: Activate User

Code

CHAPTER 6
IMPLEMENTATION

6.1INTRODUCTION

The implementation phase of a project is where the design is transformed into a functional system. It is a crucial stage in ensuring the success of the new system, as it requires gaining user confidence that the system will work effectively and accurately. User training and documentation are key concerns during this phase. Conversion may occur concurrently with user training or at a later stage. Implementation involves the conversion of a newly revised system design into an operational system. During this stage, the user department bears the primary workload, experiences the most significant upheaval, and has the most substantial impact on the existing system. Poorly planned or controlled implementation can cause confusion and chaos. Whether the new system is entirely new, replaces an existing manual or automated system, or modifies an existing system, proper implementation is essential to meet the organization's needs. System implementation involves all activities required to convert from the old to the new system. The system can only be implemented after thorough testing is done and found to be working according to specifications. System personnel evaluate the feasibility of the system. Implementation requires extensive effort in three main areas: education and training, system testing, and changeover. The implementation phase involves careful planning, investigating system and constraints, and designing methods to achieve changeover

6.2 IMPLEMENTATION PROCEDURES

Software implementation is the process of installing the software in its actual environment and ensuring that it satisfies the intended use and operates as expected. In some organizations, the software development project may be commissioned by someone who will not be using the software themselves. During the initial stages, there may be doubts about the software, but it's important to ensure that resistance does not build up. This can be achieved by: • Ensuring that active users are aware of the benefits of the new system, building their confidence in the software. • Providing proper guidance to the users so that they are comfortable using the application. Before viewing the system, users should know that the server program must be running on the server. Without the server object up and running, the intended process will not take place.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database, and call up routine that will

produce reports and perform other necessary functions.

6.2.2   Training on the Application Software

After providing the necessary basic training on computer awareness, it is essential to provide training on the new application software to the user. This training should include the underlying philosophy of using the new system, such as the flow of screens, screen design, the type of help available on the screen, the types of errors that may occur while entering data, and the corresponding validation checks for each entry, and ways to correct the data entered. Additionally, the training should cover information specific to the user or group, which is necessary to use the system or part of the system effectively. It is important to note that this training may differ across different user groups and levels of hierarchy.

6.2.3   System Maintenance

The maintenance phase is a crucial aspect of the software development cycle, as it is the time when the software is actually put to use and performs its intended functions. Proper maintenance is essential to ensure that the system remains functional, reliable, and adaptable to changes in the system environment. Maintenance activities go beyond simply identifying and fixing errors or bugs in the system. It may involve updates to the software, modifications to its functionalities, and enhancements to its performance, among other things. In essence, software maintenance is an ongoing process that requires continuous monitoring, evaluation, and improvement of the system to meet changing user needs and requirements.

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1   CONCLUSION

. In conclusion, "ColonCare" is a well-designed and user-centric web-based system that revolutionizes the process of scheduling appointments at a colon cancer center. This comprehensive platform seamlessly caters to the needs of administrators, doctors, and patients, ultimately enhancing the overall efficiency of healthcare delivery in the context of colon health and cancer care.

For administrators, "ColonCare" provides a suite of essential tools that empower them to oversee and optimize the center's operations. From managing doctor specializations to efficiently approving practitioner registrations, administrators can streamline the onboarding process for medical professionals. Additionally, they have access to appointment data and patient lists, enabling them to monitor and improve the appointment booking flow, ultimately ensuring a smoother patient experience.

Patients benefit from an intuitive and user-friendly interface that simplifies their healthcare journey. They can easily register, schedule appointments with their preferred doctors, and access detailed profiles of healthcare providers to make informed decisions about their care. The ability to view and share medical history and upload past records adds an extra layer of completeness to their healthcare profiles, ensuring comprehensive care. "ColonCare" also allows patients to manage their profiles and passwords securely, providing a personalized experience.

Doctors, too, find "ColonCare" invaluable as it simplifies appointment management and empowers them to update their professional profiles efficiently. This streamlined approach gives them the tools they need to make quick decisions on appointment requests, allowing for better patient care.

In essence, "ColonCare" represents a forward-thinking solution that places the needs and experiences of patients and healthcare providers at the forefront. By optimizing the appointment booking and management process, it enhances the overall efficiency of colon cancer center operations, ultimately contributing to improved colon health and cancer care. It is a testament to the power of technology in transforming and improving the healthcare experience for all stakeholders involved.

7.2   FUTURE SCOPE

1. **Integration with Electronic Health Records (EHRs):** Integrating "ColonCare" with EHR systems can allow for the seamless exchange of patient health data and records between healthcare providers and the platform. This integration can further improve the accuracy and completeness of patient profiles, streamlining the healthcare process.

2. **Telehealth Integration:** The incorporation of telehealth capabilities into the platform can extend its reach and offer patients the option to consult with healthcare providers remotely. This is particularly valuable for follow-up appointments, consultations, and discussions of test results.

3. **Data Analytics and Insights:** "ColonCare" can leverage data analytics to provide valuable insights to healthcare administrators and doctors. This can include data on appointment scheduling patterns, patient demographics, and the efficiency of various aspects of the center's operations. Such insights can lead to better decision-making and resource allocation.

4. **Patient Engagement Features:** Developing features that encourage patient engagement and adherence to treatment plans can improve health outcomes. This might include reminders for medication, preventive screenings, or lifestyle modifications.

5. **Expanding to Additional Healthcare Facilities:** The success of "ColonCare" in colon cancer centers could lead to its adoption in other healthcare facilities, such as gastroenterology clinics or oncology centers, broadening its application and impact.

6. **Enhanced Security and Privacy:** Given the sensitive nature of healthcare data, continuous improvements in security and privacy measures are essential to safeguard patient information. Keeping the platform compliant with evolving healthcare data regulations is crucial.

7. **Mobile Applications:** Developing mobile applications for "ColonCare" can make it even more accessible to patients and healthcare providers. Mobile apps can offer additional features, such as real-time notifications, appointment reminders, and easy access to health information on the go.

8. **AI and Machine Learning Integration:** Implementing AI and machine learning algorithms can help predict patient needs, optimize appointment scheduling, and even assist in diagnosing certain conditions based on the data collected.

9. **Patient Education and Support:** Providing patients with educational resources related to colon health and cancer prevention can empower them to take a proactive role in their healthcare. Adding features for this purpose can be valuable.

10. **Global Expansion:** If "ColonCare" proves successful in its current location, there's potential for expansion into other regions, helping more individuals access efficient colon health and cancer care services.

11. **Patient Feedback Mechanisms:** Implementing feedback mechanisms can allow patients to provide input on their experiences, helping healthcare facilities continuously improve the quality of care.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

• PankajJalote, "Software engineering: a precise approach"
• Gary B. Shelly, Harry J. Rosenblatt, "System Analysis and Design", 2009
• Ken Schwaber, Mike Beedle, Agile Software Development with Scrum, Pearson (2008)
• Roger S Pressman, "Software Engineering"
• IEEE Std 1016 Recommended Practice for Software Design Descriptions

WEBSITES:

• https://www.caritashospital.org/cancer-institute
• www.w3schools.com
• www.bootstrap.com
• https://chat.openai.com/chat
• www.jquery.com

CHAPTER 9
APPENDIX

9.1   Sample Code

LOGIN

```
{%load static%}
{% load socialaccount %}

<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login Form</title>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
<link rel="stylesheet" href="{% static 'assets/css/bootstrap.min.css' %}">

    <!-- Fontawesome CSS -->
    <link rel="stylesheet" href="{% static
'assets/plugins/fontawesome/css/fontawesome.min.css'  %}">
    <link rel="stylesheet" href="{% static 'assets/plugins/fontawesome/css/all.min.css' %}">

    <!-- Main CSS -->
    <link rel="stylesheet" href="{% static 'assets/css/style.css' %}">

</head>
<body class="account-page">

  <div class="main-wrapper">

    <!-- Header -->
    <header class="header">

    </header>
    <!-- /Header -->

    <!-- Page Content -->
    <div class="content">
      <div class="container-fluid">

        <div class="row">
          <div class="col-md-8 offset-md-2">

            <!-- Login Tab Content -->
            <div class="account-content">
              <div class="row align-items-center justify-content-center">
                <div class="col-md-7 col-lg-6 login-left">
                  <img src="{% static 'assets/img/login-banner.png' %}" class="img-fluid"
alt="Doccure Login">
                </div>
```

```html
                    <div class="col-md-12 col-lg-6 login-right">
                      <div class="login-header">
                        <h3>Login <span style="color:blue;">Coloncare</span></h3>
                      </div>
                      <form method="POST" class="register-form">
                        {% csrf_token %}
                          {% if messages %}
                          <div class="error-message" >
                          <p style="color: red; ">
                            {% for message in messages %}
                              {{ message }}
                            {% endfor %}
                          </p>
                          </div>
                          {% endif %}
                      <div class="form-group form-focus">

                          <input type="" class="form-control floating" name="username" >
                          <label class="focus-label">Email</label>
                      </div>
                      <div class="form-group form-focus">
                          <input type="password" class="form-control floating"
name="password" >
                          <label class="focus-label">Password</label>
                      </div>
                      <button class="btn btn-primary btn-block btn-lg login-btn"id="submit"
type="submit">Login</button>
                      <div class="login-or">
                          <span class="or-line"></span>
                          <span class="span-or">or</span>
                      </div>
                      <form id="signinelement2" action="{% url 'password_reset' %}"
method="post" >
                          {% csrf_token %}

                          <div class="row form-row social-login">
                            <div class="col-6">
                              <a class="forgot-link" href="{% url 'password_reset' %}">Forgot
Password ?</a>
                            </div>
                            <div class="col-6">
                              <a href="{% provider_login_url 'google'%}?next=/" class="btn
btn-google btn-block"><i class="fab fa-google mr-1"></i> Login</a>
                            </div>
                          </div>
                          </form>

                          <div class="text-center dont-have">Don't have an account? <a href="{%
url 'register' %}">Register</a></div>
                        </form>
                    </div>
```

```
                </div>
              </div>
              <!-- /Login Tab Content -->

            </div>
          </div>

        </div>

      </div>
      <!-- /Page Content -->




    </div>
    <!-- /Main Wrapper -->

    <!-- jQuery -->
    <script src="{% static 'assets/js/jquery.min.js' %}" ></script>

    <!-- Bootstrap Core JS -->
    <script src="{% static 'assets/js/popper.min.js' %}"></script>
    <script src="{% static 'assets/js/bootstrap.min.js' %}"></script>

    <!-- Custom JS -->
    <script src="{% static 'assets/js/script.js' %}"></script>
  <!-- Sing in  Form -->
</body>
</html>
```

PATIENT REGSITER

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Register-Page</title>
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
   <link rel="stylesheet" href="{% static 'assets/css/bootstrap.min.css' %}">
   <link rel="stylesheet" href="{% static 'assets/plugins/fontawesome/css/fontawesome.min.css'
%}">
   <link rel="stylesheet" href="{% static 'assets/plugins/fontawesome/css/all.min.css' %}">
   <link rel="stylesheet" href="{% static 'assets/css/style.css' %}">
    <!-- Font Icon -->
  <link rel="stylesheet" href="{% static 'fonts/material-icon/css/material-design-iconic-
font.min.css' %}">
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

  <!-- Main css -->
  {% comment %} <link rel="stylesheet" href="static/css/style.css"> {% endcomment %}
```

```
</head>

<script>
   $(document).ready(function() {
      // Name validation
      $("#full_name").keyup(function() {
         var fullName = $(this).val();
         var fullNamePattern = /^[A-Z][a-z]* [A-Z][a-z]*$/; // First and last name pattern

         if (!fullNamePattern.test(fullName)) {
            $("#fnspan").text("Name should start with a capital letter, followed by lowercase letters,
for both first and last names.");
         } else {
            $("#fnspan").text("");
         }
      });

      $("#email").keyup(function() {
         var email = $(this).val();
         var emailPattern = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

         if (!emailPattern.test(email)) {
            $("#emspan").text("Enter a valid email address, e.g.. user@gmail.com");
         } else {
            $("#emspan").text("");
         }
      });

      $("#password").keyup(function() {
         validatePasswordStrength();
      });

      $("#confirm_password").keyup(function() {
         validateConfirmPassword();
      });

      function validatePasswordStrength() {
         var password = $("#password").val();
         var hasUpperCase = /[A-Z]/.test(password);
         var hasLowerCase = /[a-z]/.test(password);
         var hasDigits = /[0-9]/.test(password);
         var hasSpecialChars = /[!@#$%^&*()_+{}[\]:;<>,.?~\/-]/.test(password);

         if (password.length < 8) {
            $("#pspan").text("Password must be at least 8 characters long.");
         } else if (!(hasUpperCase && hasLowerCase && hasDigits && hasSpecialChars)) {
            $("#pspan").text("Password must contain at least one uppercase letter, one lowercase
letter, one digit, and one special character.");
         } else {
            $("#pspan").text("");
```

```
        }

        // Validate confirm password when password changes
        validateConfirmPassword();
    }

    function validateConfirmPassword() {
        var password = $("#password").val();
        var confirmPassword = $("#confirm_password").val();

        if (confirmPassword.length > 0 && password !== confirmPassword) {
            $("#cspan").text("Passwords do not match.");
        } else {
            $("#cspan").text("");
        }
    }
});
</script>
```

```
<body>
<!-- Sign up form -->
<div class="content">
  <div class="container-fluid">

    <div class="row">
      <div class="col-md-8 offset-md-2">

        <!-- Register Content -->
        <div class="account-content">
          <div class="row align-items-center justify-content-center">
            <div class="col-md-7 col-lg-6 login-left">
              <img src="{% static 'assets/img/login-banner.png' %}" class="img-fluid"
alt="Doccure Register">
            </div>
            <div class="col-md-12 col-lg-6 login-right">
              <div class="login-header">
                <h3>Patient Register <a href="{% url 'DoctorReg' %}">Are you a
Doctor?</a></h3>
              </div>

              <!-- Register Form -->
              <form  href="{% url 'login_page' %}" method="POST" class="register-form">
                {% csrf_token %}

                <div class="form-group form-focus">
                  <input type="text" class="form-control floating"name="name"
id="full_name">
                  <label class="focus-label">Name</label>
                  <span id="fnspan" style="font-size: small; color:red; margin-left:10px;
```

```
"></span> <br>

                                </div>
                                <div class="form-group form-focus">
                                    <input type="email" class="form-control floating"name="email"
id="email">

                                    <label class="focus-label">Email</label>
                                    <span id="emspan" style="font-size: small; color:red; margin-left:10px;
"></span> <br>

                                </div>
                                <div class="form-group form-focus">
                                    <input type="password" class="form-control floating"name="password"
id="password">

                                    <label class="focus-label">Password</label>
                                    <span id="pspan" style="font-size: small; color:red; margin-left:10px;
"></span> <br>

                                </div>
                                <div class="form-group form-focus">
                                    <input type="password" class="form-control
floating"name="confirm_password" id="confirm_password">
                                    <label class="focus-label">Confirm Password</label>
                                    <span id="cspan" style="font-size: small; color:red; margin-left:10px;
"></span> <br>

                                </div>
                                <div class="text-right">
                                    <a class="forgot-link" href="{% url 'login_page' %}">Already have an
account?</a>
                                </div>
                                <button class="btn btn-primary btn-block btn-lg login-btn"
type="submit">Signup</button>
                            </form>
                            <!-- /Register Form -->

                        </div>
                    </div>
                </div>
                <!-- /Register Content -->

            </div>
        </div>

    </div>
    <script src="{% static 'assets/js/jquery.min.js' %}" ></script>

    <!-- Bootstrap Core JS -->
    <script src="{% static 'assets/js/popper.min.js' %}"></script>
    <script src="{% static 'assets/js/bootstrap.min.js' %}"></script>
```

```
    <!-- Custom JS -->
    <script src="{% static 'assets/js/script.js' %}"></script>
</div>
</body>
</html>
```

ADMIN DASHBOARD

```
{% load static %}
<!DOCTYPE html>
<html lang="en">

<!-- Mirrored from dreamguys.co.in/demo/doccure/admin/index.html by HTTrack Website
Copier/3.x [XR&CO'2014], Sat, 30 Nov 2019 04:12:20 GMT -->
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=0">
    <title>Admin - Dashboard</title>
    <link rel="stylesheet" href="{% static 'assets/css/bootstrap.min.css'%}">

    <!-- Favicon -->
    <link rel="shortcut icon" href="{% static 'admin/assets/img/favicon.png' %}">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="{% static 'admin/assets/css/bootstrap.min.css' %}">

    <!-- Fontawesome CSS -->
    <link rel="stylesheet" href="{% static 'admin/assets/css/font-awesome.min.css' %}">

    <!-- Feathericon CSS -->
    <link rel="stylesheet" href="{% static 'admin/assets/css/feathericon.min.css' %}">

    <link rel="stylesheet" href="{% static 'admin/assets/plugins/morris/morris.css' %}">

    <!-- Main CSS -->
    <link rel="stylesheet" href="{% static 'admin/assets/css/style.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/bootstrap.min.css'%}">

  <!-- Fontawesome CSS -->
  <link rel="stylesheet" href="{% static
'assets/plugins/fontawesome/css/fontawesome.min.css'%}">
  <link rel="stylesheet" href="{% static 'assets/plugins/fontawesome/css/all.min.css'%}">

  <!-- Main CSS -->
  <link rel="stylesheet" href="{% static 'assets/css/style.css'%}">

    <!--[if lt IE 9]>
       <script src="assets/js/html5shiv.min.js"></script>
       <script src="assets/js/respond.min.js"></script>
    <![endif]-->
    <style>
```

```
       i.fa.fa-user-circle {
          color: black;
       }
       /* Style for the user menu */
.user-menu {
   list-style: none;
}

/* Style for the user icon */
.user-icon {
   width: 40px;
   height: 40px;
   background-color: #3498db;
   border-radius: 50%;
   display: flex;
   align-items: center;
   justify-content: center;
   color: #fff;
}

/* Style for the user welcome message */
.user-welcome {
   margin: 10px 0;
   font-weight: bold;
}

/* Style for the dropdown */
.dropdown-menu {
   box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
   border: none;
}

    </style>

  </head>
  <body>

    <!-- Main Wrapper -->
    <div class="main-wrapper">

       <!-- Header -->
       <div class="header">

          <!-- Logo -->
          <div class="header-left">
            <a href="index.html" class="logo">
               <img src="{% static 'assets/img/logo.png' %}" class="img-fluid" alt="Logo">
            </a>
            <a href="index.html" class="logo logo-small">
               <img src="{% static 'assets/img/logo.png' %}" class="img-fluid" alt="Logo"
width="30" height="100">
```

```
          </a>
        </div>
        <!-- /Logo -->

        {% comment %} <a href="javascript:void(0);" id="toggle_btn">
          <i class="fe fe-text-align-left"></i>
        </a> {% endcomment %}

        {% comment %} <div class="top-nav-search">
          <form>
            <input type="text" class="form-control" placeholder="Search here">
            <button class="btn" type="submit"><i class="fa fa-search"></i></button>
          </form>
        </div> {% endcomment %}

        <!-- Mobile Menu Toggle -->
        <a class="mobile_btn" id="mobile_btn">
          <i class="fa fa-bars"></i>
        </a>
        <!-- /Mobile Menu Toggle -->

        <ul class="nav user-menu">
          <!-- User Menu -->
          <li class="nav-item dropdown has-arrow">
            <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown">
              <i class="fa fa-user-circle"></i>
            </a>
            <div class="dropdown-menu">
              {% if user.is_authenticated %}
              <ul class="submenu">
                <li>Welcome,{{ user.username }}</li>
                <li><a class="dropdown-item" href="{% url 'loggout' %}">Logout</a></li>
              </ul>
            </div>
          </li>


          {% else %}
          <li><a href=""></a></li>
          {% endif %}
        </ul>


    <!-- Sidebar -->
    <div class="sidebar" id="sidebar">
      <div class="sidebar-inner slimscroll">
        <div id="sidebar-menu" class="sidebar-menu">
          <ul>
            <li class="menu-title">
              <span>Main</span>
            </li>
```

```
            <li class="active">
              <a href="{% url 'admins_index' %}"><i class="fe fe-home"></i>
<span>Dashboard</span></a>
            </li>
            <li>
              <a href="{% url 'admins_registereduser' %}"><i class="fe fe-user"></i>
<span>Registered Users</span></a>
            </li>
            <li>
              <a href="{% url 'admins_appointmentlist' %}"><i class="fe fe-layout"></i>
<span>Appointments</span></a>
            </li>
            <li>
              <a href="{% url 'admins_specialities' %}"><i class="fe fe-users"></i>
<span>Specialities</span></a>
            </li>
            <li>
              <a href="{% url 'admins_dashlegal' %}"><i class="fe fe-user-plus"></i>
<span>Doctors</span></a>
            </li>
            <li>
              <a href="{% url 'admins_patientlist' %}"><i class="fe fe-users"></i>
<span>Patients</span></a>
            </li>
             {% comment %} <li>
              <a href="{% url 'admins_bookscreening' %}"></i> <span>Screening
Timeslots</span></a>
            </li>  {% endcomment %}


        </div>
      </div>
    </div>
    <!-- /Sidebar -->

    <!-- Page Wrapper -->
    <div class="page-wrapper">

      <div class="content container-fluid">

        <!-- Page Header -->
        <div class="page-header">
          <div class="row">
            <div class="col-sm-12">
              <h3 class="page-title">Welcome Admin!</h3>
              <ul class="breadcrumb">
                <li class="breadcrumb-item active">Dashboard</li>
              </ul>
            </div>
          </div>
        </div>
        <!-- /Page Header -->
```

```html
<div class="row">
   <div class="col-xl-3 col-sm-6 col-12">
      <div class="card">
         <div class="card-body">
            <div class="dash-widget-header">
               <span class="dash-widget-icon text-primary border-primary">
                  <i class="fe fe-users"></i>
               </span>
               <div class="dash-count">
                  <h3>{{ doctors_count }}</h3>
               </div>
            </div>
            <div class="dash-widget-info">
               <h6 class="text-muted">Doctors</h6>
               <div class="progress progress-sm">
                  <div class="progress-bar bg-primary w-50"></div>
               </div>
            </div>
         </div>
      </div>
   </div>
   <div class="col-xl-3 col-sm-6 col-12">
      <div class="card">
         <div class="card-body">
            <div class="dash-widget-header">
               <span class="dash-widget-icon text-success">
                  <i class="fe fe-credit-card"></i>
               </span>
               <div class="dash-count">
                  <h3>{{ patients_count }}</h3>
               </div>
            </div>
            <div class="dash-widget-info">
               <h6 class="text-muted">Patients</h6>
               <div class="progress progress-sm">
                  <div class="progress-bar bg-success w-50"></div>
               </div>
            </div>
         </div>
      </div>
   </div>
</div>

   <div class="col-xl-3 col-sm-6 col-12">
      <div class="card">
         <div class="card-body">
            <div class="dash-widget-header">
               <span class="dash-widget-icon text-danger border-danger">
                  <i class="fe fe-money"></i>
               </span>
```

```
                          <div class="dash-count">
                             <h3>485</h3>
                          </div>
                       </div>
                       <div class="dash-widget-info">

                          <h6 class="text-muted">Appointment</h6>
                          <div class="progress progress-sm">
                             <div class="progress-bar bg-danger w-50"></div>
                          </div>
                       </div>
                    </div>
                 </div>
              </div>
           {% comment %} <div class="col-xl-3 col-sm-6 col-12">
              <div class="card">
                 <div class="card-body">
                    <div class="dash-widget-header">
                       <span class="dash-widget-icon text-warning border-warning">
                          <i class="fe fe-folder"></i>
                       </span>
                       <div class="dash-count">
                          <h3>$62523</h3>
                       </div>
                    </div>
                    <div class="dash-widget-info">

                       <h6 class="text-muted">Revenue</h6>
                       <div class="progress progress-sm">
                          <div class="progress-bar bg-warning w-50"></div>
                       </div>
                    </div>
                 </div>
              </div>
           </div>
            {% endcomment %}

        </div>

             <!-- /Recent Orders -->

          </div>
        </div>

     </div>
   </div>
   <!-- /Page Wrapper -->

 </div>
 <!-- /Main Wrapper -->
```

```
    <!-- jQuery -->
    <script src="assets/js/jquery-3.2.1.min.js"></script>

    <!-- Bootstrap Core JS -->
    <script src="assets/js/popper.min.js"></script>
    <script src="assets/js/bootstrap.min.js"></script>

    <!-- Slimscroll JS -->
    <script src="assets/plugins/slimscroll/jquery.slimscroll.min.js"></script>

    <script src="assets/plugins/raphael/raphael.min.js"></script>
    <script src="assets/plugins/morris/morris.min.js"></script>
    <script src="assets/js/chart.morris.js"></script>

    <!-- Custom JS -->
    <script  src="assets/js/script.js"></script>
    <script src="{% static 'assets/js/jquery.min.js'%} " ></script>

<!-- Bootstrap Core JS -->
<script src="{% static 'assets/js/popper.min.js'%}">
</script>
<script src="{% static 'assets/js/bootstrap.min.js'%}"></script>

<!-- Slick JS -->
<script src="{% static 'assets/js/slick.js'%}"></script>

<!-- Custom JS -->
<script src="{% static 'assets/js/script.js'%}"></script>

    </body>
```

<!-- Mirrored from dreamguys.co.in/demo/doccure/admin/index.html by HTTrack Website Copier/3.x [XR&CO'2014], Sat, 30 Nov 2019 04:12:34 GMT -->
</html>

ADD SPECIALIZATION


```
{% load static %}
<!DOCTYPE html>
<html lang="en">

<!-- Mirrored from dreamguys.co.in/demo/doccure/admin/index.html by HTTrack Website Copier/3.x
[XR&CO'2014], Sat, 30 Nov 2019 04:12:20 GMT -->
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=0">
    <title>Admin - Dashboard</title>
    <link rel="stylesheet" href="{% static 'assets/css/bootstrap.min.css'%}">
```

```
<!-- Favicon -->
<link rel="shortcut icon" href="{% static 'admin/assets/img/favicon.png' %}">

<!-- Bootstrap CSS -->
<link rel="stylesheet" href="{% static 'admin/assets/css/bootstrap.min.css' %}">

<!-- Fontawesome CSS -->
<link rel="stylesheet" href="{% static 'admin/assets/css/font-awesome.min.css' %}">

<!-- Feathericon CSS -->
<link rel="stylesheet" href="{% static 'admin/assets/css/feathericon.min.css' %}">

<link rel="stylesheet" href="{% static 'admin/assets/plugins/morris/morris.css' %}">

<!-- Main CSS -->
<link rel="stylesheet" href="{% static 'admin/assets/css/style.css' %}">

<!--[if lt IE 9]>
    <script src="assets/js/html5shiv.min.js"></script>
    <script src="assets/js/respond.min.js"></script>
<![endif]-->
</head>
<body>

    <!-- Main Wrapper -->
    <div class="main-wrapper">

        <!-- Header -->
        <div class="header">

            <!-- Logo -->
            <div class="header-left">
                <a href="index.html" class="logo">
                    <img src="{% static 'assets/img/logo.png' %}" class="img-fluid" alt="Logo">
                </a>
                <a href="index.html" class="logo logo-small">
                    <img src="{% static 'assets/img/logo.png' %}" class="img-fluid" alt="Logo" width="30"
height="100">
                </a>
            </div>
            <!-- /Logo -->

            {% comment %} <a href="javascript:void(0);" id="toggle_btn">
                <i class="fe fe-text-align-left"></i>
            </a> {% endcomment %}

            {% comment %} <div class="top-nav-search">
                <form>
                    <input type="text" class="form-control" placeholder="Search here">
                    <button class="btn" type="submit"><i class="fa fa-search"></i></button>
                </form>
            </div> {% endcomment %}

            <!-- Mobile Menu Toggle -->
            <a class="mobile_btn" id="mobile_btn">
```

```
                <i class="fa fa-bars"></i>
        </a>
        <!-- /Mobile Menu Toggle -->

        <!-- Header Right Menu -->
        <ul class="nav user-menu">

            <!-- Notifications -->
            <li class="nav-item dropdown noti-dropdown">
                <a href="#" class="dropdown-toggle nav-link" data-toggle="dropdown">
                    <i class="fe fe-bell"></i> <span class="badge badge-pill">3</span>
                </a>
                <div class="dropdown-menu notifications">
                    <div class="topnav-dropdown-header">
                        <span class="notification-title">Notifications</span>
                        <a href="javascript:void(0)" class="clear-noti"> Clear All </a>
                    </div>
                    <div class="noti-content">
                        <ul class="notification-list">
                            <li class="notification-message">
                                <a href="#">
                                    <div class="media">
                                        <span class="avatar avatar-sm">
                                            <img class="avatar-img rounded-circle" alt="User Image"
src="assets/img/doctors/doctor-thumb-01.jpg">
                                        </span>
                                        <div class="media-body">
                                            <p class="noti-details"><span class="noti-title">Dr. Ruby Perrin</span>
Schedule <span class="noti-title">her appointment</span></p>
                                            <p class="noti-time"><span class="notification-time">4 mins
ago</span></p>
                                        </div>
                                    </div>
                                </a>
                            </li>
                            <li class="notification-message">
                                <a href="#">
                                    <div class="media">
                                        <span class="avatar avatar-sm">
                                            <img class="avatar-img rounded-circle" alt="User Image"
src="assets/img/patients/patient1.jpg">
                                        </span>
                                        <div class="media-body">
                                            <p class="noti-details"><span class="noti-title">Charlene Reed</span>
has booked her appointment to <span class="noti-title">Dr. Ruby Perrin</span></p>
                                            <p class="noti-time"><span class="notification-time">6 mins
ago</span></p>
                                        </div>
                                    </div>
                                </a>
                            </li>
                            <li class="notification-message">
                                <a href="#">
                                    <div class="media">
                                        <span class="avatar avatar-sm">
```

```
                                <img class="avatar-img rounded-circle" alt="User Image"
src="assets/img/patients/patient2.jpg">
                            </span>
                            <div class="media-body">
                            <p class="noti-details"><span class="noti-title">Travis Trimble</span> sent
a amount of $210 for his <span class="noti-title">appointment</span></p>
                            <p class="noti-time"><span class="notification-time">8 mins
ago</span></p>
                            </div>
                        </div>
                    </a>
                </li>
                <li class="notification-message">
                    <a href="#">
                        <div class="media">
                            <span class="avatar avatar-sm">
                                <img class="avatar-img rounded-circle" alt="User Image"
src="assets/img/patients/patient3.jpg">
                            </span>
                            <div class="media-body">
                            <p class="noti-details"><span class="noti-title">Carl Kelly</span> send a
message <span class="noti-title"> to his doctor</span></p>
                            <p class="noti-time"><span class="notification-time">12 mins
ago</span></p>
                            </div>
                        </div>
                    </a>
                </li>
            </ul>
        </div>
        <div class="topnav-dropdown-footer">
            <a href="#">View all Notifications</a>
        </div>
    </div>
</li>
<!-- /Notifications -->

<!-- User Menu -->
<li class="nav-item dropdown has-arrow">
    <a href="#" class="dropdown-toggle nav-link" data-toggle="dropdown">
        <span class="user-img"><img class="rounded-circle" src="assets/img/profiles/avatar-
01.jpg" width="31" alt="Ryan Taylor"></span>
    </a>
    <div class="dropdown-menu">
        <div class="user-header">
            <div class="avatar avatar-sm">
                <img src="assets/img/profiles/avatar-01.jpg" alt="User Image" class="avatar-img
rounded-circle">
            </div>
            <div class="user-text">
                {% if user.is_authenticated %}
                <h6>{{ user.username }} {{ user.username }}</h6>
                {% endif %}
                {% comment %} <h6>Ryan Taylor</h6> {% endcomment %}
                <p class="text-muted mb-0">Administrator</p>
```

```
                  </div>
                </div>
                <a class="dropdown-item" href="profile.html">My Profile</a>
                <a class="dropdown-item" href="settings.html">Settings</a>
                <a class="dropdown-item" href="login.html">Logout</a>
              </div>
            </li>
            <!-- /User Menu -->

          </ul>
          <!-- /Header Right Menu -->

        </div>
        <!-- /Header -->

        <!-- Sidebar -->
        <div class="sidebar" id="sidebar">
          <div class="sidebar-inner slimscroll">
            <div id="sidebar-menu" class="sidebar-menu">
              <ul>
                <li class="menu-title">
                  <span>Main</span>
                </li>
                <li >
                  <a href="{% url 'admins_index' %}"><i class="fe fe-home"></i>
<span>Dashboard</span></a>
                </li>
                <li>
                  <a href="{% url 'admins_registereduser' %}"><i class="fe fe-user"></i>
<span>Registered Users</span></a>
                </li>
                <li>
                  <a href="appointment-list.html"><i class="fe fe-layout"></i>
<span>Appointments</span></a>
                </li>
                <li class="active">
                  <a href="{% url 'admins_specialities' %}"><i class="fe fe-users"></i>
<span>Specialities</span></a>
                </li>
                <li>
                  <a href="{% url 'admins_dashlegal' %}"><i class="fe fe-user-plus"></i>
<span>Doctors</span></a>
                </li>
                <li>
                  <a href="{% url 'admins_patientlist' %}"><i class="fe fe-users"></i>
<span>Patients</span></a>
                </li>

            </div>
          </div>
        </div>
        <!-- /Sidebar -->
```

```
<!-- Add Modal -->
    <div class="modal fade" id="Add_Specialities_details" aria-hidden="true" role="dialog">
        <div class="modal-dialog modal-dialog-centered" role="document" >
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title">Add Specialities</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                <div class="modal-body">
                    <form>
                        <div class="row form-row">
                            <div class="col-12 col-sm-6">
                                <div class="form-group">
                                    <label>Specialities</label>
                                    <input type="text" class="form-control">
                                </div>
                            </div>

                        </div>
                        <button type="submit" class="btn btn-primary btn-block">Save Changes</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
    <!-- /ADD Modal -->
    <!-- Edit Specialization Modal -->
<div class="modal fade" id="edit_specialities_details_{{ specialization.id }}" tabindex="-1" role="dialog"
aria-labelledby="editSpecializationModal" aria-hidden="true">
   <div class="modal-dialog modal-dialog-centered" role="document">
      <div class="modal-content">
        <div class="modal-header">
           <h5 class="modal-title">Edit Specialization</h5>
           <button type="button" class="close" data-dismiss="modal" aria-label="Close">
              <span aria-hidden="true">&times;</span>
           </button>
        </div>
        <div class="modal-body">
           <!-- Edit Specialization Form -->
           <form method="POST" action="{% url 'admins_specialities' %}">
              {% csrf_token %}
              <input type="hidden" name="specialization_id" value="{{ specialization.id }}">
              <div class="form-group">
                 <label>Specialization Name</label>
                 <input type="text" class="form-control" name="specialization_name" value="{{
specialization.specialization_name }}" required>
              </div>
              <button type="submit" class="btn btn-primary">Save Changes</button>
           </form>
           <!-- /Edit Specialization Form -->
        </div>
      </div>
   </div>
```

```
</div>

        <div class="page-wrapper">
          <div class="content container-fluid">

            <!-- Page Header -->
            <div class="page-header">
              <div class="row">
                <div class="col-sm-7 col-auto">
                  <h3 class="page-title">Specialities</h3>
                  <ul class="breadcrumb">
                    <li class="breadcrumb-item"><a href="index.html">Dashboard</a></li>
                    <li class="breadcrumb-item active">Specialities</li>
                  </ul>
                </div>

              </div>
            </div>
            <!-- /Page Header -->

            <div class="row">
              <div class="col-sm-12">
                <div class="card">
                  <div class="card-body">
                    <!-- Add Specialization Form -->
                    <form method="POST" action="{% url 'admins_specialities' %}">
                      {% csrf_token %}
                      <input type="hidden" name="action" value="add">
                      <div class="row form-row">
                        <div class="col-12 col-sm-6">
                          <div class="form-group">
                            <label for="specialization_name">Specialization Name</label>
                            <input type="text" class="form-control" id="specialization_name"
name="specialization_name" required
                                    minlength="3" maxlength="50" pattern="[A-Za-z\s]+" title="Only
letters and spaces are allowed">
                            <div class="invalid-feedback">
                              Please enter a valid specialization name (3-50 characters, only letters
and spaces).
                            </div>
                          </div>
                        </div>
                        <div class="col-12">
                          <button type="submit" class="btn btn-primary">Add
Specialization</button>
                        </div>
                      </div>
                    </form>

                    <!-- /Add Specialization Form -->
                    <div class="table-responsive">
                      <table class="datatable table table-hover table-center mb-0">
                        <thead>
                          <tr>
```

```
                                              <th>Specialization Name</th>
                                              <th class="text-right">Actions</th>
                                          </tr>
                                      </thead>
                                      <tbody>
                                        {% for specialization in specializations %}
                                        <tr>

                                          <td>
                                             <h2 class="table-avatar">
                                                <a href="#">
                                                   {{ specialization.specialization_name }}
                                                </a>
                                             </h2>
                                          </td>

                                             <td class="text-right">
                                                <div class="actions">

                                                        <!-- Button to trigger the modal -->
<a href="" class="btn btn-sm bg-success-light editSpecializationButton" data-toggle="modal" data-
target="#edit_specialities_details" data-specialization-id="{{ specialization.id }}">
   <i class="fe fe-pencil"></i> Edit
</a>

<!-- Modal Definition -->
<div class="modal fade" id="edit_specialities_details" tabindex="-1" role="dialog" aria-
labelledby="editModalLabel" aria-hidden="true">
   <div class="modal-dialog" role="document">
      <div class="modal-content">
         <!-- Modal Header -->
         <div class="modal-header">
            <h5 class="modal-title" id="editModalLabel">Edit Specialization</h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
               <span aria-hidden="true">&times;</span>
            </button>
         </div>

         <!-- Modal Body with Edit Form -->
         <form method="post"  action="{% url 'edit_specialization' specialization.id %}">
            {% csrf_token %}
         <div class="modal-body">

             <div class="form-group">
                {% comment %} <h1>{{ specialization.id }}</h1> {% endcomment %}
                <label for="specializationName">Enter Specialization Name</label>
                <input type="text" class="form-control" id="specializationName" value="{{
specialization.specialization_name }}" placeholder="Enter specialization name"
name="specialization_name">
                <input type="hidden" name="specialization_id" id="specializationId" />
                <input type="hidden" name="action" value="edit">
             </div>
```

```
        </div>

        <!-- Modal Footer with Save and Close Buttons -->
        <div class="modal-footer">
           {% comment %} <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button> {% endcomment %}
              <button type="submit" class="btn btn-primary" id="saveChangesBtn">Save changes</button>
        </div>
     </form>

     </div>
   </div>
</div>
<!-- Delete Button -->

 <a class="btn btn-sm bg-danger-light delete-specialization"
   href="{% url 'delete_specialization' specialization.id %} "
   data-specialization-id="{{ specialization.id }}">
   <i class="fe fe-trash"></i> Delete
</a>

                                    </div>
                                 </td>
                              </tr>
                           {% endfor %}
                        </tbody>
                     </table>
                  </div>
                  <!-- /Specialization List -->
               </div>
            </div>
         </div>
         </div>
         <!-- /Page Wrapper -->
     <script>


     </script>
   <!-- jQuery -->
   <script src="{% static 'assets/js/jquery.min.js'%}" ></script>

   <script src="{% static 'admin/assets/js/jquery-3.2.1.min.js'%}"></script>

   <!-- Bootstrap Core JS -->
   <script src="{% static 'admin/assets/js/popper.min.js'%}"></script>
   <script src="{% static 'admin/assets/js/bootstrap.min.js'%}"></script>

   <!-- Slimscroll JS -->
   <script src="{% static 'admin/assets/plugins/slimscroll/jquery.slimscroll.min.js'%}"></script>

   <!-- Datatables JS -->
   <script src="{% static 'admin/assets/plugins/datatables/jquery.dataTables.min.js'%}"></script>
   <script src="{% static 'admin/assets/plugins/datatables/datatables.min.js'%}"></script>
```

```
<!-- Custom JS -->
<script  src="{% static 'admin/assets/js/script.js'%}"></script>

</body>

</html>
```

VIEWS.PY

#LOGIN

```python
def login_page(request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']

        user = authenticate(request, username=username, password=password)
        if user is not None:
            auth_login(request, user)
            if user.is_superuser:
                request.session['user_id'] = user.id
                request.session['username'] = user.email
                return redirect('admins_index')
            elif user.is_staff:
                request.session['user_id'] = user.id
                request.session['username'] = user.email
                return redirect('doctors_basedoctor')
            else:
                request.session['user_id'] = user.id
                request.session['username'] = user.email
                return redirect('basepatient')

    return render(request, 'Login.html')
```

#REGISTER

```python
def register(request):

    if request.method == "POST":
        name = request.POST['name']
        username = request.POST['email']
        email = request.POST['email']
        password = request.POST['password']
        confirm_password = request.POST['confirm_password']
        if password == confirm_password:
            if User.objects.filter(username=username).exists():
                messages.info(request, 'Username already exists')
                return redirect('register')
            if User.objects.filter(email=email).exists():
                messages.info(request, "Email already taken")
                return redirect('register')
            else:
                user = User.objects.create_user(username=username,email = email,password=password)
                user_profile=UserProfile(user=user, email=email, name=name)
```

```
                patient_info=PatientInfo(user=user)
                user_profile.save()
                messages.info(request, "Registered Succesfully")
                return redirect('login_page')
        else:
            messages.info(request, " ")
            return redirect('register')
    return render(request, 'register.html')
```

#ADMIN

```
def admin_adddoctor(request):
    if request.method == 'POST':
        # Retrieve data from the POST request
        Name= request.POST.get('Name')
        email= request.POST.get('email')
        password = request.POST.get('password')
        phn= request.POST.get('phn')
        dep_id = request.POST.get('depp')
        role=CustomUser.DOCTOR
        print(role)
        if CustomUser.objects.filter(email=email,role=CustomUser.DOCTOR).exists():
            messages.info(request, 'Email already exists')
            return redirect('admin_adddoctor')
        else:
            user = CustomUser.objects.create_user(email=email, password=password)
            user.role = CustomUser.DOCTOR
            user.save()
            doctor = Docs(user=user,Name=Name,Dep_id_id=dep_id,phn=phn)
            doctor.save()


            subject = 'Doctor Login Details'
            message = f'Registered as an Doctor. Your username: {email}, Password: {password}'
            from_email = settings.EMAIL_HOST_USER
            recipient_list = [user.email]

            send_mail(subject, message, from_email, recipient_list)

            return redirect('admin_doctors')
    else:
        depts = Deps.objects.all()
        context = {  'depts': depts}
        return render(request, 'admin_adddoctor.html', context)
```

#ADD SPECIALIZATION

```
@login_required
def specializations(request):
    if request.method == 'POST':
        action = request.POST.get('action')

        if action == 'add':
            # Add a new specialization
```

```
            specialization_name = request.POST.get('specialization_name')
            formatted_specialization_name = specialization_name.capitalize()
            try:
                Specialization.objects.create(specialization_name=formatted_specialization_name)
                messages.success(request, 'Specialization created successfully.')
            except Exception as e:
                messages.error(request, 'Error creating specialization: {}'.format(str(e)))


        elif action == 'edit':
            try:
                # Edit an existing specialization
                specialization_id = request.POST.get('specialization_id')
                specialization = get_object_or_404(Specialization, pk=specialization_id)
                new_specialization_name = request.POST.get('specialization_name')

                if
Specialization.objects.filter(specialization_name=new_specialization_name).exclude(id=specialization.id).
exists():
                    messages.error(request, 'Specialization with this name already exists.')
                else:
                    specialization.specialization_name = new_specialization_name
                    specialization.save()
                    messages.success(request, 'Specialization updated successfully.')
            except Exception as e:
                messages.error(request, 'An error occurred during specialization update: {}'.format(str(e)))


        elif action == 'delete':
            # Delete an existing specialization
            specialization_id = request.POST.get('specialization_id')
            specialization = get_object_or_404(Specialization, pk=specialization_id)
            specialization.delete()
            messages.success(request, 'Specialization deleted successfully.')

    # Get all specializations for displaying in the template
    specializations = Specialization.objects.filter(status=False)
    return render(request, 'admins/specialities.html', {'specializations': specializations})
```
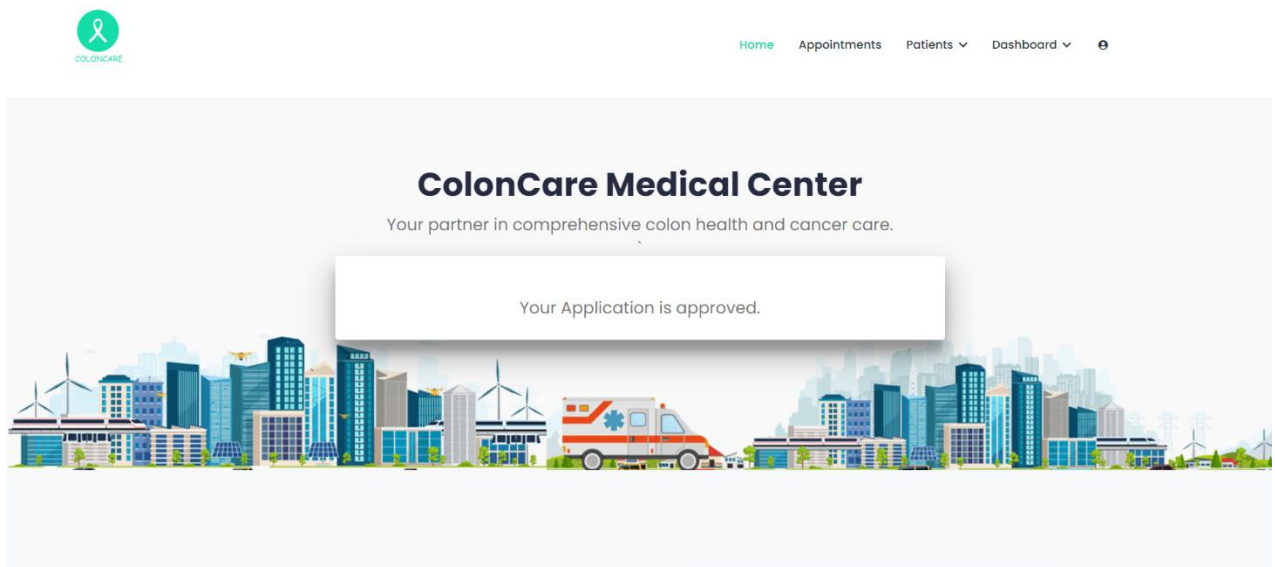
9.1   Screen Shots

HOME PAGE
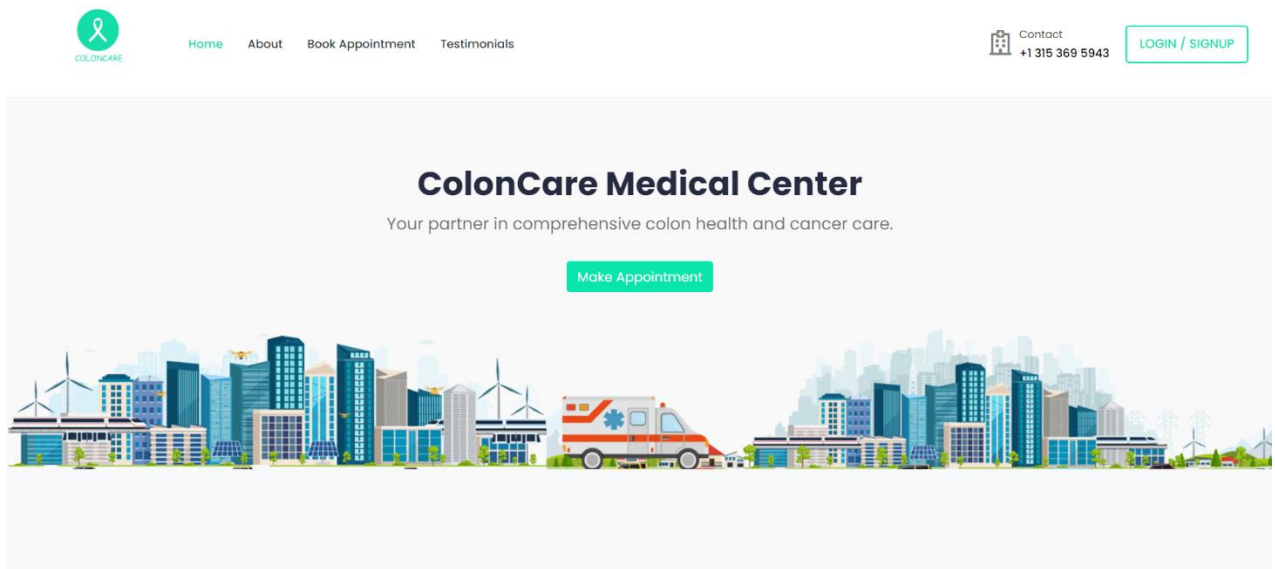
DOCTOR'S HOME PAGE

APPOINTMENTS PAGE

PATIENTS'S HOME PAGE

ADMIN DASHBOARD

REGISTERED USERS

Dashboard / Registered Users

Name	Email	Role	Status	Actions
admin	devika@gmail.com	Admin	Active	Deactivate
alfred@gmail.com	alfred@gmail.com	Doctor	Active	Deactivate
nancy@gmail.com	nancy@gmail.com	Doctor	Active	Deactivate
liza@gmail.com	liza@gmail.com	Doctor	Active	Deactivate
ashna@gmail.com	ashna@gmail.com	Doctor	Active	Deactivate
hari@gmail.com	hari@gmail.com	Doctor	Active	Deactivate