

AllergyCare

Allergy Specialist Application

Project Report

Submitted by

Ashwin S Pillai

Reg. No.: AJC22MCA-2027

In Partial fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS
(MCA TWO YEAR)
[Accredited by NBA]

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. 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, “AllergyCare” is the bona-fide work of **Ashwin S Pillai (Regno:AJC22MCA-2027)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Lisha Varghese

Internal Guide

Ms. Meera Rose Mathew

Coordinator

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

External Examiner

DECLARATION

I hereby declare that the project report “**AllergyCare**” is a bona-fide work of done at Amal Jyothi College of Engineering, towards the partial fulfillment 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:12-04-2024

Ashwin S Pillai

KANJIRAPPALLY

Reg: AJC22MCA-2027

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 Director (Administration) **Rev. Fr. Dr. Roy Abraham Pazhayaparampil** 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 **Ms. Meera Rose Mathew** for her 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 **Ms. Lisha Varghese** for her 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.

Ashwin S Pillai

ABSTRACT

A comprehensive digital platform created to improve the treatment of patients with skin allergies is the Skin Allergy Specialist Application. With components including Doctors, Image Evaluation by Doctor, Medicines, Appointments, and Clinic Locations, it provides an easy to use interface.

It is simple for patients to sign up, log in, locate dermatologists, submit image-based diagnosis requests, obtain recommended medications, make appointments, learn clinic locations, locate the closest clinics, and offer insightful comments.

Logging in, reviewing patient requests, diagnosing accurately from photos, prescribing medications, and scheduling visits are all possible for doctors.

Administrators are essential to the platform's smooth operation since they oversee user accounts and evaluate user feedback.

This application is an example of how medical knowledge and technical innovation may coexist in the age of digital health. It draws attention to how telemedicine and online consultations might enhance medical care and change the dynamic between patients and experts.

Faster diagnosis and treatment are made possible by the application, which benefits patients by improving outcomes and reducing suffering. Additionally, it lessens the burden on healthcare systems by making the most of expert time and empowering patients to actively participate in their own treatment.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	6
2.4	PROPOSED SYSTEM	6
2.5	ADVANTAGES OF PROPOSED SYSTEM	7
3	REQUIREMENT ANALYSIS	8
3.1	FEASIBILITY STUDY	9
3.1.1	ECONOMICAL FEASIBILITY	9
3.1.2	TECHNICAL FEASIBILITY	10
3.1.3	BEHAVIORAL FEASIBILITY	10
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	11
3.2	SYSTEM SPECIFICATION	13
3.2.1	HARDWARE SPECIFICATION	13
3.2.2	SOFTWARE SPECIFICATION	13
3.3	SOFTWARE DESCRIPTION	14
3.3.1	DJANJO	14
3.3.2	SQLITE	14
4	SYSTEM DESIGN	15
4.1	INTRODUCTION	16
4.2	UML DIAGRAM	16
4.2.1	USE CASE DIAGRAM	16
4.2.2	SEQUENCE DIAGRAM	18
4.2.3	STATE CHART DIAGRAM	19
4.2.4	ACTIVITY DIAGRAM	20
4.2.5	CLASS DIAGRAM	21
4.2.6	OBJECT DIAGRAM	23
4.2.7	COMPONENT DIAGRAM	25

4.2.8	DEPLOYMENT DIAGRAM	26
4.2.9	COLLABORATION DIAGRAM	26
4.3	USER INTERFACE DESIGN USING FIGMA	27
4.4	DATABASE DESIGN	30
5	SYSTEM TESTING	40
5.1	INTRODUCTION	41
5.2	TEST PLAN	41
5.2.1	UNIT TESTING	42
5.2.2	INTEGRATION TESTING	42
5.2.3	VALIDATION TESTING	42
5.2.4	USER ACCEPTANCE TESTING	42
5.2.5	AUTOMATION TESTING	43
5.2.6	SELENIUM TESTING	43
6	IMPLEMENTATION	60
6.1	INTRODUCTION	61
6.2	IMPLEMENTATION PROCEDURE	61
6.2.1	USER TRAINING	62
6.2.2	TRAINING ON APPLICATION SOFTWARE	62
6.2.3	SYSTEM MAINTENANCE	62
6.2.4	HOSTING	62
7	CONCLUSION & FUTURE SCOPE	65
7.1	CONCLUSION	66
7.2	FUTURE SCOPE	66
8	BIBLIOGRAPHY	67
9	APPENDIX	69
9.1	SAMPLE CODE	70
9.2	SCREEN SHOTS	85

List of Abbreviation

IDE	Integrated Development Environment.
HTML	Hyper Text Markup Language.
CSS	Cascading Style Sheet
UML	Unified Modeling Language
OpenCV	Open-Source Computer Vision
RDBMS	Relational Database Management System
UML	Unified Modeling Language
OS	Operating System
PHP	Hypertext Preprocessor
GUI	Graphical User Interface
ORM	Object-Relational Mapping
URL	Uniform Resource Locator
MTV	Model-Template View
MVC	Model-View-Controller
SQL	Structured Query Language
SQLite	Structured Query Language Lite
JS	Java Script
AI	Artificial Intelligence
PK	Primary Key
FK	Foreign Key
1NF	First Normal Form
2NF	Second Normal Form
3NF	Third Normal Form

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

By facilitating seamless communication between patients, physicians, and administrators, the Skin Allergy Specialist Application is a cutting-edge and user-focused platform that is completely changing the dermatological care environment. This all-inclusive application improves interaction efficiency and takes skin allergy management to a whole new level with its modules for doctors, image detection, medications, appointments, and clinic locations.

Patients can easily register, log in, browse dermatologists, submit image-based requests, access medicine recommendations, schedule appointments, explore clinic locations, access quick remedies, and leave insightful feedback thanks to user-friendly functionalities that streamline the patient experience. This guarantees a unique and interesting experience while also empowering patients to manage their skin sensitivities.

With the application's customized interface, medical professionals can log in, examine patient requests, diagnose patients accurately from photographs, prescribe treatments, and schedule appointments with ease. This more efficient method speeds up the diagnosis and treatment procedures, improving results and lessening patient suffering. The focus on virtual consultations and telemedicine reinterprets the doctor-patient relationship and emphasizes how digital health solutions might improve overall patient care.

Keeping an eye on user accounts, handling comments, and preserving the integrity of the system are all crucial tasks for administrators. This administrative layer keeps everything running smoothly and securely, which helps to keep the application getting better over time.

1.2 PROJECT SPECIFICATION

A cutting-edge digital platform called the Skin Allergy Specialist Application was created to improve communication between medical professionals, administrators, and patients in the field of skin allergy treatment. With the integration of essential modules including Doctors, Image Detection, Medicines, Appointments, and Clinic Locations, this comprehensive system forms a cohesive ecosystem that meets the diverse needs of its customers.

The application provides a user-centric experience for patients by streamlining the login and registration processes. It makes it simple for patients to navigate dermatologists according to their areas of expertise, assisting them in making wise choices. The ability to make image-based requests provides a useful way to make accurate diagnoses in addition to personalized medication recommendations. Patients may quickly make appointments with their preferred dermatologists thanks to the streamlined process.

The Clinic Locations module provides thorough information to improve the patient experience, and the rapid cures address common skin sensitivities. By encouraging active patient feedback, the application fosters a collaborative and adaptable healthcare environment.

1.2.1 ADMIN MODULE

In order to guarantee that the program runs smoothly, administrators are essential. They are in charge of several things that are essential to the platform's functionality, including: The administration of patient and physician accounts is one of the primary responsibilities given to administrators. This include managing the user registration process and maintaining up-to-date user profiles. Administrators maintain the integrity of the platform and ensure that user access is consistent with the intended use of the application by skillfully managing these components.

1.2.2 PATIENT MODULE

The option to sign up and log in allows patients to access features that are specifically tailored to their needs as a patient. The foundation for a patient-centered experience is laid by this safe and tailored access, which guarantees that the patient may make the most of the extensive array of healthcare resources offered by the application. Patients can look through a carefully curated list of highly qualified physicians. This entails looking over physician profiles and selecting medical specialists who are most qualified to deal with their particular skin allergy problems. The patient's autonomy in controlling their health is increased when they are able to make well-informed decisions regarding healthcare providers.

1.2.3 DOCTOR MODULE

The doctor module is a crucial component of the healthcare industry that is especially meant for dermatologists who treat patients with skin allergies. This module serves as a thorough resource, including a variety of features meant to streamline and improve the standard of care provided by medical professionals. The doctor module's ability to securely register and log in medical professionals is one of its primary features. This first step guarantees that access to the platform is restricted to persons who possess the requisite medical knowledge. This preserves the integrity of the healthcare system and emphasizes how important patient information security and confidentiality are.

Technologies Used:

- *Frontend:* HTML, CSS, JavaScript, AJAX.
- *Backend:* Django.
- *Database:* SQLite.
- *Payment Gateway Integration:* Razorpay.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

For AllergyCare, system analysis include gathering information, evaluating the system's functionality, and making recommendations for improvements. Analysts determine inputs, outputs, and process relationships to organizational results using surveys and interviews. The objective is to identify problems, comprehend operations, and suggest fixes. Designers refine recommendations until users are happy by comparing suggested solutions with the current system. The success of the project depends on the preliminary investigation, which is the first stage of data collection and analysis.

2.2 EXISTING SYSTEM

The purpose of AllergyCare programs is to match users with licensed allergy specialists for consultations. These encounters can be arranged in person or virtually via chat and video. This strategy highlights how important it is to offer the best allergy care services possible in order to improve health and increase accessibility to excellent care. The intention is to enhance allergy management support by providing simple access to professional guidance and assistance.

2.2.1 NATURAL SYSTEM STUDIED

Currently, AllergyCare's appointment booking procedure primarily uses manual communication channels including phone calls and emails. In order to schedule appropriate appointment times, clients must make the first move in contacting providers or specialists. This might lead to drawn-out conversations. The inquiry into AllergyCare's operational methods is primarily focused on appointment scheduling because there is a need for a more technologically updated approach to improve efficiency..

2.2.2 DESIGNED SYSTEM STUDIED

To get around the drawbacks of the present manual appointment administration procedures, we suggest upgrading the AllergyCare Appointment administration System. This technology seeks to transform the way allergy doctors and patients communicate by providing a simplified, easy-to-use, and effective solution. The digital platform, which enables patients to book appointments, view the profiles of allergy specialists, and get real-time notifications, is a crucial component of the system. Convenience, openness, and the overall quality of the service experience are prioritized by doing away with the necessity for time-consuming phone calls and emails. Furthermore, the system leverages the advantages of digital technology in the allergy care industry while addressing the limitations of the current framework.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Inefficiency** : In order to acquire a suitable appointment time, patients and allergy experts must frequently communicate back and forth in multiple rounds during the manual process of booking appointments for allergy care, which can be quite inefficient.
- **Limited Access** : The implementation of this approach could potentially restrict access to allergy care services to those who possess advanced navigational skills, thereby placing less tech-savvy folks at a disadvantage when seeking allergy care.
- **Lack of Transparency** : It may be difficult for patients to make educated decisions about their allergy care if they lack access to thorough information about allergy experts.
- **Missed Appointments** : Patients and allergy experts may become frustrated if visits are missed due to a lack of automated reminders and notifications within the allergy care application.
- **Limited Availability** : Due to the system's reliance on manual procedures, patients with hectic schedules or those who need allergy care after regular business hours may find it difficult to get appointments.

2.4 PROPOSED SYSTEM

To get around the drawbacks of the present manual appointment administration procedures, we suggest upgrading the AllergyCare Appointment administration System. This technology seeks to transform the way allergy doctors and patients communicate by providing a simplified, easy-to-use, and effective solution. The digital platform, which enables patients to book appointments, view the profiles of allergy specialists, and get real-time notifications, is a crucial component of the system. Convenience, openness, and the overall quality of the service experience are prioritized by doing away with the necessity for time-consuming phone calls and emails. Furthermore, the system leverages the advantages of digital technology in the allergy care industry while addressing the limitations of the current framework.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- **Convenience:** Through the AllergyCare app, patients may receive allergy treatment services from the comfort of their homes, doing away with the necessity for in-person trips to medical facilities.
- **Transparency:** In order to help them make educated decisions regarding their allergy care, patients have access to comprehensive biographies, reviews, and information about the expertise of allergy specialists.
- **Secure Payments:** The AllergyCare application's payment integration enhances overall security and confidence by ensuring safe financial transactions for allergy treatment services.
- **Empowering Users:** By giving interns and students studying allergy management chances, the system closes the gap between allergy education and practice.
- **Enhanced Scheduling:** The AllergyCare application's integration of allergy specialist calendars enables patients to efficiently arrange appointments by matching their schedules with the specialists' availability.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Any software development project's viability and practicality must be assessed in terms of its feasibility. A thorough feasibility assessment has been carried out in the instance of the Skin Allergy Specialist Application to ascertain whether the suggested solution is feasible in the context of software development and in line with organizational goals.

A cutting-edge and crucial medical tool designed to satisfy the particular needs of people with skin allergies is the Skin Allergy Specialist Application. As a common health issue, skin allergies require prompt, accurate diagnosis and treatment. By creating an integrated digital platform that streamlines every step of the process from patient consultation to treatment this application seeks to close the current gap.

This application's main goal is to offer a comprehensive and user-friendly digital environment that makes effective skin allergy care possible. Important features include the ability to register patients, schedule appointments, do image-based diagnosis, issue prescriptions, gather feedback, and purchase medications.

Meeting Organizational Requirements: Evaluating if the software can effectively close the current gap in healthcare services by meeting the special needs of people with skin allergies.

Technical Feasibility: Assessing the talent, resources, and technology needed for the project's development and deciding whether it can be implemented successfully from a technological standpoint.

Operational Feasibility: Evaluating the software's viability in terms of user acceptance, operational efficiency, and human resources in particular, and envisioning its flawless operation after development.

Economic Feasibility: Carrying out a comprehensive economic study to assess the expenses related to software development as well as the anticipated long-term gains and advantages that the system can offer the company.

3.1.1 ECONOMIC FEASIBILITY

The costs of software development and the anticipated long-term profits are thoroughly examined in the economic feasibility evaluation. Crucial elements consist of:

Resource Costs: Assessing the hardware, software licenses, requirements analysis, elicitation, and development time and resource requirements for software development.

Return on Investment (ROI): Evaluating the potential gains and benefits that the Skin Allergy Specialist Application should have for the company, taking into account things like increased

customer happiness, increased competitiveness, and lower administrative expenses.

Cost Reduction: Realizing that the system can be developed profitably and that open-source software resources are easily accessible will lower development expenses.

Finally, the feasibility analysis confirms the applicability and viability of the Skin Allergy Specialist Application. To guarantee its successful development, user acceptance, and long-term advantages for people with skin allergies and healthcare organizations alike, it has undergone extensive technical, operational, and economic examinations.

3.1.2 TECHNICAL FEASIBILITY

One crucial factor to take into account is the Skin Allergy Specialist Application's technological viability. It entails a thorough assessment of the technical elements necessary for effective deployment. Important elements consist of:

Resource Assessment: Assessing the availability of the hardware and software resources required for the application's development and implementation.

Technology Evaluation: Evaluating the members of the software development team's technical skills and talents to make sure they have the knowledge required for developing mobile applications, image processing, and websites.

Stability and Established Technology: Verifying the stability and proven nature of the selected technology stack to ensure security and dependability throughout the development process.

User Community: Ensuring that the technologies chosen have a sizable user base that makes it possible to access knowledge and assistance when problems arise or changes are needed.

3.1.3 BEHAVIORAL FEASIBILITY

The following inquiries are part of the suggested system:

- Is user support sufficient?
- Will the suggested system do harm to anyone?

The project would be beneficial as, once designed and implemented, it would achieve the goals. The project is deemed behaviorally feasible after a thorough evaluation of all behavioral factors. Because AllergyCare GUI is user-friendly, users can use it without any training.

FEASIBILITY STUDY QUESTIONNAIRE

1.Project Overview?

A web-based tool called the AllergyCare Application aims to transform the dermatological and allergy treatment industries. It provides a wide range of services, such as managing prescriptions, making appointments, learning about allergies, and making treatment recommendations. Using cutting-edge technologies, the main goal of this application is to improve patient care and streamline the operations of dermatological and allergy clinics. In order to promote health and allergy management, the AllergyCare is intended for use by both individuals seeking specialized care and healthcare providers specializing in dermatology and allergy therapies. The redesigned program enhances patient care, simplifies clinic procedures, and makes it easier for patients, physicians, and administrators to communicate effectively. The redesigned platform revolutionizes the field of dermatology and allergy care by integrating state-of-the-art technology including intelligent scheduling, an ML-powered quiz, online consultations, and allergy medications.

2.To what extend the system is proposed for?

By streamlining dermatological and allergy clinic administration, the system seeks to enhance patient satisfaction and treatment results. With features including appointment scheduling, allergy information access, treatment planning, and prescription management, it serves a range of stakeholders, including medical professionals, patients, and healthcare providers.

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

- Patients seeking dermatology or allergy care.
- Dermatologists, allergists, and medical staff.
- Administrators managing clinic operations.

4.List the Modules included in your System?

The Skin Allergy Specialist Application comprises the following modules:

- Doctors Module
- Image Detection Module
- Medicine Module
- Appointment Module
- Clinic Module
- ML symptom identifier Module
- Diseases Module
- Payment Module
- Patient Module
- Online quiz

- Real Time Chatting

5. Identify the users in your project?

- Patients: Individuals seeking dermatology or allergy care services.
- Doctors: Dermatologists and allergists providing medical expertise.
- Administrators: Clinic managers and staff responsible for daily operations.
- Knowledge Purpose : To know about any diseases or allergies

6. Who owns the system?

Clinic managers or healthcare institutions own and manage the AllergyCare Application.

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

The Skin Allergy Specialist Application, which focuses on dermatology and allergy treatment, is strongly associated with the healthcare sector. Its goal is to improve patient outcomes and services in dermatological and allergy clinics.

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

- Dr Ramendran (Dermatologist)
- Aster Hospital (Allergy Clinic)

9.Questionnaire to collect details about the project?

- a) **How should the application's user interface be designed to allow patients to access their treatment plans, medical data, and allergy information?**

Our objective is for an interface that is easy to use, safe, and navigable. The user or patient can quickly grasp how various processes operate to ensure their smooth operation.

- b) **How can the system guarantee that patient data and medical records are handled securely?**

The implementation of robust security measures can aid in mitigating threats and data leaks.

- c) **Which general dermatological and allergy treatment services does your clinic provide?**

In response, we offer a range of services including medical care, allergy testing, diagnosis evaluations, and treatment plans.

- d) **What do you think about using telemedicine to provide treatments related to allergies and dermatology?**

In particular, telemedicine could make follow-up consultations and prescription refills more accessible for patients.

- e) **Which features or resources, in your opinion, would improve patients' overall experience receiving dermatological and allergy care?**

It would be beneficial to have features like appointment reminders, educational materials, and telemedicine consultations.

f) What difficulties do you foresee while switching from manual to an online system?

To guarantee a seamless shift for both personnel and patients, it could be necessary to provide training and assistance.

g) In order to continuously enhance the application and service quality, how do you intend to integrate patient feedback analysis and testimonials?

In order to find areas for improvement in the application and service quality, we want to apply natural language processing (NLP) algorithms to analyze patient testimonials and comments.

h) What other features or resources would you recommend to improve dermatological and allergy patients', physicians', and administrators' overall experience receiving care?

Personalized treatment regimens based on medical history and patient preferences, along with real-time patient progress tracking for physicians and administrators, would significantly improve the entire experience of dermatology and allergy care.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	Intel core i5
RAM	8GB
Hard disk	512 SSD

3.2.2 Software Specification

Front End	HTML5, CSS, Bootstrap
Back End	Django, SQLite
Database	SQLite
Client on PC	Windows 7 and above.
Technologies used	Django, HTML5, AJAX, Bootstrap, JS.

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

With its efficiency and variety, Django, a popular open-source web framework, represents the height of contemporary online development. Django, which is based on Python, prioritizes flexibility and simplicity in its Model-View-Controller architecture. Its user-friendly template engine, efficient URL routing, and Object-Relational Mapping system are noteworthy characteristics. While strong security measures and middleware support improve application integrity, the integrated admin interface makes data management simpler. Django easily expands to meet the needs of increasing datasets and traffic volumes. Its applicability to API development is further expanded by its REST Framework expansion. With the help of a vibrant community and a wealth of documentation, Django is a potent toolset that easily shapes the creation of safe, dynamic websites for a variety of uses, including content management systems and Restful API. Django provides developers with an all-inclusive and flexible framework that allows them to create complex and marketable web applications.

3.3.2 SQLite

The simplicity, portability, and performance of SQLite, a lightweight relational database management system, are highly praised. SQLite functions as a standalone, single-file database that requires little setup and does not require a separate server process. Due to its design, it is the best option for small- to medium-sized projects, mobile applications, and embedded systems.

SQLite simplifies database management with zero configuration and a server-less design. Because it supports ACID transactions, data stability and integrity are guaranteed even in the event of system outages. Different data types can be stored in the same column with flexibility thanks to the dynamic typing capability.

It is noteworthy because SQLite is cross-platform compatible, which allows it to be used in a variety of contexts and operating systems.

Its widespread use for local data storage can be seen in a variety of applications, from web browsers to mobile apps. When it comes to lightweight, self-contained, and quickly deployable database systems, SQLite is the clear choice since it balances simplicity with functionality and dependability.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The design phase is where the development process of any designed system or product starts. A well-executed design, which requires creativity, is the foundation of an effective system. For a system or procedure to be implemented, it must be fully specified using a range of methods and ideas. The design stage is crucial in software engineering, regardless of the development paradigm that is employed. It serves as the technical foundation of the process and aims to generate the architectural detail required to develop a system or product.

In order to optimize all aspects of performance, accuracy, and efficiency, this program went through a thorough design phase. During the design process, a document intended for users gets transformed into one intended for programmers or database administrators.

4.2 UML DIAGRAM

Software engineering models, develops, and documents software systems using the standardized visual language Unified Modeling Language (UML). UML diagrams give developers, stakeholders, and designers a common communication tool for representing many aspects of a software system in a clear and structured manner. UML diagrams come in a variety of forms, such as use case, class, and sequence diagrams, among others. Each is intended to convey a specific bit of information regarding the layout, functionality, and relationships inside the system. Because they aid in the visualization, analysis, and design of complex systems, UML diagrams are crucial to the software development process because they increase its effectiveness and efficiency.

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

4.2.1 USE CASE DIAGRAM

Use Case Software engineering's main tool, diagrams provide a visual depiction of how a system interacts with its external environment. Essentially, they give an organized way to recognize and characterize the numerous features a system has to offer and the ways in which different actors or entities can utilize those features. The particular use cases that actors interact with are illustrated

beside them, whether they are acting as users, systems, or outside entities. The nature of these interactions is explained by associations between actors and use cases, which also define the roles and responsibilities of each component of the system.

- **Actor Definition:** Give each actor in the system a precise definition and name. The external entities that interact with the system are portrayed by actors.
- **Use Case Naming:** To effectively communicate the functionality that use cases represent, give them names that are descriptive.
- **Association Lines:** To depict associations between actors and use cases, use solid lines. This represents the communication between the entities.
- **System Boundary:** To show the borders and extent of the system, draw a box around it. This delineates the internal and external components of the system.
- **Include and Extend connections:** To depict shared functionalities across several use cases, utilize "include" connections. To demonstrate alternative or additional functionality, use "extend" relationships.

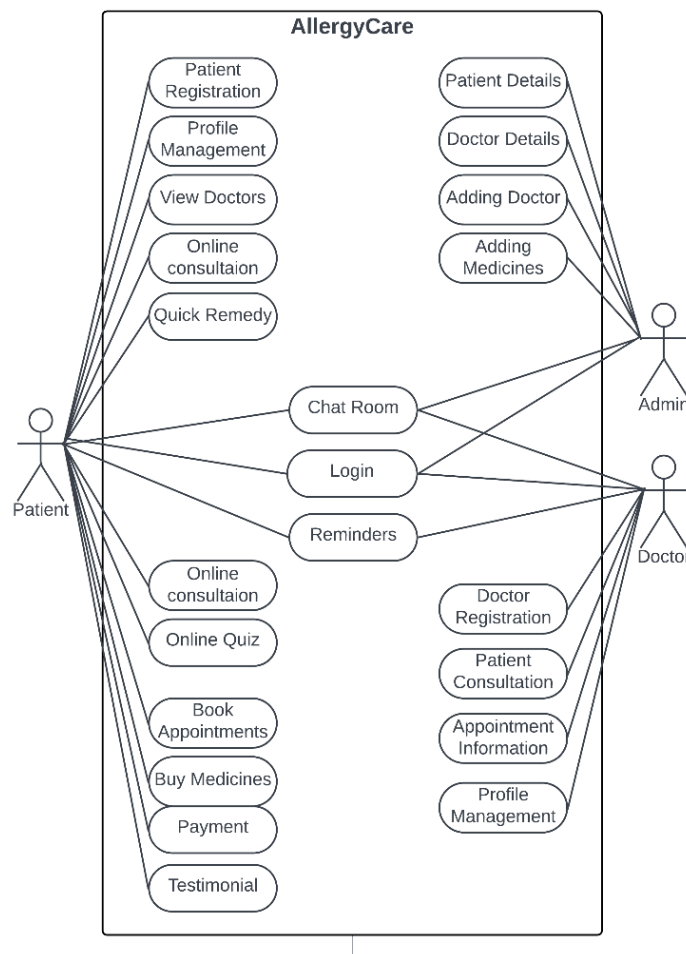


Figure 1: Use Case Diagram

4.2.2 SEQUENCE DIAGRAM

In software engineering, a sequence diagram, a type of Unified Modeling Language (UML) diagram, is used to show the exchanges and interactions that occur across time between different system components or objects. It provides a real-time perspective of how various items interact to do a task or deal with a particular circumstance.

In a sequence diagram, objects are represented as lifelines, and vertical lines illustrate their existence across time. Lifelines are represented by lines and arrows that indicate the flow of calls or messages between objects. These diagrams are highly useful in helping you determine the order of operations and the functions that each item performs inside a given use case. They also show how various objects interact with one another. Sequence diagrams provide a clear and complete illustration of how various items or components work together to accomplish a specific activity, which is crucial for comprehending system behavior and ensuring that software operates as intended.

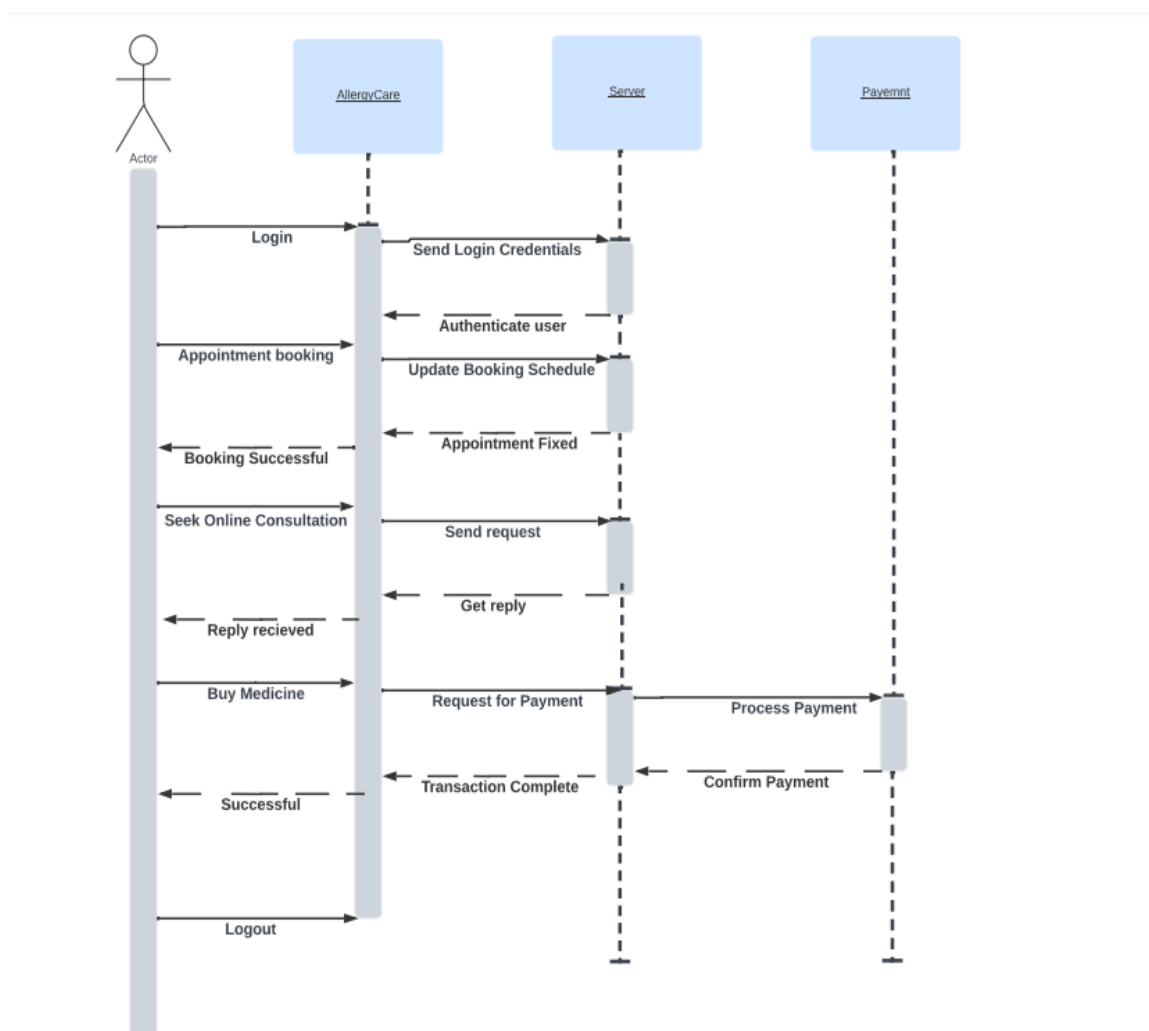


Figure 2: Sequence diagram

4.2.3 STATE CHART DIAGRAM

A State Chart Diagram, often created using the Unified Modeling Language (UML), visually represents the different states an object can undergo and how it transitions between them. Also known as a state machine diagram, it illustrates the behavioral aspects of a system or object overtime. The diagram comprises several elements, including the Initial State denoting the starting point, States representing the system's or object's current condition, Transitions depicting movements between states, Events triggering transitions, and Actions defining transition behaviors. Additionally, Signals, which are messages triggered by events, initiate state transitions. The diagram concludes with a Final State, indicating the completion of the system's or object's behavior.

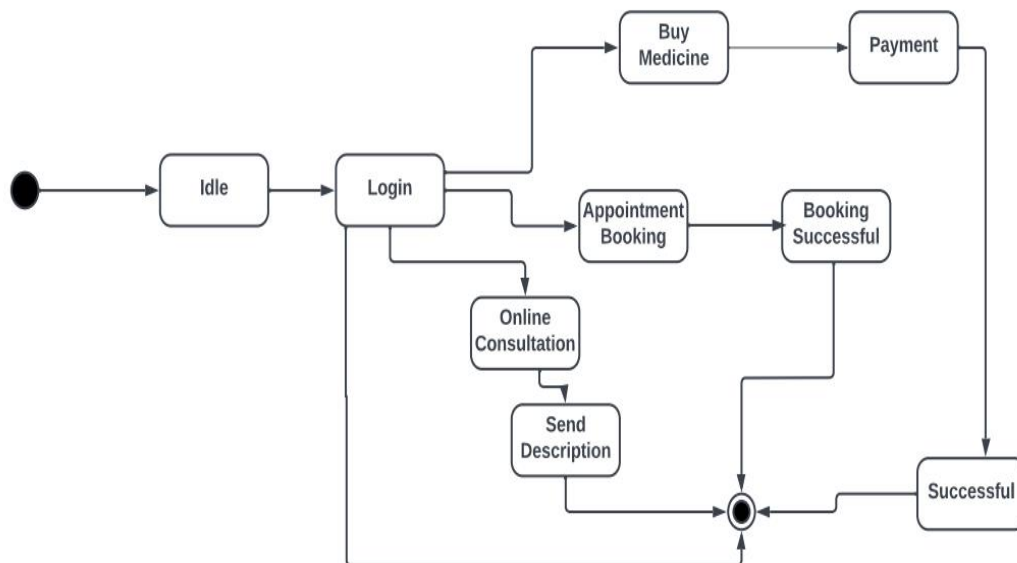


Figure 3: State Chart diagram

4.2.4 ACTIVITY DIAGRAM

An activity diagram is a visual representation of a workflow, illustrating how one activity leads to another in a system's operation. It portrays system behavior by depicting the flow of control from the start point to the end point, including different decision paths encountered during execution. Key components include the Initial node marking the starting point, Activities representing tasks or actions, Control flow arrows indicating the sequence of actions, Decision nodes for branching, Merge nodes to consolidate branches, Fork nodes for parallel flows, Join nodes to reunite parallel, and Final nodes marking the endpoint. Activity diagrams are valuable for clarifying complex processes, identifying potential issues, and effectively communicating process flows to stakeholders and project team members.



Figure 4: Activity diagram

4.2.5 CLASS DIAGRAM

A key component of UML, a class diagram shows classes, their properties, methods, and relationships to visually illustrate a system's structure. Rectangles represent classes, which in a system hold both data and behavior. Relationships between classes are indicated by associations, which highlight their interactions. The cardinality of associations is specified via multiplicity notations. An arrow pointing to the subclass that is deriving from a super-class indicates inheritance. The links between classes' complete parts are shown through aggregation and composition. Interfaces, represented by a circle, lay forth the behavioral contract that a class has to follow. Dependencies draw attention to how dependent one class is on another. Additional information on associations is made possible by association classes. Packages help organize the system by putting related classes together. Class diagrams are essential for system design since they help with software architecture planning and conceptualization. They act as a guide for the development process, guaranteeing an organized and transparent method for creating reliable software systems. Important symbols for class diagrams:

- **Class:** Shown as a rectangle, it includes the methods, attributes, and class name.
- **Attributes:** These represent the qualities or traits of the class and are shown as a list inside the class.
- **Methods:** They specify the class's behaviors and operations and are also listed within the class.
- **Associations:** Lines that represent links and relationships between classes.
- **Multiplicity Notation:** The number of instances in which one class is related to another is indicated by the multiplicity notation.
- **Inheritance:** It is indicated by an arrow, signifying that properties and behaviors are inherited by one class from another.

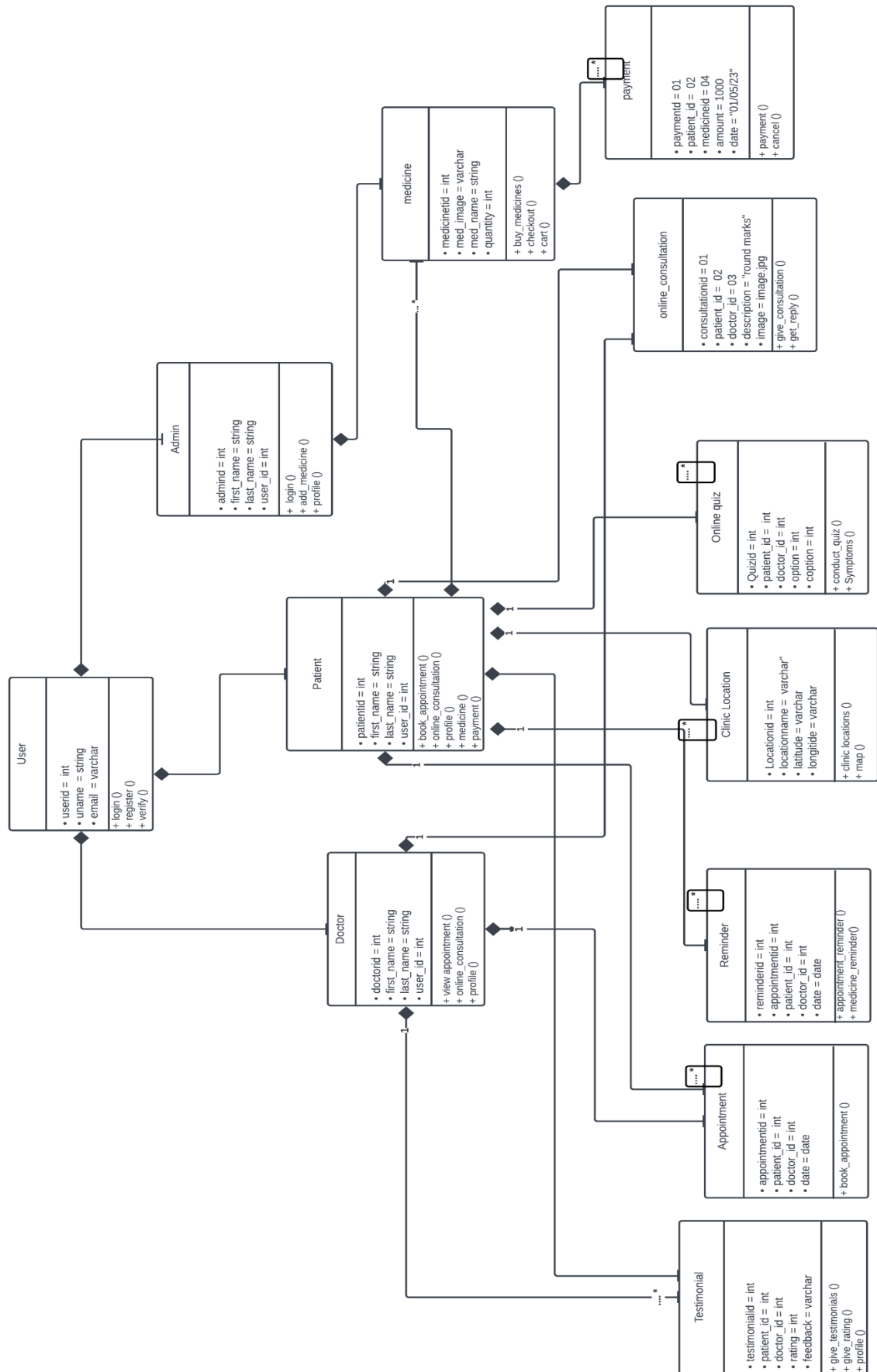


Figure 5: Class diagram

4.2.6 OBJECT DIAGRAM

A UML Object Diagram shows the instances of classes and their relationships, giving a snapshot of the system at a particular point in time. Rectangles are objects that display the condition and actions of individual instances. Links between items show relationships between them and emphasize their interactions. The number of instances engaged in associations is indicated by multiplicity notations. Attributes and their associated values show the status of the object. Object Diagrams provide a thorough understanding of runtime interactions, which helps with testing and system comprehension. They give a concrete depiction of class ties by concentrating on real-world examples. Object Diagrams are comparable to Class Diagrams, except they place more emphasis on actual instances than class definitions.

They are useful instruments for confirming that classes and associations function as anticipated in practice and validating system design. System validation relies heavily on object diagrams to make sure that the parts of the system and how they interact match the requirements and intended design. Important symbols for diagrams of objects:

- **Object:** Shown as a rectangle, it has the name of the object and its attributes, along with their values.
- **Links:** Lines that join items together to show connections or relationships.
- **Multiplicity Notation:** Shows how many occurrences are included in an association.
- **Attributes with Values:** These are displayed inside the item and show its condition at a particular moment in time.
- **Role Names:** Labels assigned to associations that give further details about the type of relationship.
- **Object Name:** Identifies the particular instance by name.
- **Dependency Arrow:** Shows the relationship of dependence between two objects.
- **Composition Diamond:** Depicts a more robust type of possession, in which an item encloses another.

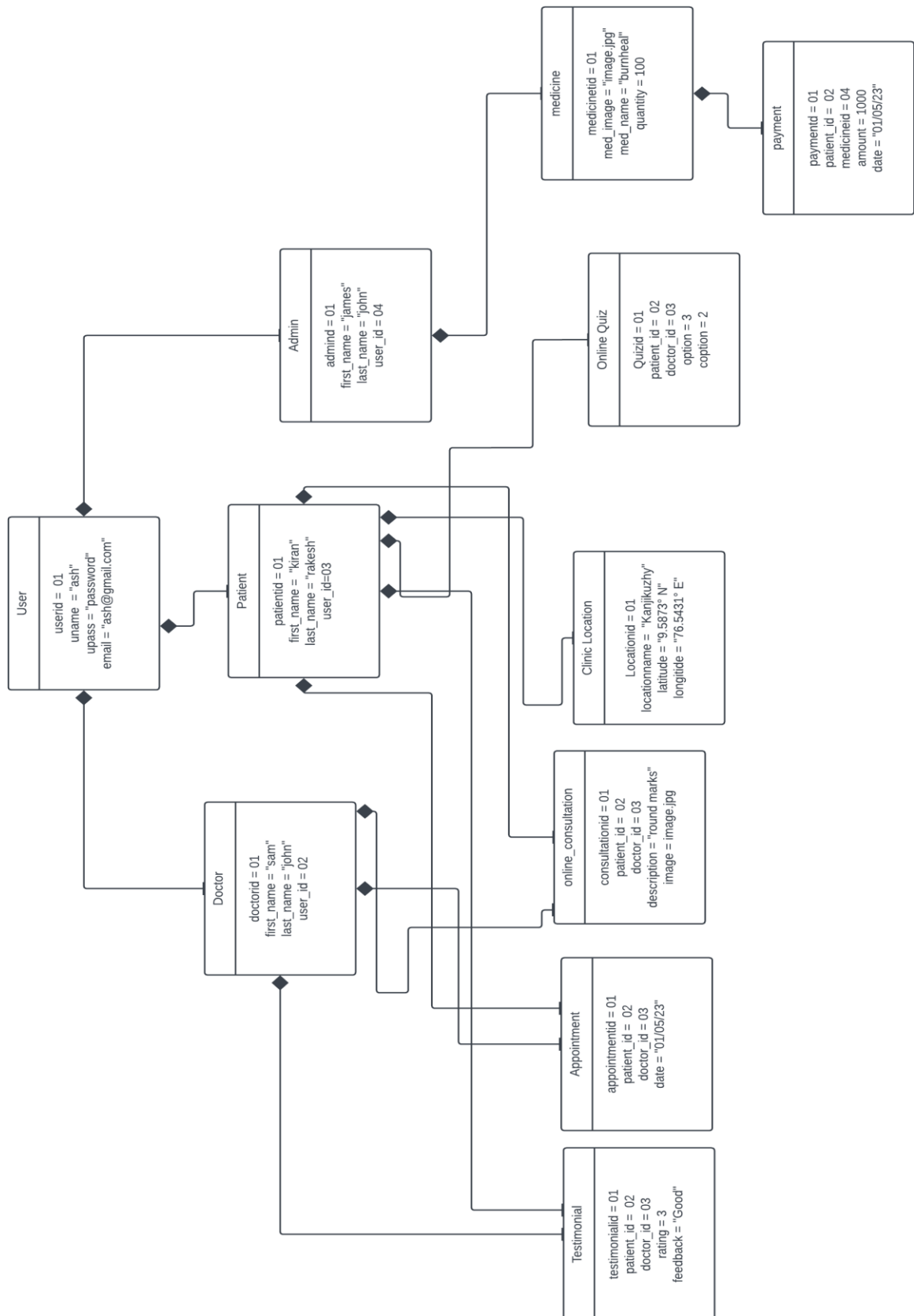


Figure 6: Object diagram

4.2.7 COMPONENT DIAGRAM

A key component of UML is the Component Diagram, which provides a visual depiction of the architecture of a system by illustrating the high-level components and their relationships. Rectangle-shaped objects called components can represent classes, modules, or even whole systems. Component dependencies are shown as arrows, indicating how dependent one component is on the others. Interfaces are little circles that show what services a component needs or provides. Interfaces are connected by connectors to indicate necessary or offered services. Ports are tiny squares that are used to represent the points of connection between an interface and a component. Stereotypes provide further details about a component's function or role. When designing a system, component diagrams play a crucial role in organizing and visualizing the system architecture.

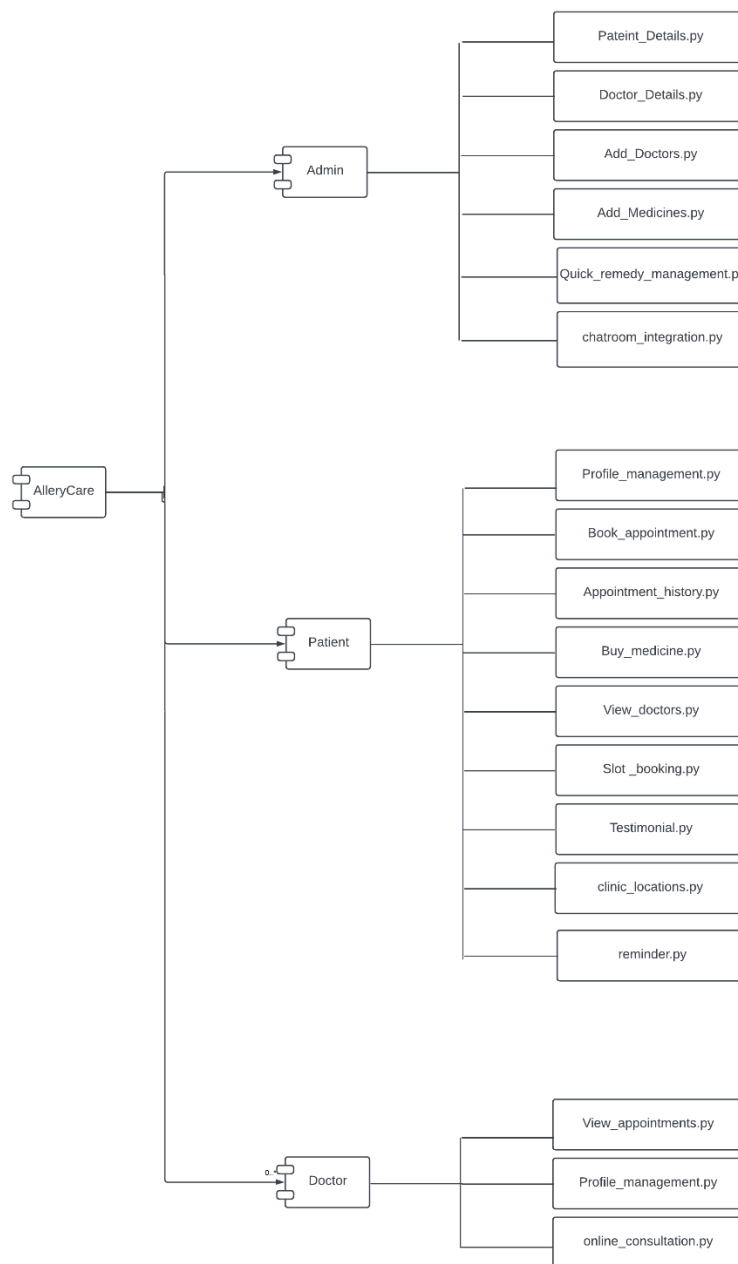


Figure 7: Component diagram

4.2.8 DEPLOYMENT DIAGRAM

The deployment diagram visually represents the placement of software components within the physical computer or server. It provides insights into the static arrangement of the system's elements, including nodes and their interconnections. The diagram outlines the implementation of software on the computer system, detailing the architecture's composition and organization.

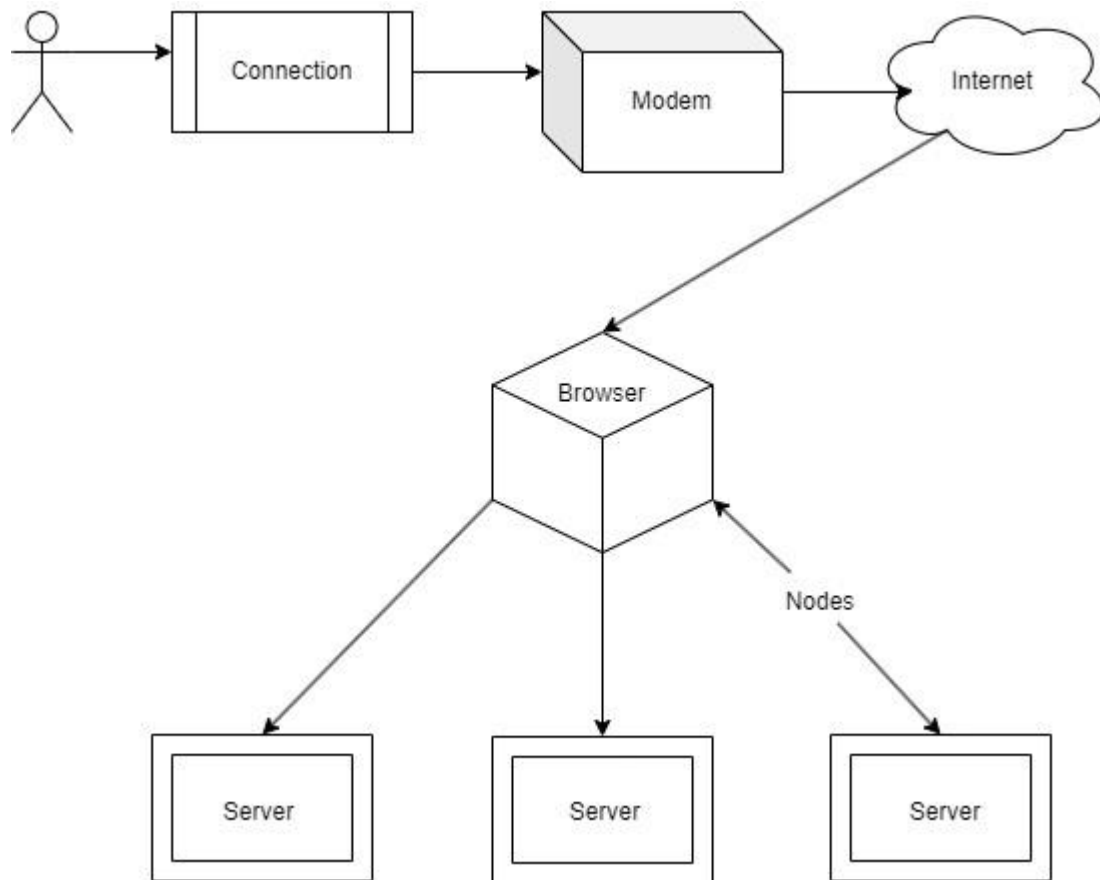
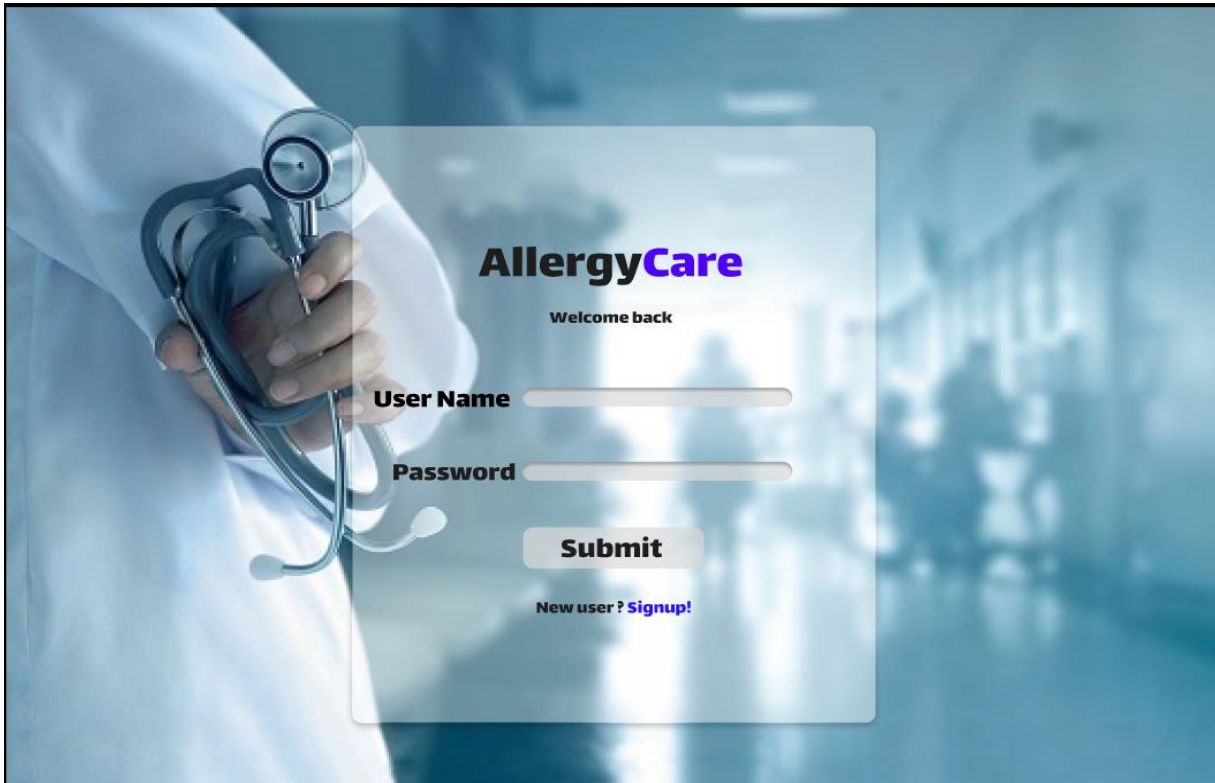


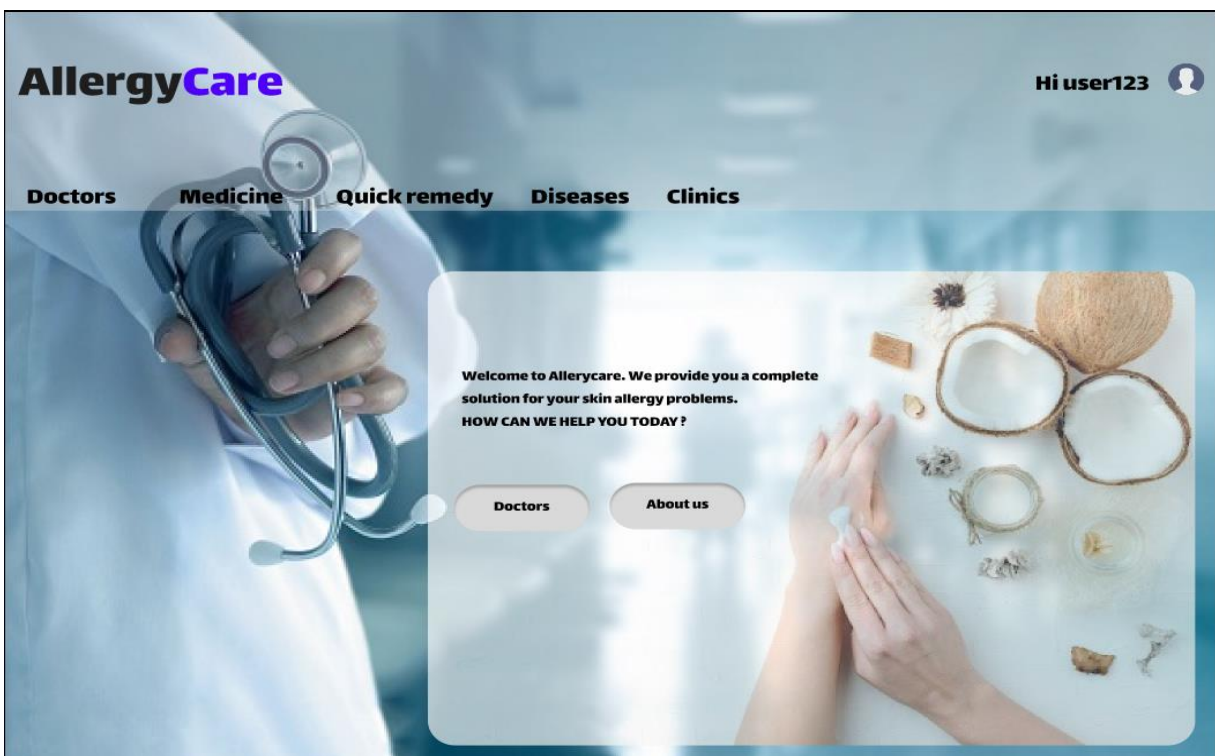
Figure 8: Deployment diagram

4.3 USER INTERFACE DESIGN USING FIGMA

Login page



Landing page



Doctor Page :

AllergyCare Hi user123

Doctors Medicine Quick remedy Diseases Clinics

Dr. Shekar Kumar
Dermatology
MBBS, MD Dermatology
Apollo Hospitals Health City
Online Consult Appointment

Dr. Rajesh Naveen
Oncology
MBBS, MD, DNB
Artis Private Hospital
Online Consult Appointment

Dr. Mukesh Thomas
Dermatology
MBBS, MD Dermatology
Kims Hospital
Online Consult Appointment

Clinics :

AllergyCare Hi user123

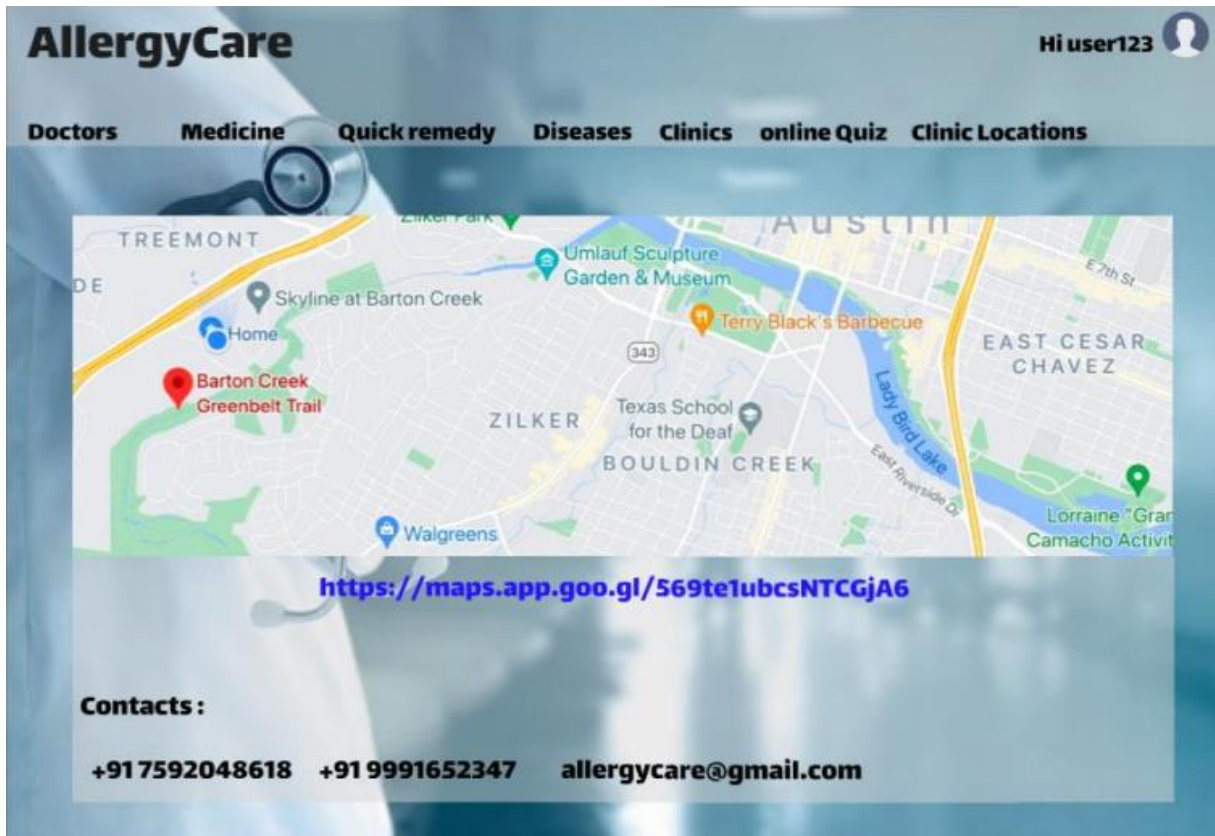
Doctors Medicine Quick remedy Diseases Clinics online Quiz Clinic Locations

CLINICS

Medihub Clinics
★★★★★
Doctor Location Contact

Omega Clinics
★★★★★
Doctor Location Contact

Genesis Clinics
★★★★★
Doctor Location Contact

Clinic Locations :The image shows a web page for AllergyCare. At the top, the logo "AllergyCare" is on the left, and "Hi user123" with a profile icon is on the right. Below the logo is a navigation bar with links: "Doctors", "Medicine", "Quick remedy", "Diseases", "Clinics", "online Quiz", and "Clinic Locations". The main content area features a map of Austin, Texas, with various landmarks labeled, including "Barton Creek Greenbelt Trail", "Walgreens", "Terry Black's Barbecue", and "Lorraine Gran Camacho Activit". Below the map is a Google Maps link: <https://maps.app.goo.gl/569te1ubcsNTCGjA6>. At the bottom, under the heading "Contacts :", there are three contact details: "+91 7592048618", "+91 9991652347", and "allergycare@gmail.com".

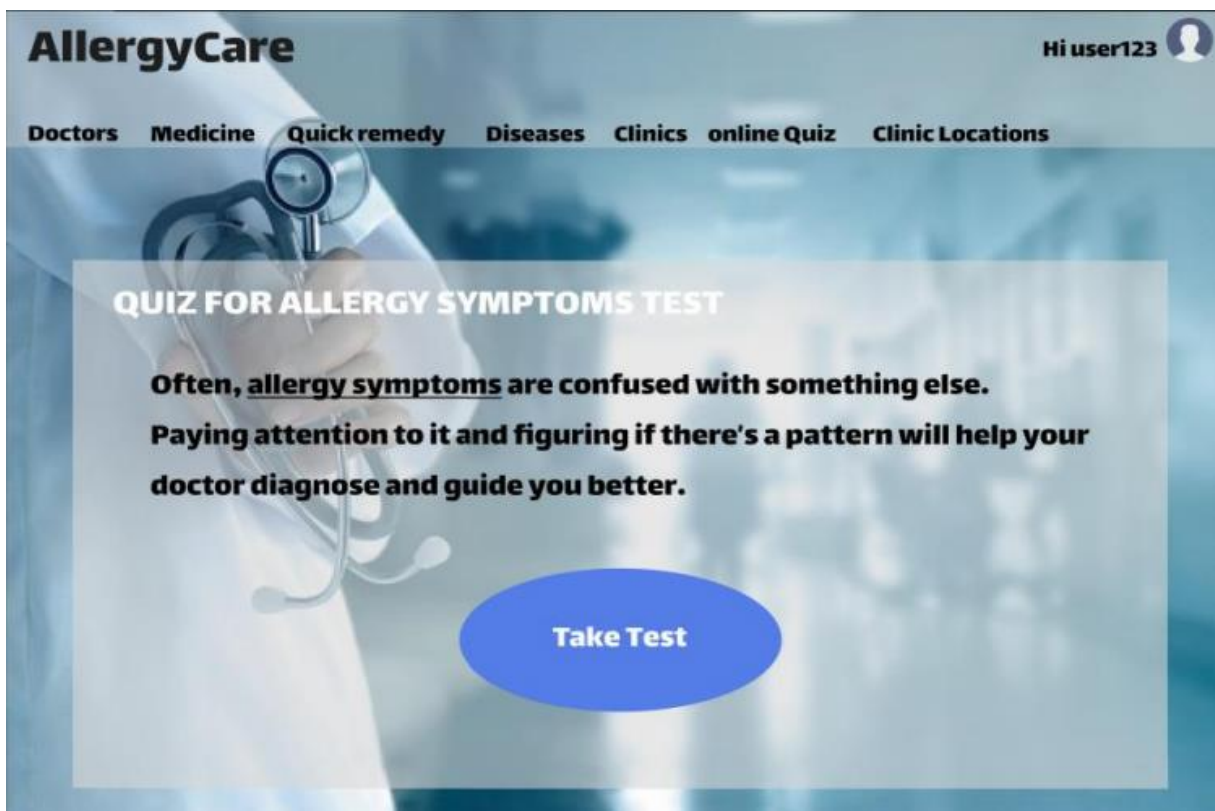
AllergyCare Hi user123

Doctors Medicine Quick remedy Diseases Clinics online Quiz Clinic Locations

<https://maps.app.goo.gl/569te1ubcsNTCGjA6>

Contacts :

+91 7592048618 +91 9991652347 allergycare@gmail.com

Online Quiz :The image shows a web page for AllergyCare. At the top, the logo "AllergyCare" is on the left, and "Hi user123" with a profile icon is on the right. Below the logo is a navigation bar with links: "Doctors", "Medicine", "Quick remedy", "Diseases", "Clinics", "online Quiz", and "Clinic Locations". The main content area features a large blue oval button with the text "Take Test". Above the button, the text reads: "QUIZ FOR ALLERGY SYMPTOMS TEST", "Often, allergy symptoms are confused with something else.", "Paying attention to it and figuring if there's a pattern will help your doctor diagnose and guide you better.".

AllergyCare Hi user123

Doctors Medicine Quick remedy Diseases Clinics online Quiz Clinic Locations

QUIZ FOR ALLERGY SYMPTOMS TEST

Often, allergy symptoms are confused with something else.
Paying attention to it and figuring if there's a pattern will help your doctor diagnose and guide you better.

Take Test

4.4 DATABASE DESIGN

An information collection that has been arranged to facilitate administration, quick access, and overhauls is called a database. Information security may be one of the main goals of any database. There are two steps in the database design process. To build a database that as clearly as possible satisfies user requirements, user requirements are obtained in the first stage. Information-level design is this process, and it's done without reference to any database management system. The design is transformed from an information-level design to a particular database management system (DBMS) design in the second stage. The features of the particular DBMS are taken into consideration at this step, which is referred to as physical-level design.

4.4.1 Relational Database Management System (RDBMS)

A common kind of database that arranges data into tables to enable relationships with other stored data sets is the relational database management system (RDBMS). Large volumes of data, hundreds or millions of rows, each of which is referred to as a record, can be found in tables. A table is a relation, a column heading is an attribute, and a row is referred to as a tuple in the formal language of relational models. There are several tables in a relational database, each with its own name. In a table, each row denotes a group of associated values.

Relationships between tables are pre-established in relational databases to guarantee the integrity of referential and entity relationships. A domain D is a collection of atomic values. Typically, a domain is defined by selecting a data type that serves as the basis for all of the domain's data values. Giving the domain a name will help to clarify the meaning of the values it holds. A relation's values are all atomic and cannot be split further.

The two most crucial types of keys used to create table associations in relational databases are the main key and the foreign key. These keys can be used to establish linkages between entity integrity and referential integrity. Referential integrity guarantees that every unique foreign key value must have a corresponding primary key value in the same domain, whereas entity integrity guarantees that no primary key may have null values. Other kinds of keys exist as well, including candidate and super keys.

4.4.2 Normalization

Normalization is a database design process that eliminates annoying features like Insertion, Update, and Deletion Anomalies and reduces overt repetition in the information. Larger tables are divided into smaller tables by normalization rules, which then connect them using connections. The goal of SQL standardization is to eliminate unnecessary (duplicate) data and ensure that data is stored correctly. Edgar Codd, the creator of the relational model, introduced the First Normal Form, which introduced the notion of data normalization. He then went on to expand the theory with the Second and Third Normal Forms. Afterwards, he collaborated with Raymond F. Boyce to advance the Boyce-Codd Normal Form theory.

First Normal Form(1NF):

If a table meets the atomicity requirement—that is, if each attribute and column has a single value and the values of the table cannot be broken down into smaller, more precise components—it is said to be in the First Normal Form. To put it another way, each column should only contain a single item of data; rows should not contain arrays or recurring groupings of values.

In this context, the term "atomicity" refers to the requirement that a database cell only store a single-valued property and cannot contain multiple values.

numerous-valued attributes, attributes made up of numerous sub-attributes, and attributes that combine both are restricted by the first normal form in database normalization. Thus, these attributes must be changed or eliminated, a relation cannot have multi-valued or composite attributes, and any such attributes must be divided into individual attributes to generate atomic values in order to meet the requirements of the first normal form.

For instance, the table includes details on the students, such as their age, course of study, roll number, and name.

	rollno	name	course	age
▶	1	Rahul	c/c++	22
	2	Harsh	java	18
	3	Sahil	c/c++	23
	4	Adam	c/c++	22
	5	Lisa	java	24
	6	James	c/c++	19
*	NULL	NULL	NULL	NULL

The presence of two separate values in the course column of the student records table results in a violation of the First Normal Form. The table has been altered, creating a new table, in order to guarantee adherence to the First Normal Form.

	rollno	name	course	age
►	1	Rahul	c	22
	1	Rahul	c++	22
	2	Harsh	java	18
	3	Sahil	c	23
	3	Sahil	c++	23
	4	Adam	c	22
	4	Adam	c++	22
	5	Lisa	java	24
	6	James	c	19
	6	James	c++	19

To verify atomicity in the system—that is, that every column has a unique value—the First Normal Form is employed. The data is arranged into distinct atomic values by adhering to this normalization method, which also removes any repetitions or redundant groups. Data integrity is preserved as a result, and the system is capable of handling complicated data changes and manipulations with ease.

Second Normal Form(2NF):

A table must first meet the requirements of First Normal Form in order to meet those of Second Normal Form. In addition, the table cannot show partial dependency, which is the condition in which an attribute that is not prime depends on a suitable subset of the candidate key. Stated differently, no characteristic in the table should be based solely on a subset of the main key.

Now, using an example, let's understand the Second Normal Form.

Examine the following table: Location

	cust_id	storeid	store_location
►	1	D1	Toronto
	2	D3	Miami
	3	T1	California
	4	F2	Florida
	5	H3	Texas

Store location is a non-key attribute of the Location database, which presently contains a composite primary key made up of customer ID and stored. Nonetheless, the feature of store location is directly decided by

the fundamental key component known as the storied characteristic. This table does not comply with second normal form standards as a result. It is required to divide the Location table into two distinct tables in order to resolve this problem and guarantee that the second normal form is satisfied. As a consequence, two unique tables will be created, one for store IDs and customer IDs and the other for store IDs and their corresponding locations, both of which precisely capture the pertinent data and connections.

	cust_id	storeid
▶	1	D1
	2	D3
	3	T1
	4	F2
	5	H3

	storeid	store_location
▶	D1	Toronto
	D3	Miami
	T1	California
	F2	Florida
	H3	Texas

Third Normal Form(3NF):

For a table to be evaluated in Third Normal Form, it must satisfy two extra requirements in addition to the requirements of Second Normal Form. In order to prevent transitive dependencies, the second criterion stipulates that non-prime attributes cannot rely on non-prime qualities that are not included in the candidate key inside the same table. When $A \rightarrow C$ indirectly through $A \rightarrow B$ and $B \rightarrow C$, where B is not functionally dependent on A, a transitive dependency is created. Attaining Third Normal Form primarily aims to ensure data integrity and minimize redundancy.

Let's take a look at a student table, for example, where the columns are the student's address, subject name, student ID, and student name.

	stu_id	name	subid	sub	address
▶	1	Arun	11	SQL	Delhi
	2	Varun	12	Java	Bangalore
	3	Harsh	13	C++	Delhi
	4	Keshav	12	Java	Kochi

You must now divide the table as indicated below in order to convert it to the third normal form: It is evident from the above student table that the sub_id attribute defines the topic (sub), and the stu_id property determines the sub_id attribute. This suggests the existence of a transitive functional reliance on sub and stu_id. Consequently, the table fails to meet the requirements for the third normal form. The table needs to be broken into distinct tables, each of which represents a distinct entity or relationship, in order to comply with the third normal form. The table in this instance can

be split into two tables: one containing the topic information (sub_id and sub) and another containing the student-subject relationship (stu_id and sub_id):

	stu_id	name	subid	address
▶	1	Arun	11	Delhi
	2	Varun	12	Bangalore
	3	Harsh	13	Delhi
	4	Keshav	12	Kochi

	subid	subject
▶	11	SQL
	12	java
	13	C++
	12	Java

The two tables show how the non-key qualities depend entirely on and are decided by the primary key. The name, sub_id, and address columns in the first table are all directly related to the stu_id. Similarly, the second table shows that the sub column depends only on the sub_id.

4.4.3 Sanitization

To ensure that no remaining data can be recovered even after thorough forensic investigation, data sanitization entails the safe and permanent removal of sensitive material from datasets and media. Although there are many uses for data sanitization, the two most common ones are the disposal of electronic devices that have reached the end of their useful lives and the sharing and utilization of sizable datasets that include sensitive data. Physical destruction, cryptographic erasure, and data erasure are the three primary methods for removing personal data from equipment. Although some people would assume that data sanitization is limited to electronic media, it actually encompasses a wide range of physical media, including paper copies. For electronic files, these data kinds are referred to as soft, and for hard copies on paper medium, as physical media.

4.4.4 Indexing

By reducing the amount of disk accesses needed to process a query, indexing helps databases operate at their best. One kind of data structure is the index. It helps you swiftly find and access data in a database table.

- **Primary Index** – An ordered data file defines the primary index. A key field is used to sort the data file. The primary key of the relation is typically the key field.
- **Secondary Index** – A field that is a candidate key and has a unique value in each record, or a non-key with duplicate values, can be used to create a secondary index.
- **Clustering Index** – An ordered data file defines the clustering index. A non-key field determines the order of the data file.

4.5 TABLE DESIGN

1.Tbl_allapp_user

Primary key: **id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier for the user
2	first_name	Varchar(150)	NOT_NULL	User's First Name
3	last_name	Varchar(150)	NOT_NULL	User's Last Name
4	email	EmailField	UNIQUE	User's Email
5	password	Varchar(150)	NOT_NULL	Password
6	username	Varchar(128)	UNIQUE	Unique Username
7	dob	DateField	NOT_NULL	Date of Birth
8	is_superuser	BOOL	NOT_NULL	Whether the user is a superadmin
9	Is_patient	BOOL	NOT_NULL	Whether the user is a patient
10	Is_doctor	BOOL	NOT_NULL	Whether the user is a doctor

2.Tbl_allapp_patient

Primary key : **id**

Foreign key: **user_id** references table **Tbl_allapp_user**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier for the patient
2	first_name	Varchar(150)	NOT_NULL	Patient's First Name
3	last_name	Varchar(150)	NOT_NULL	Patient's Last Name
4	role	Varchar(150)	UNIQUE	Patient's Email
5	user_id	INT(11)	FOREING KEY	References the user associated with this profile
6	username	Varchar(150)	UNIQUE	Unique Username

7	dob	DATE	NOT_NULL	Date of Birth
8	email	EmailField	UNIQUE	Patient's Email

3.Tbl_allapp_doctor

Primary key : **id**

Foreign key: **user_id** references table **Tbl_allapp_user**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier for the doctor
2	first_name	Varchar(150)	NOT_NULL	Doctor's First Name
3	last_name	Varchar(150)	NOT_NULL	Doctor's Last Name
4	role	Varchar(150)	UNIQUE	Doctor's Email
5	user_id	Varchar(150)	FOREING KEY	References the user associated with this profile
6	username	Varchar(150)	UNIQUE	Unique Username
7	dob	DATE	NOT_NULL	Date of Birth
8	email	EmailField	UNIQUE	Doctor's Email

4.Tbl_allapp_doctoradditionaldetails

Primary key : **id**

Foreign key: **doctor_id** references table **Tbl_allapp_doctor**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier
2	picture	Varchar(150)	NOT_NULL	Doctor's image
3	registration_number	Varchar(150)	UNIQUE	Doctor's Registration number
4	experience	INT(11)	NOT_NULL	Doctor's Experience
5	speciality	Varchar(150)	NOT_NULL	Doctor's Speciality
6	education	Varchar(128)	NOT_NULL	Doctor's Education

7	doctor_id	INT(11)	FOREING KEY	References the doctor associated with this profile
---	-----------	---------	-------------	--

5.Tbl_allapp_appointment

Primary key : **id**

Foreign key: **doctor_id** references table **Tbl_allapp_doctor**

Foreign key: **patient_id** references table **Tbl_allapp_patient**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier
2	date	DateField	NOT_NULL	Appointment date
3	time_slot	Varchar(150)	UNIQUE	Doctor's Registration number
4	patient_name	Varchar(150)	NOT_NULL	Patient's name
5	Patient_email	EmailField	FOREING KEY	Patient's email
6	doctor_id	INT(11)	FOREING KEY	References the doctor associated with this profile
7	patient_id	INT(11)	FOREING KEY	References the patient associated with this profile

6.Tbl_allapp_medicine

Primary key : **id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier
2	med_image	Varchar(150)	NOT_NULL	Medicine image
3	name	Varchar(150)	NOT_NULL	Medicine's name
4	price	DecimalField	NOT_NULL	Price of medicine
5	company_name	Varchar(150)	NOT_NULL	Company that produces that medicine
6	medicine_info	Varchar(150)	NOT_NULL	Medicine's information
7	quantity	INT(11)	NOT_NULL	Quantity

7.Tbl_allapp_cart

Primary key : **id**

Foreign key : **patient_id** references table **Tbl_allapp_patient**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier
2	patient_id	Varchar(150)	FOREING KEY	References the patient associated.

8.Tbl_allapp_cartitem

Primary key : **id**

Foreign key : **cart_id** references table **Tbl_allapp_cart**

Foreign key : **medicine_id** references table **Tbl_allapp_medicine**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier
2	quantity	INT(11)	NOT_NULL	Amount of medicine.
3	cart_id	INT(11)	FORIENG KEY	Reference to the cart items
4	medicine_id	INT(11)	FORIENG KEY	Reference to the medicine

9.Tbl_allapp_consultaionrequest

Primary key : **id**

Foreign key : **doctor_id** references table **Tbl_allapp_doctor**

Foreign key : **patient_id** references table **Tbl_allapp_patient**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier
2	image	Varchar(150)	NOT_NULL	Image of affected area
3	description	Varchar(150)	NOT_NULL	Description of problem
4	timestamp	DateTimeField	NOT_NULL	Time and date of request

6	doctor_id	INT(11)	FORIENG KEY	References the doctor associated with this profile
7	patient_id	INT(11)	FORIENG KEY	References the patient associated with this profile

10. Tbl_allapp_order

Primary key : **id**

Foreign key : **medicine_id** references table **Tbl_allapp_medicine**

Foreign key : **patient_id** references table **Tbl_allapp_patient**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier
2	quantity	INT(11)	NOT_NULL	Amount of medicine.
3	order_date	DateTimeField	NOT_NULL	Time of the order
4	amount_paid	DecimalField	NOT_NULL	The amount paid
5	medicine_id	INT(11)	FORIENG KEY	Reference to the medicine
6	patient_id	INT(11)	FORIENG KEY	Reference to the patient

11. Tbl_allapp_address

Primary key : **id**

Foreign key : **patient_id** references table **Tbl_allapp_patient**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	id	INT(11)	PRIMARY_KEY	Unique identifier
2	quantity	INT(11)	NOT_NULL	Amount of medicine.
3	mobile_number	INT(15)	NOT_NULL	Patient's number
4	pin_code	INT(11)	NOT_NULL	Pin code
5	address	Varchar(250)	NOT_NULL	Patient's address
6	district	Varchar(150)	NOT_NULL	Patient's district
7	state	Varchar(150)	NOT_NULL	Patient's state
8	patient_id	INT(11)	FORIENG KEY	Reference to the patient

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is a means of verifying that a computer program functions as intended. To ensure that the software performs as intended, we test it. Validation is the process of examining or testing products, such as software, to ensure that they fulfill the specifications and norms that they are meant to adhere to. Software testing is a technique to evaluate a program's functionality. It complements other approaches such as program walking and checking. Verifying that the user received what they intended is known as validation. There are numerous guidelines that can be used as goals for testing.

The process of running a software with the goal of identifying errors is called testing.

- A test case with a high likelihood of discovering an unidentified fault is good..
- A test that finds an unidentified fault is considered successful.
- A well-functioning test that adheres to its objectives can identify errors in the program. That was demonstrated by the test.
- The computer program is performing well and operating as intended.
- Three methods exist for testing the program.
- For accuracy
- For effectiveness of implementation
- For computational complexity

5.2 TEST PLAN

A Unit testing is a significant organize of computer program confirmation that concentrates on testing person components or modules, which are the principal building pieces of computer program plan. This testing process involves investigating the component-level plan determinations to identify any blunders inside the module's boundaries by analyzing key control ways. The complexity of the test and the untested areas are decided amid the prepare of unit testing, which is planned to be white-box centered, and numerous components can be tried at the same time. To guarantee legitimate usefulness of the program unit being tried, the measured interface experiences checks to guarantee adjust information stream in and out. Moreover, the astuteness of incidentally put away information in the nearby information structure is inspected throughout the algorithm's execution.

5.2.1 UNIT TESTING

Software components or modules, which are the smallest units of a software design, are tested through unit testing. Using the design guidance, test key control pathways in a module to identify issues. This implies the level of difficulty of the tests for each minor component of a program and the components that haven't been put to the test. One kind of testing called unit testing examines the internal workings of the code and can include completed concurrently for several program components. We must first determine whether data is correctly flowing between the various components of the computer software before beginning any more tests. All other checks are useless if the data isn't entering and exiting the system appropriately. When creating anything, it's critical to consider potential difficulties and develop a plan to address them. This could entail rerouting the procedure or ending it altogether.

5.2.2 INTEGRATION TESTING

Integration testing is a systematic approach to building a program system and conducting evaluate simultaneously for detect faults related to objects. The objective is for combine individually evaluation comp into a program system that aligns with the project. The whole program is then tested as a unified whole. Resolving errors in integration testing can be challenging due to the complexity of identifying their causes within the extensive scope of the program. Once unit testing is completed on the system's modules, they are merged and tested to identify any inconsistencies in the interfaces.

5.2.3 VALIDATION TESTING OR SYSTEM TESTING

This concludes the testing phase. During this test, the whole system was examined to ensure that all of the various building blocks and instructions interacted as intended. This type of testing is called system testing or black box testing. One technique to make sure the software performs as intended is to use black box testing. Through the use of several input formats, black box testing assists software engineers in identifying every issue present in a program. Black box testing examines code for errors in startup and termination, performance, data access, functions, and interfaces.

5.2.4 OUTPUT TESTING OR USER ACCEPTANCE TESTING

When a system is put to the test for user approval, it should meet the needs of the company. While it is being developed, the software should stay in communication with the user and the perspective system, making adjustments as needed.

In light of the following, this is done:

- Input Screen Designs,

- Output Screen Designs,

The testing mentioned above uses a variety of test data types. In order to test the system, test data preparation is essential. The system under study is tested using the test data that has been prepared. Errors in the test data are found during system testing and fixed using the previously mentioned testing procedures; the adjustments are also recorded for future reference.

5.2.5 AUTOMATION TESTING

Automation testing is a software testing methodology that involves running a test case suite through the use of specialized automated testing software tools. In essence, it's an examination to confirm that the hardware or software performs as intended. It checks for flaws, errors, and any other problems that might occur during the development of a product. While manual testing is possible for certain types of testing, including regression or functional testing, doing testing automatically has more advantages. Any time of day can be used for automation testing. It looks at the software using scheduled sequences. After that, it provides an analysis of the findings, which can be contrasted with data from previous test runs. Programming languages used by automation developers often include the following ones: C++, JavaScript, and Ruby.

5.2.6 SELENIUM TESTING

An open-source automated testing framework called Selenium is used to validate web applications on many platforms and browsers. Test scripts can be created with Selenium in a number of computer languages, including Python, C#, and Java. While working on a web application that needed frequent testing in 2004, Thought Works engineer Jason Huggins created Selenium. His "JavaScriptTestRunner" JavaScript application was designed to automate browser functions and boost testing effectiveness. Since then, a group of collaborators has continued to develop selenium. Apart from Selenium, Cucumber is another widely used tool for automated testing. A software testing framework that is available for free that facilitates behavior-driven development is called Cucumber (BDD). It enables the development of executable specifications in the Gherkin human-readable format. Cucumber's ability to close the communication gap between technical teams and business stakeholders is one of its benefits. Cucumber makes it easier to collaborate and communicate effectively during the testing process by providing a common language. It facilitates a common understanding of the specifications and aids in making sure the created software achieves the desired business objectives. Cucumber and Selenium can be combined to maximize the advantages of each tool. Cucumber offers a formal framework for planning and carrying out tests, while Selenium is used to interface with web browsers and automate browser behaviors. This

combination makes it possible to create end-to-end tests that confirm how web applications behave in various browsers and platforms, using a business-readable and maintainable format.

Test Case 1

Code :

```
package definition;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.openqa.selenium.NoSuchSessionException;
import org.junit.Assert; // Import the Assert class

public class LoginSteps {

    WebDriver driver;
    boolean testPassed = true; // Initialize a flag to check test status

    @Given("browser is open")
    public void browser_is_open() {
        System.setProperty("webdriver.gecko.driver", "C:\\Users\\ashwin\\eclipse-
workspace\\cucumberjava\\src\\test\\resources\\Drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("user is on the login page")
    public void user_is_on_the_login_page() {
        driver.get("http://127.0.0.1:8000/login/");
    }
}
```

```
@When("user enters username and password")
public void user_enters_username_and_password() {
    // Enter username and password using the elements from code snippet 1
    driver.findElement(By.id("email")).sendKeys("ashwinspillai230501@gmail.com");
    driver.findElement(By.id("password")).sendKeys("A@123456");
}

@And("user clicks on login")
public void user_clicks_on_login() {
    // Click on the login button using the element from code snippet 1
    driver.findElement(By.cssSelector("form.login-form button")).click();
}

@Then("user is navigated to the home page")
public void user_is_navigated_to_the_home_page() throws Exception {
    try {
        // Add a delay to ensure the page loads completely
        // Thread.sleep(2000);

        // Check if the test is passed
        if (driver.getCurrentUrl().contains("http://127.0.0.1:8000/")) {
            System.out.println("Test Passed: User is on the home page");
        } else {
            System.out.println("Test Failed: User is not on the home page");
            testPassed = false;
        }
        // Assert the test status using JUnit's Assert class
        Assert.assertTrue(testPassed);
    } catch (NoSuchSessionException e) {
        // Ignore the exception as the test has already completed
    } finally {
        // Close the driver after all assertions are made
        if (driver != null) {
            driver.quit();
        }
    }
}
```

```

    }
  }
}

```

Output :

```

Dec 03, 2023 2:20:28 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check login is successful with valid credentials # src/test/resources/Features/login.feature:3
1701593430386  geckodriver      INFO    Listening on 127.0.0.1:61589
1701593430710  mozrunner::runner      INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "-no-remote" "-profile"
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1701593431730  Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1701593432854  Marionette      INFO    Listening on port 53467
Read port: 53467
1701593433231  RemoteAgent     WARN    TLS certificate errors will be ignored for this session
Dec 03, 2023 2:20:35 PM org.opengaselenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
  Given browser is open                                # definition.LoginSteps.browser_is_open()
  And user is on the login page                        # definition.LoginSteps.user_is_on_the_login_page()
  When user enters username and password                # definition.LoginSteps.user_enters_username_and_password()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
  And user clicks on login                             # definition.LoginSteps.user_clicks_on_login()
Test Passed: User is on the home page
1701593438859  Marionette      INFO    Stopped listening on port 53467
JavaScript warning: https://code.jquery.com/jquery-3.4.1.min.js, line 2: Script terminated by timeout at:
i@https://code.jquery.com/jquery-3.4.1.min.js:2:87473

Dynamically enable window occlusion 1
  Then user is navigated to the home page              # definition.LoginSteps.user_is_navigated_to_the_home_page()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m11.102s

```

Test Report

Test Case 1					
Project Name: AllergyCare					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Ashwin S Pillai		
Test Priority(Low/Medium/High): High			Test Designed Date: 01-12-23		
Module Name: Login Page			Test Executed By : Ms. Lisha Varghese		
Test Title : Verify login with valid email and password			Test Execution Date: 29-03-2024		
Description: Test the Login Page					
Pre-Condition : User has valid email id and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be	Login page displayed	Pass

			displayed		
2	Provide Valid email	Email: ashwinspillai230501@gmail.com	User should be able to Login	User Logged in and navigated to the dashboard with records	Pass
3	Provide Valid Password	Password: A@123456			
4	Click on Sign In button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2:**Code :**

```

package definition;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.junit.Assert;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class LoginAndProfileUpdateSteps {
    WebDriver driver;
    boolean testPassed = true;

    @Given("the user is on the login page")
    public void the_user_is_on_the_login_page() {
        System.setProperty("webdriver.gecko.driver", "C:\\Users\\ashwin\\eclipse-
workspace\\cucumberjava\\src\\test\\resources\\Drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
    }

```

```
        driver.manage().window().maximize();
        driver.get("http://127.0.0.1:8000/login/");
    }
    @When("the user enters valid credentials and logs in")
    public void the_user_enters_valid_credentials_and_logs_in() {

driver.findElement(By.id("email")).sendKeys("ashwinspillai230501@gmail.com");
        driver.findElement(By.id("password")).sendKeys("A@123456");
        driver.findElement(By.cssSelector("form.login-form button")).click();
        // Directly navigate to the profile page
        driver.get("http://127.0.0.1:8000/patient_profile/");
    }
    @Then("the user is navigated to the profile page")
    public void the_user_is_navigated_to_the_profile_page() throws Exception {
        try {
            WebElement profileHeader =
driver.findElement(By.xpath("//h2[text()='My Profile']"));
            if (profileHeader.isDisplayed()) {
                System.out.println("Test Passed: User is on the profile page");
            } else {
                System.out.println("Test Failed: User is not on the profile page");
                testPassed = false;
            }

            Assert.assertTrue(testPassed);
        } finally {
            // If you want to keep the browser open for debugging purposes, comment
out the following line
            if (driver != null) {
                // driver.quit();
            }
        }
    }
}
```



```

@When("the user updates the profile with username {string}")
public void the_user_updates_the_profile_with_username(String newUsername) {
    // Implement the code to update the profile with the new username
    // You may need to locate and interact with the relevant elements on the profile
update page

    // For example, assuming the username input field has an id "username" on the
profile update page
    WebElement usernameField = driver.findElement(By.id("username"));
    usernameField.clear();
    usernameField.sendKeys(newUsername);
}

@And("the user clicks on the Update Profile button")
public void the_user_clicks_on_the_update_profile_button() {
    // Use the class name to locate the button
    driver.findElement(By.cssSelector(".btn-update-profile")).click();
}
}

```

Output :

```

Dec 03, 2023 2:42:13 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Login, Navigate to Profile, and Update Profile # src/test/resources/Features/LoginAndProfileUpdate.feature:3
1701594735704 geckodriver INFO Listening on 127.0.0.1:32036
1701594736184 mozrunner::runner INFO Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "-no-remote" "-profile"
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1701594736957 Marionette INFO Marionette enabled
Dynamically enable window occlusion 0
1701594737184 Marionette INFO Listening on port 54219
Read port: 54219
1701594737514 RemoteAgent WARN TLS certificate errors will be ignored for this session
Dec 03, 2023 2:42:20 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
  Given the user is on the login page # definition.LoginAndProfileUpdateSteps.the_user_is_on_the_login_page()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/static/js/main.js, line 23: TypeError: $(...).datetimepicker is not a function
  When the user enters valid credentials and logs in # definition.LoginAndProfileUpdateSteps.the_user_enters_valid_credentials_and_logs_in()
Test Passed: User is on the profile page
  Then the user is navigated to the profile page # definition.LoginAndProfileUpdateSteps.the_user_is_navigated_to_the_profile_page()
  When the user updates the profile with username "AshwinSE" # definition.LoginAndProfileUpdateSteps.the_user_updates_the_profile_with_username(java.lang.
JavaScript error: http://127.0.0.1:8000/static/js/main.js, line 23: TypeError: $(...).datetimepicker is not a function
  And the user clicks on the Update Profile button # definition.LoginAndProfileUpdateSteps.the_user_clicks_on_the_update_profile_button()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m11.148s

```

Test report

Test Case 2					
Project Name: AllergyCare					
Update Profile Test Case					
Test Case ID: Test_2			Test Designed By: Ashwin S Pillai		
Test Priority(Low/Medium/High): High			Test Designed Date: 01-12-23		
Module Name: Update Page			Test Executed By : Ms. Lisha Varghese		
Test Title : Verify update profile functionality			Test Execution Date: 12-10-2023		
Description: Test the update profile feature					
Pre-Condition : Require valid profile fields and update button					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to profile Page		Profile page should be displayed	Profile page displayed	Pass
2	Change the current username to provided username	User Name: AshwinSP	User should be able to change the username to provided username	The username should ne updated	
3	Press the submit button				
Post-Condition: User is able to successfully update profile.					

Test Case 3:**Code :**

```

package definition;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.junit.Assert;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;

```

```
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
public class MedicineSearchSteps2 {

    WebDriver driver;
    boolean testPassed = true;

    @Given("the user is on the login page for Medicine Search")
    public void the_user_is_on_the_login_page_for_medicine_search() {
        System.setProperty("webdriver.gecko.driver", "C:\\Users\\ashwin\\eclipse-
workspace\\cucumberjava\\src\\test\\resources\\Drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.get("http://127.0.0.1:8000/login/");
    }

    @When("the user enters valid credentials and logs in for Medicine Search")
    public void the_user_enters_valid_credentials_and_logs_in_for_medicine_search() {
        driver.findElement(By.id("email")).sendKeys("d@gmail.com");
        driver.findElement(By.id("password")).sendKeys("D@123456");
        driver.findElement(By.cssSelector("form.login-form button")).click();
    }

    @And("the user navigates to the Medicines page")
    public void the_user_navigates_to_the_medicines_page() {
        driver.findElement(By.linkText("Medicines")).click();
    }

    @And("the user searches for the medicine {string}")
    public void the_user_searches_for_the_medicine(String medicineName) {
        WebElement searchInput = driver.findElement(By.id("search-medicine"));
        WebElement searchButton = driver.findElement(By.cssSelector(".search-
button"));
    }
```

```

        searchInput.sendKeys(medicineName);

        searchButton.click();
    }

    @Then("the user should see {string} in the list of medicines")
    public void the_user_should_see_in_the_list_of_medicines(String medicineName)
    {
        WebElement medicineCard = driver.findElement(By.xpath("//h2[text()='\" +
        medicineName + "\"]"));

        Assert.assertTrue(medicineCard.isDisplayed());
    }
}

```

Output :

```

Dec 03, 2023 3:23:21 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: User searches for a specific medicine # src/test/resources/Features/MedicineSearch.feature:3
1701597203998 geckodriver INFO Listening on 127.0.0.1:50231
1701597204489 mozrunner::runner INFO Running command: "C:\\Program Files\\Mozilla Firefox\\Firefox.exe" "--marionette" "--no-remote" "--profile"
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1701597205549 Marionette INFO Marionette enabled
Dynamically enable window occlusion 0
1701597205856 Marionette INFO Listening on port 54565
Read port: 54565
1701597206322 RemoteAgent WARN TLS certificate errors will be ignored for this session
Dec 03, 2023 3:23:30 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
    Given the user is on the login page for Medicine Search # definition.MedicineSearchSteps2.the_user_is_on_the_login_page_for_medicine_sear
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
    When the user enters valid credentials and logs in for Medicine Search # definition.MedicineSearchSteps2.the_user_enters_valid_credentials_and_logs_in_f
JavaScript error: http://127.0.0.1:8000/static/js/main.js, line 23: TypeError: $(...).datetimepicker is not a function
    And the user navigates to the Medicines page # definition.MedicineSearchSteps2.the_user_navigates_to_the_medicines_page()
    And the user searches for the medicine "Burnheal" # definition.MedicineSearchSteps2.the_user_searches_for_the_medicine(java.lang.St
    Then the user should see "Burnheal" in the list of medicines # definition.MedicineSearchSteps2.the_user_should_see_in_the_list_of_medicines(ja

1 Scenarios (1 passed)
5 Steps (5 passed)
0m12.596s

```

Test report

Test Case 3					
Project Name: AllergyCare					
Search Medicine Test Case					
Test Case ID: Test_3			Test Designed By: Ashwin S Pillai		
Test Priority(Low/Medium/High): High			Test Designed Date: 01-12-23		
Module Name: Search Medicine Page			Test Executed By : Ms. Lisha Varghese		
Test Title : Verifying the search option in medicine			Test Execution Date: 12-10-2023		
Description: Test the search feature					
Pre-Condition : Require valid search data					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to medicine Page		Medicine page should be displayed	medicinee page displayed	Pass
2	Search for the given medicine name	User Name: Burnheal	User should be able to get the searched medicine	Burnheal medicine was found.	Pass
3	Press the submit button				
Post-Condition: User is able to successfully get the medicine.					

Test Case 4:**Code :**

```

package definition;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.junit.Assert;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

```

```
public class TestSubmitTestimonial {

    WebDriver driver;

    @Given("the user is on the login page for Testimonial")
    public void the_user_is_on_the_login_page_for_testimonial() {
        System.setProperty("webdriver.gecko.driver", "C:\\Users\\ashwin\\eclipse-
workspace\\cucumberjava\\src\\test\\resources\\Drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.get("http://127.0.0.1:8000/login/");
    }

    @When("the user logs in with email {string} and password {string}")
    public void the_user_logs_in_with_email_and_password(String email, String
password) {
        driver.findElement(By.id("email")).sendKeys(email);
        driver.findElement(By.id("password")).sendKeys(password);
        driver.findElement(By.cssSelector("form.login-form button")).click();
    }

    @And("the user navigates to the Doctors page")
    public void the_user_navigates_to_the_doctors_page() {
        driver.findElement(By.linkText("Doctors")).click();
    }

    @And("the user clicks on the Submit Testimonial for the first doctor")
    public void the_user_clicks_on_the_submit_testimonial_for_the_first_doctor() {
        driver.findElement(By.cssSelector(".btn-info")).click();
    }

    @And("the user selects {int} stars and provides feedback {string}")
    public void the_user_selects_stars_and_provides_feedback(int stars, String
```

```

feedback) {

    for (int i = 1; i <= stars; i++) {

        WebElement star = driver.findElement(By.id("star" + i));

        star.click();

    }

    WebElement feedbackInput = driver.findElement(By.id("id_feedback"));

    feedbackInput.sendKeys(feedback);

}

@And("the user submits the testimonial")

public void the_user_submits_the_testimonial() {

    WebElement submitButton = driver.findElement(By.cssSelector(".btn-

primary"));

    submitButton.click();

}

}

```

Output :

```

Dec 03, 2023 3:35:22 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: User submits a testimonial # src/test/resources/Features/submit_testimonial.feature:3
1701597924788 geckodriver INFO Listening on 127.0.0.1:42509
1701597925276 mozrunner:runner INFO Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "-no-remote" "-profile"
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1701597926545 Marionette INFO Marionette enabled
Dynamically enable window occlusion 0
1701597927936 Marionette INFO Listening on port 54743
Read port: 54743
1701597928399 RemoteAgent WARN TLS certificate errors will be ignored for this session
Dec 03, 2023 3:35:31 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
    Given the user is on the login page for Testimonial # definition.TestSubmitTestimonial.the_user_is_on_the_login_page_for_testimonial()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
    When the user logs in with email "d@gmail.com" and password "D@123456" # definition.TestSubmitTestimonial.the_user_logs_in_with_email_and_password(java.
JavaScript error: http://127.0.0.1:8000/static/js/main.js, line 23: TypeError: $(...).datetimepicker is not a function
    And the user navigates to the Doctors page # definition.TestSubmitTestimonial.the_user_navigates_to_the_doctors_page()
JavaScript error: resource://gre/modules/XULStore.sys.mjs, line 60: Error: Can't find profile directory.
JavaScript error: http://127.0.0.1:8000/static/js/main.js, line 23: TypeError: $(...).datetimepicker is not a function
    And the user clicks on the Submit Testimonial for the first doctor # definition.TestSubmitTestimonial.the_user_clicks_on_the_submit_testimonial_for_
    And the user selects 5 stars and provides feedback "Kind words" # definition.TestSubmitTestimonial.the_user_selects_stars_and_provides_feedback(i
JavaScript error: http://127.0.0.1:8000/static/js/main.js, line 23: TypeError: $(...).datetimepicker is not a function
    And the user submits the testimonial # definition.TestSubmitTestimonial.the_user_submits_the_testimonial()

1 Scenarios (1 passed)
6 Steps (6 passed)
0m14.147s

```

Test report

Test Case 4					
Project Name: AllergyCare					
Submit Testimonial Test Case					
Test Case ID: Test_4			Test Designed By: Ashwin S Pillai		
Test Priority(Low/Medium/High): High			Test Designed Date: 01-12-23		
Module Name: Submit Testimonial			Test Executed By : Ms. Lisha Varghese		
Test Title : Verifying the testimonial			Test Execution Date: 12-10-2023		
Description: Test the testimonial feature					
Pre-Condition : Require valid doctors					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to doctor Page		All the doctors should be present	Doctor page was viewed	Pass
2	Press the submit testimonial button		It should lead to the respective doctors testimonial page	Testimonial page was viewed	Pass
3	Give rating and feedback	Rating: 5 Feedback:Kind words	Patient should be able to give rating and feedback successfully	Successfully provided rating and feedback	Pass
4	Press the submit button				
Post-Condition: User is able to successfully give the testimonial					

Test Case 5:**Code :**

```

package definition;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.junit.Assert;

```



```
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class TestCancelAppointment {

    WebDriver driver;

    @Given("the user is on the login page for Cancel appointment")
    public void the_user_is_on_the_login_page_cancel_appointment() {
        System.setProperty("webdriver.gecko.driver", "C:\\Users\\ashwin\\eclipse-
workspace\\cucumberjava\\src\\test\\resources\\Drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.get("http://127.0.0.1:8000/login/");
    }

    @When("the user logs in with credentials email {string} and password {string}")
    public void the_user_logs_in_with_email_and_for_cancel_appointment(String
email, String password) {
        driver.findElement(By.id("email")).sendKeys(email);
        driver.findElement(By.id("password")).sendKeys(password);
        driver.findElement(By.cssSelector("form.login-form button")).click();
    }

    @And("the user navigates to the Appointments page")
    public void the_user_navigates_to_the_appointments_page() {
        WebElement dropdown =
driver.findElement(By.xpath("//a[contains(text(),'Appointments')]"));
        dropdown.click();
    }
}
```

```

    @And("the user cancels the appointment with Doctor: Ilow Peter")
    public void the_user_cancels_the_appointment_with_doctor_iloc_peter() {
        WebElement cancelButton =
driver.findElement(By.xpath("//h2[contains(text(),'Doctor: Ilow Peter')]/following-
sibling::form/button"));
        cancelButton.click();
    }

    @Then("the appointment with Doctor: Ilow Peter should be cancelled")
    public void the_appointment_with_doctor_iloc_peter_should_be_cancelled() {
        WebElement successMessage =
driver.findElement(By.xpath("//div[contains(@class,'alert-
success')]/strong[contains(text(),'Booking has been cancelled.')]"));
        Assert.assertNotNull(successMessage);
    }
}

```

Output :

```

Dec 03, 2023 2:42:13 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Login, Navigate to Profile, and Update Profile # src/test/resources/Features/LoginAndProfileUpdate.feature:3
1701594735704 geckodriver INFO Listening on 127.0.0.1:32036
1701594736184 mozrunner::runner INFO Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "-no-remote" "-profile"
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1701594736957 Marionette INFO Marionette enabled
Dynamically enable window occlusion 0
1701594737184 Marionette INFO Listening on port 54219
Read port: 54219
1701594737514 RemoteAgent WARN TLS certificate errors will be ignored for this session
Dec 03, 2023 2:42:20 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
  Given the user is on the login page # definition.LoginAndProfileUpdateSteps.the_user_is_on_the_login_page()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/static/js/main.js, line 23: TypeError: $(...).datetimepicker is not a function
  When the user enters valid credentials and logs in # definition.LoginAndProfileUpdateSteps.the_user_enters_valid_credentials_and_logs_in()
Test Passed: User is on the profile page
  Then the user is navigated to the profile page # definition.LoginAndProfileUpdateSteps.the_user_is_navigated_to_the_profile_page()
  When the user updates the profile with username "AshwinSP" # definition.LoginAndProfileUpdateSteps.the_user_updates_the_profile_with_username(java.lang.
JavaScript error: http://127.0.0.1:8000/static/js/main.js, line 23: TypeError: $(...).datetimepicker is not a function
  And the user clicks on the Update Profile button # definition.LoginAndProfileUpdateSteps.the_user_clicks_on_the_update_profile_button()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m11.148s

```

Test report

Test Case 5					
Project Name: AllergyCare					
Submit Testimonial Test Case					
Test Case ID: Test_5			Test Designed By: Ashwin S Pillai		
Test Priority(Low/Medium/High): High			Test Designed Date: 01-12-23		
Module Name: Cancel Appointment			Test Executed By : Ms. Lisha Varghese		
Test Title : Verifying appointment cancellation			Test Execution Date: 12-10-2023		
Description: Test the cancellation feature					
Pre-Condition : Require valid doctors					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to appointment page		All the scheduled appointments should be present	Appointment page was viewed	Pass
2	Press the cancel button of the doctor named 'Ilow Peter'		Cancellation button should be present	Cancellation button was present	Pass
3	The appointment was successfully cancelled		Appointment with Ilow Peter cancelled	Successfully Cancelled an appointment	Pass
Post-Condition: User is able to successfully cancel an appointment					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

In order to successfully complete a project and provide users trust in the efficacy and accuracy of the system, the theoretical design must be translated into a working system during the implementation stage. During this phase, the main priorities are user training and documentation; conversion usually occurs concurrently with or after user training. The process of implementation, which entails turning the new system design into an operational one, can be chaotic and confusing if it is not properly organized or managed. The tasks required to transition from the old system to the new one whether it be a whole new or modified system are all included in the implementation process. Offering a dependable solution that satisfies organizational needs is essential.

Only when extensive testing has been completed and it has been determined that the system is operating as intended can it be put into use. System staff conduct feasibility checks; the more complicated the system being deployed, the more work will be needed in terms of system analysis and design for changeover, testing, and teaching and training.

The following tasks are part of the implementation state:

- Meticulous preparation.
- System and constraint investigation
- Methods design to accomplish the switchover

6.2 IMPLEMENTATION PROCEDURES

The ultimate installation of the program in its actual setting, as well as the system's functionality and satisfaction with its intended usage, are all considered aspects of software implementation. A software development project is often commissioned by a person who will not be using it. People have doubts about the software at first, but we need to make sure that resistance doesn't grow by making sure that:

- The active user's confidence in the program is increased; they need to understand the advantages of using the new system.
- Appropriate instructions are given to the user to ensure his comfort level with the application.

It is important for the user to understand that in order to access the system, the server software needs to be running on the server. The real process won't happen if the server object isn't up and running on the server.

6.2.1 USER TRAINING

Giving users the confidence to test and modify computer-based systems in order to ultimately achieve the desired goals is the aim of user training. As systems get more complex, training becomes increasingly crucial. Participants in user training are exposed to a range of critical tasks, including data entry, handling error alarms, database querying, and calling up procedures to generate reports, among other things.

6.2.2 TRAINING ON THE APPLICATION SOFTWARE

Before using the new application software, users must finish a basic computer literacy training course. It is important to demonstrate to them how to use the program panels, get support, fix mistakes in data entering, and amend previously entered data. The training should also address specific subjects related to the user group and how they use the system or its components. Program training needs to be tailored to the needs of different user groups and levels of hierarchy.

6.2.3 SYSTEM MAINTENANCE

The software maintenance phase, which comes after a software product has been effectively implemented and is producing meaningful work, is an integral aspect of the software development life cycle. To make sure the system can continue to adapt to changes in the system environment, maintenance is required. Although it is one task in software maintenance, error discovery is not the sole one. Maintenance also includes a wide range of other tasks, like introducing new features, improving current features, addressing bugs, and updating documentation. Over time, the software's functionality, dependability, and efficiency can all be maintained with the aid of efficient maintenance.

6.2.4 HOSTING

The act of offering a platform or service for the online storage, serving, and management of webpages, apps, or data is known as hosting. A popular type of hosting that enables people and businesses to publish their websites online is called web hosting. Web hosting comes in various flavors: cloud hosting, managed hosting, VPS hosting, shared hosting, and dedicated hosting. Because shared hosting entails sharing server resources with other websites, it is the most economical option. Whereas dedicated hosting gives you exclusive usage of a server, PS hosting gives you more freedom and control over the resources on the server. With managed hosting, a third-party provider handles the technical parts of hosting, freeing up organizations to concentrate on their main business operations. Selecting the best hosting company is essential since it has an impact on the uptime, security, and performance of websites. When choosing a hosting company,

factors like price, functionality, scalability, dependability, and support should be taken into account. In general, hosting is essential function in making it possible for companies and people to create an online presence and connect with a worldwide audience.

AWS Web hosting

Amazon Web Services (AWS) offers a range of web hosting services to suit the needs of businesses of all sizes. AWS provides a scalable and reliable infrastructure for hosting e-commerce platforms, online apps, and websites, among other things. Users can choose from a variety of hosting solutions, including Amazon Lightsail, Amazon S3, AWS Lambda, Amazon Elastic Beanstalk, and Amazon Elastic Compute Cloud (EC2). Elastic Beanstalk provides an easy-to-use web application delivery and scalability architecture, but EC2 gives you total control over the virtual servers. Lambda and other serverless computing services allow users to run code without the need to provision or manage servers. While Lightsail is a simple, cost-effective option for individuals who just need a basic website or application, Amazon S3 is an object storage service that can be used to store and retrieve files and static website content. Modern security features like network firewalls, SSL/TLS encryption, and distributed denial-of-service (DDoS) prevention are provided by AWS web hosting services, guaranteeing the security and high availability of online applications hosted on AWS.

EC2 (Elastic cloud compute)

Amazon Elastic Compute Cloud (EC2) is a web service provided by Amazon Web Services (AWS) that allows customers to rent virtual computing resources, such as virtual machines (VMs) or instances, to run their own applications. With EC2 instances, users may build them with various operating systems, networking configurations, and hardware configurations, giving them flexibility and scalability. Because EC2 instances are easy to build and terminate as needed and users just pay for the resources they use, they are an affordable option for businesses and individuals who require on-demand computing capacity. Moreover, AWS services like EC2 combine to offer a comprehensive cloud computing platform.

How to create an instance:

1. Launch the AWS Management Console, then log in to the EC2 dashboard. • To begin creating a new instance, select the "Launch Instance" option. To start a new instance, click the "Launch Instance" button.
2. Select the Amazon Machine Image (AMI) that will act as your instance's base.

Typically, this will be an operating system or software stack image that has already been

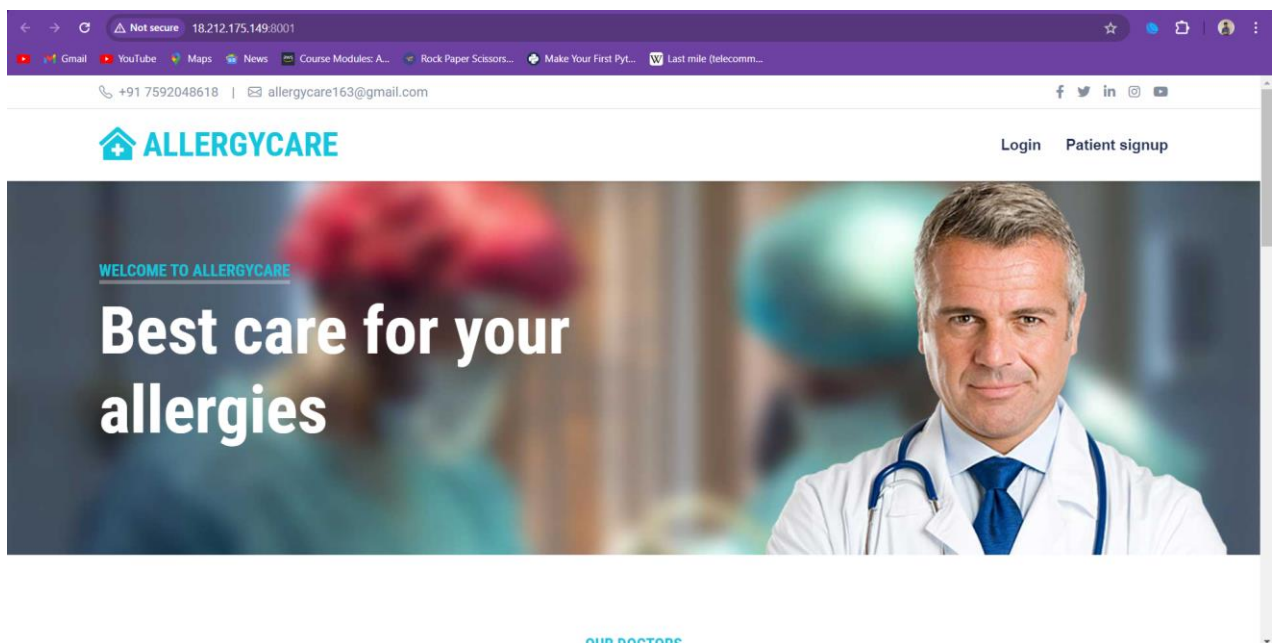
configured.

3. Select the instance type that you wish to use. This determines the CPU, memory, storage, and network capacity of your instance.
4. Configure the instance's parameters, such as the number of instances you want to run, security groups, and network setups.
5. List any extra storage devices or volumes that your instance needs.

Examine and configure any advanced settings (placement groups, user datascripsts, and IAM roles) that are required. After checking the instance's launch setup, which includes the instance type, storage, network configuration, and security groups, finally launch the instance.

Hosted Website:

Hosted Link: <http://18.212.175.149:8001/>



CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The AllergyCare initiative is a trailblazing innovation in allergy care services during a time of rapid technology growth. It redefines the landscape of allergy care by eschewing conventional procedures and embracing efficiency and user-friendliness. The main objective of AllergyCare is to provide a state-of-the-art online platform that streamlines the process of finding licensed allergy specialists and different allergy care modalities, hence improving the user's experience. In order to effectively manage allergy care options and keep the platform safe and current for users, administrators are essential. The system is easy to use and informative, allowing patients to simply schedule appointments, view the profiles of allergy specialists, and access a variety of allergy care modalities. The system is put through a comprehensive testing process to guarantee a smooth and reliable user experience, promoting allergies well-being and raising standard of living in general. Essentially, AllergyCare wants to be a full-featured, personalized, automated healing platform that helps users move closer to their overall health and allergy control objectives.

7.2 FUTURE SCOPE

In the future, AllergyCare has a great deal of room to expand and innovate in the allergy care industry. The platform might expand the range of allergy care services it offers by adding other modalities like lifestyle coaching, exercise regimens, and nutritional advice. The amalgamation of wearable data and health tracking functionalities can provide significant perspectives, augmenting the evaluation of allergy management advancement. experts.

- Using Machine Learning to Identify Allergens:

Use machine learning techniques to diagnose and identify allergies more accurately.

- Expansion of Private Clinic Locations: To give consumers a variety of practical options for allergy care, integrate more private clinic locations.

- Expansion of Care Options for Allergies:

To meet the demands of a wider range of users, increase the selection of allergy-related care alternatives on the site.

- Enhanced Functionalities via Novel Formats:

Constantly to improve features, user interaction, and the overall user experience, reinvent the formats available on the platform.

.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009 : The concepts and techniques for efficient system analysis and design procedures are examined in this book.
- Roger S Pressman, “*Software Engineering*”, 1994 : In the subject of software engineering, this is a foundational text that covers software development processes and procedures.
- Pankaj Jalote, “*Software engineering: a precise approach*”, 2006 :With an emphasis on accuracy, this book offers a thorough and in-depth examination of software engineering processes and principles.
- IEEE Std 1016 Recommended Practice for Software Design Descriptions :This IEEE standard offers instructions on how to write software system design descriptions. It provides a structure and format for recording a software system's design, with a focus on the documentation's clarity and thoroughness.

WEBSITES:

- www.w3schools.com
- <https://stackoverflow.com/>
- <https://www.youtube.com/brototype/>
- <https://app.diagrams.net/>

CHAPTER 9

APPENDIX

9.1 Sample Code

Views.py

Login:

```
#testing new login
```

```
def login(request):
```

```
    if request.method == 'POST':
```

```
        email = request.POST.get('email')
```

```
        password = request.POST.get('password')
```

```
    try:
```

```
        # Attempt to retrieve the user by email
```

```
        user = CustomUser.objects.get(email=email)
```

```
        # Check the password
```

```
        if user and check_password(password, user.password):
```

```
            auth_login(request, user, backend='django.contrib.auth.backends.ModelBackend')
```

```
# Specify the backend
```

```
        # Redirect based on user's role and flags
```

```
        if user.is_superuser:
```

```
            return redirect('admin') # Redirect to admin dashboard
```

```
        elif user.is_patient:
```

```
            return redirect('phome') # Redirect to patient's page
```

```
        elif user.is_doctor:
```

```
            return redirect('dhome') # Redirect to doctor's page
```

```
        else:
```

```
            error_message = "Invalid role"
```

```
            messages.error(request, error_message)
```

```
    else:
```

```
        error_message = "Invalid credentials"
```

```
        messages.error(request, error_message)
```

```
except CustomUser.DoesNotExist:
```

```
    # User with the given email does not exist
```

```
error_message = "User with this email does not exist"
messages.error(request, error_message)

response = render(request, 'login.html')
response['Cache-Control'] = 'no-store, must-revalidate'
return response
```

Signup:

#patient signup

from .models import CustomUser, Patient

```
def signup(request):
    if request.method == "POST":
        username = request.POST.get('username')
        firstname = request.POST.get('firstname')
        lastname = request.POST.get('lastname')
        dob = request.POST.get('dob')
        email = request.POST.get('email')
        password = request.POST.get('password')
        confirmPassword = request.POST.get('confirmPassword')

        if CustomUser.objects.filter(email=email).exists():
            messages.error(request, "Email already exists")
        elif CustomUser.objects.filter(username=username).exists():
            messages.error(request, "Username already exists")
        elif password != confirmPassword:
            messages.error(request, "Passwords do not match")
        else:
            # Create a CustomUser with role 'PATIENT'
            user = CustomUser.objects.create_user(
                username=username,
                first_name=firstname,
                last_name=lastname,
                dob=dob,
```

```
        email=email,
        password=password,
        is_patient=True,
        role=CustomUser.PATIENT # Assuming you have defined PATIENT as
'Patient' in your model
    )

    # Create a corresponding Patient record
    Patient.objects.create(
        user=user,
        first_name=firstname,
        last_name=lastname,
        dob=dob,
        email=email,
        username=username
    )

    messages.success(request, "Registered successfully")
    return redirect("login")

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

Signup1:

```
# new doctor signup
from .models import CustomUser, Doctor
def signup1(request):
    if request.method == "POST":
        username = request.POST.get('username')
        firstname = request.POST.get('firstname')
        lastname = request.POST.get('lastname')
        dob = request.POST.get('dob')
        email = request.POST.get('email')
        password = request.POST.get('password')
        confirmPassword = request.POST.get('confirmPassword')
```



```
# Check if the email or username already exists
if CustomUser.objects.filter(email=email).exists():
    messages.error(request, "Email already exists")
elif CustomUser.objects.filter(username=username).exists():
    messages.error(request, "Username already exists")
elif password != confirmPassword:
    messages.error(request, "Passwords do not match")
else:
    # Create a CustomUser with role 'DOCTOR'
    user = CustomUser.objects.create_user(
        username=username,
        first_name=firstname,
        last_name=lastname,
        dob=dob,
        email=email,
        password=password,
        is_doctor=True,
        role=CustomUser.DOCTOR # Assuming you have defined DOCTOR as 'Doctor'
    )

    # Create a corresponding Doctor record
    Doctor.objects.create(
        user=user,
        first_name=firstname,
        last_name=lastname,
        dob=dob,
        email=email,
        username=username
    )

    messages.success(request, "Registered successfully")
    return redirect("login")
```

```
return render(request, 'signup1.html')
```

Patient_profile:

```
# @login_required
@never_cache
def patient_profile(request):
    patient = request.user

    if request.method == 'POST':
        form = PatientProfileForm(request.POST, instance=patient)
        if form.is_valid():
            form.save()
    else:
        form = PatientProfileForm(instance=patient)

    return render(request, 'patient_profile.html', {'patient': patient, 'form': form})
```

All doctor:

```
from django.views.generic import ListView
from .models import Doctor, DoctorAdditionalDetails
```

```
class AllDoctorsListView(ListView):
    template_name = 'all_doctor_list.html'
    context_object_name = 'doctors'

    def get_queryset(self):
        # Fetch all doctors and their additional details
        doctors = Doctor.objects.all()
        details = DoctorAdditionalDetails.objects.all()

        # Combine the data into a list of tuples (doctor, additional_details)
```

```
doctor_data = []
for doctor in doctors:
    additional_details = details.filter(doctor=doctor).first()
    doctor_data.append((doctor, additional_details))

return doctor_data
```

Cart:

cart trial

```
from .models import Medicine, Cart, CartItem
```

```
@login_required
```

```
def add_to_cart(request, medicine_id):
```

```
    if request.method == 'POST':
```

```
        # Get the medicine object based on the medicine_id
```

```
        medicine = Medicine.objects.get(pk=medicine_id)
```

```
        user = request.user # Assuming the user is authenticated
```

```
        # Check if the user has a cart, create one if not
```

```
        cart, created = Cart.objects.get_or_create(user=user)
```

```
        # Check if the medicine is already in the cart
```

```
        cart_item, item_created = CartItem.objects.get_or_create(cart=cart, medicine=medicine)
```

```
        if not item_created:
```

```
            # If the item already exists in the cart, increase the quantity
```

```
            cart_item.quantity += 1
```

```
            cart_item.save()
```

```
        # Reduce the quantity of the medicine in your inventory
```

```
        medicine.quantity -= 1
```

```
        medicine.save()
```

```
    return redirect('view_cart') # Redirect to the cart view after adding the item
```

Payment:

```
from decimal import Decimal
```

```
import razorpay
```

```
def checkout(request):
```

```
    user = request.user
```

```
    cart, created = Cart.objects.get_or_create(user=user)
```

```
    amount = cart.get_total_price()
```

```
    amount_in_paise = int(amount * 100) # Convert to integer
```

```
    DATA = {
```

```
        "amount": amount_in_paise, # Use the integer value
```

```
        "currency": "INR",
```

```
        "receipt": "receipt#1",
```

```
        "notes": {
```

```
            "key1": "value3",
```

```
            "key2": "value2"
```

```
        }
```

```
    }
```

```
    client = razorpay.Client(auth=("rzp_test_aWcyAl6q9LJYqx",  
"j1dFxiB5MzxmkXTMo6IYQlnP"))
```

```
    payment = client.order.create(data=DATA)
```

```
    context = {
```

```
        'cart': cart,
```

```
        'total': cart.get_total_price(),
```

```
        'amount': amount_in_paise, # Pass the integer value to the context
```

```
        'payment': payment
```

```
    }
```

```
    return render(request, 'checkout.html', context)
```

payment_successful:

```
from django.contrib import messages
```

```
from django.http import HttpResponse
from django.shortcuts import render, redirect
from django.views.decorators.csrf import csrf_exempt
from .models import CustomUser, Patient, Order
from .models import CartItem, Cart # Assuming you have Cart and CartItem models
from reportlab.pdfgen import canvas
from decimal import Decimal
from django.db.models import Sum

@csrf_exempt
def payment_success(request):
    user = request.user
    patient = Patient.objects.get(user=user)
    cart_items = CartItem.objects.filter(cart__user=user)

    # Calculate total amount paid using aggregate
    total_amount_paid =
cart_items.aggregate(total_amount_paid=Sum('medicine__price'))['total_amount_paid'] or 0

    # Create an order after successful payment
    order = Order.objects.create(
        patient=patient,
        medicine=cart_items.first().medicine if cart_items else None,
        quantity=sum(cart_item.quantity for cart_item in cart_items),
        amount_paid=total_amount_paid,
    )

    # Clear the patient's cart after creating the order
    cart_items.delete()

    # Generate a PDF bill
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = f'attachment;
filename="bill_{order.order_date.strftime("%Y%m%d%H%M%S")}.pdf"'
```

```
# Create PDF
p = canvas.Canvas(response)
p.setFont("Times-Bold", 16)
p.drawString(100, 800, f"Bill for Order ID: {order.id}")

# Add more information to the PDF as needed
# Example:
if order.medicine:
    p.drawString(100, 780, f"Medicine: {order.medicine.name}")
p.drawString(100, 760, f"Quantity: {order.quantity}")
p.drawString(100, 740, f"Amount Paid: {order.amount_paid}")

p.showPage()
p.save()

messages.success(request, 'Payment successful! Your order has been placed.')
return response
```

Consultation:

```
#online consulting
from .forms import ConsultationRequestForm
@login_required
def submit_request(request, doctor_id):
    doctor = get_object_or_404(Doctor, id=doctor_id)
    if request.method == 'POST':
        form = ConsultationRequestForm(request.POST, request.FILES)
        if form.is_valid():
            consultation_request = form.save(commit=False)
            consultation_request.patient = request.user.patient
            consultation_request.doctor = doctor
            consultation_request.save()
```

```
        return redirect('all_doctors') # Redirect to the doctor list after submission

    else:

        form = ConsultationRequestForm()

    return render(request, 'submit_request.html', {'form': form})


from .forms import ConsultationForm
from .models import ConsultationRequest, Reply


@login_required
def doctor_consultation(request, request_id):
    consultation_request = ConsultationRequest.objects.get(pk=request_id)
    # replies = Reply.objects.filter(consultation_request=consultation_request).order_by('-
timestamp')
    replies = Reply.objects.filter(consultation_request=consultation_request)

    if request.method == 'POST':
        form = ConsultationForm(request.POST)
        if form.is_valid():
            message = form.cleaned_data['message']
            consultation_fee = form.cleaned_data['consultation_fee']
            appointment_needed = form.cleaned_data['appointment_needed']

            # Create a new Reply and save it
            reply = Reply(consultation_request=consultation_request, doctor=request.user.doctor,
message=message, consultation_fee=consultation_fee,
appointment_needed=appointment_needed)
            reply.save()

            # Redirect back to the same consultation page after submitting the reply
            return redirect('doctor_consultation', request_id)
```

```
else:
    form = ConsultationForm()

    return render(request, 'doctor_consultation.html', {'form': form, 'consultation_request':
consultation_request, 'replies': replies})
```

Appointment:

```
# appointment
from .models import Appointment, Doctor
from .forms import AppointmentForm
from django.core.mail import send_mail
from django.template.loader import render_to_string
from django.utils.html import strip_tags
from django.http import JsonResponse

@login_required
def create_appointment(request):
    if request.method == 'POST':
        # Get the doctor instance
        doctor_id = request.GET.get('doctor_id')
        doctor = get_object_or_404(Doctor, id=doctor_id)

        # Automatically populate patient name and email
        patient = request.user.patient
        form_data = request.POST.copy()
        form_data['patient_name'] = f'{patient.first_name} {patient.last_name}'
        form_data['patient_email'] = patient.email

        form = AppointmentForm(form_data)

        if form.is_valid():
```



```
appointment = form.save(commit=False)
appointment.doctor = doctor # Assign the doctor to the appointment
date = appointment.date
time_slot = appointment.time_slot

# Check if the slot is available for the selected doctor and date
slot_exists = Appointment.objects.filter(
    doctor=doctor,
    date=date,
    time_slot=time_slot
).exists()

if not slot_exists:
    # Check if the patient already has an appointment on the same day
    existing_appointment = Appointment.objects.filter(
        patient=patient,
        date=date
    ).first()

    if existing_appointment:
        response_data = {'success': False, 'message': 'You already have an
appointment on the same day.'}
    else:
        appointment.patient = patient
        appointment.save()

# Send an email to the user with the appointment details
subject = 'Appointment Confirmation'
from_email = 'your_email@example.com'
to_email = patient.email
appointment_data = {
    'appointment': appointment,
}
```

```
        html_message = render_to_string('appointment_email.html',
appointment_data)
        plain_message = strip_tags(html_message)

        send_mail(subject, plain_message, from_email, [to_email],
html_message=html_message)

        response_data = {'success': True, 'message': 'Appointment successfully
booked.'}

        return redirect('booking_success') # Redirect to the booking success page
using the URL name
    else:
        response_data = {'success': False, 'message': 'This slot is already booked. Please
choose another.'}

    # Send a toast notification
    if not response_data['success']:
        message = response_data['message']
        message_tags = 'error' # You can customize this based on your styling

        return JsonResponse(response_data, status=400)
    else:
        # If the appointment was successful, return a success message
        return JsonResponse(response_data)

else:
    form = AppointmentForm()

    return render(request, 'create_appointment.html', {'form': form})

@login_required
def booking_success(request):
    return render(request, 'booking_success.html')
```

```
@login_required
def appointment_detail(request, appointment_id):
    appointment = get_object_or_404(Appointment, id=appointment_id)
    return render(request, 'appointment_detail.html', {'appointment': appointment})
```

```
@login_required
def list_appointments(request):
    appointments = Appointment.objects.filter(patient=request.user.patient)
    return render(request, 'list_appointments.html', {'appointments': appointments})
```

Testimonial:

```
from .models import Testimonial
from .forms import TestimonialForm # Create a form for testimonials
```

```
@login_required
def submit_testimonial(request, doctor_id):
    doctor = get_object_or_404(Doctor, id=doctor_id)

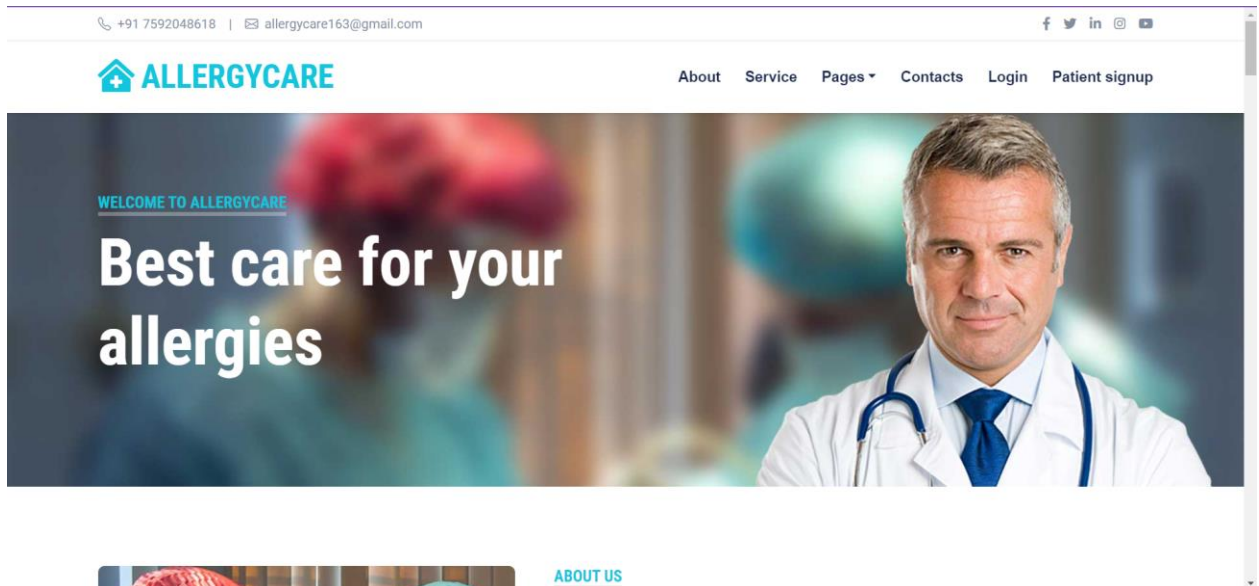
    if request.method == 'POST':
        form = TestimonialForm(request.POST)
        if form.is_valid():
            testimonial = form.save(commit=False)
            testimonial.doctor = doctor
            testimonial.patient = request.user.patient # Assuming you have a Patient model
            testimonial.save()
            messages.success(request, 'Testimonial submitted successfully!')
            return redirect('all_doctors')
        else:
            messages.error(request, 'Error submitting testimonial. Please check the form.')
    else:
        form = TestimonialForm()

    return render(request, 'submit_testimonial.html', {'form': form, 'doctor': doctor})
```

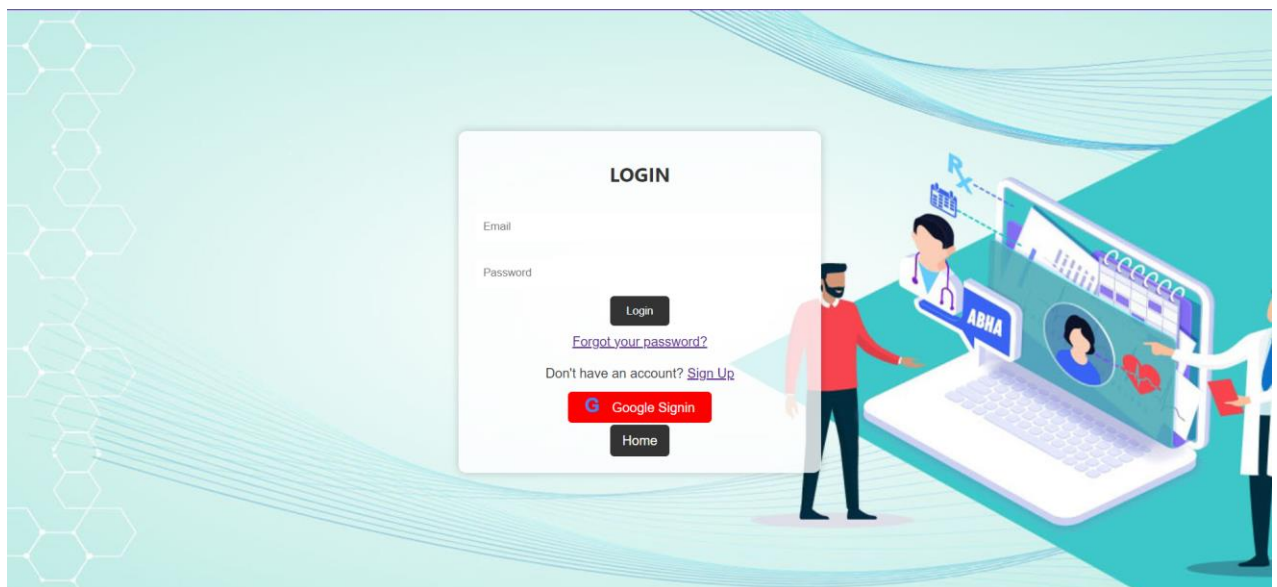
```
def view_testimonials(request):  
    testimonials = Testimonial.objects.all()  
    return render(request, 'view_testimonials.html', {'testimonials': testimonials})  
  
from .models import Doctor, Testimonial  
  
def doctor_testimonials(request, doctor_id):  
    doctor = Doctor.objects.get(pk=doctor_id)  
    testimonials = Testimonial.objects.filter(doctor=doctor)  
  
    return render(request, 'doctor_testimonials.html', {'doctor': doctor, 'testimonials':  
testimonials})
```

9.1 Screen Shots

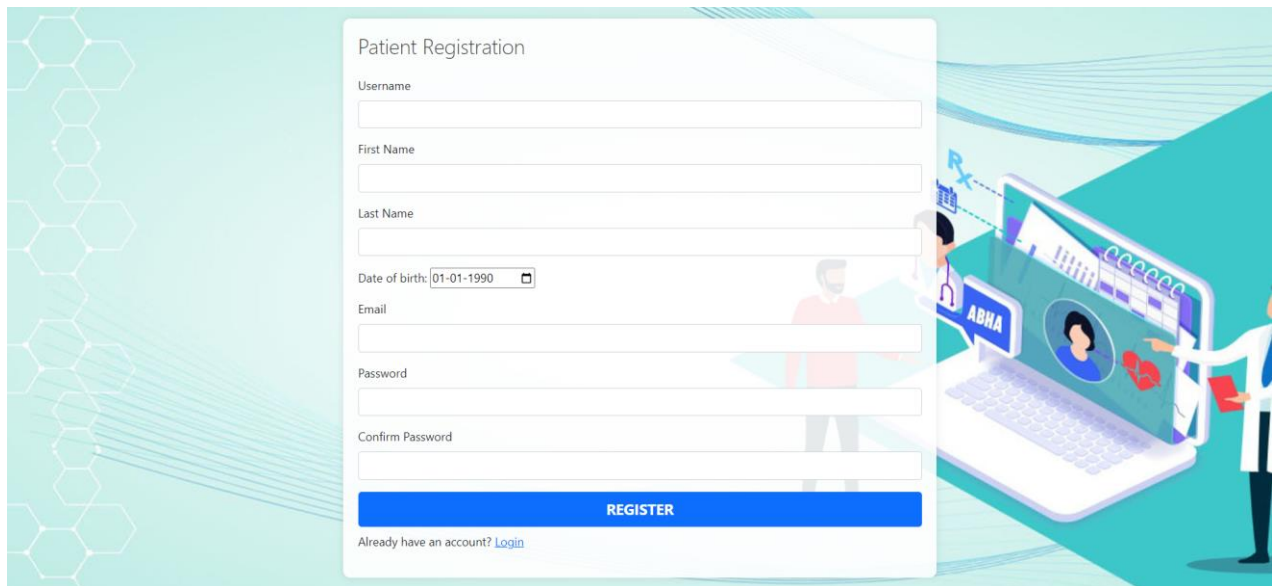
Homepage



Login page



Registration Page

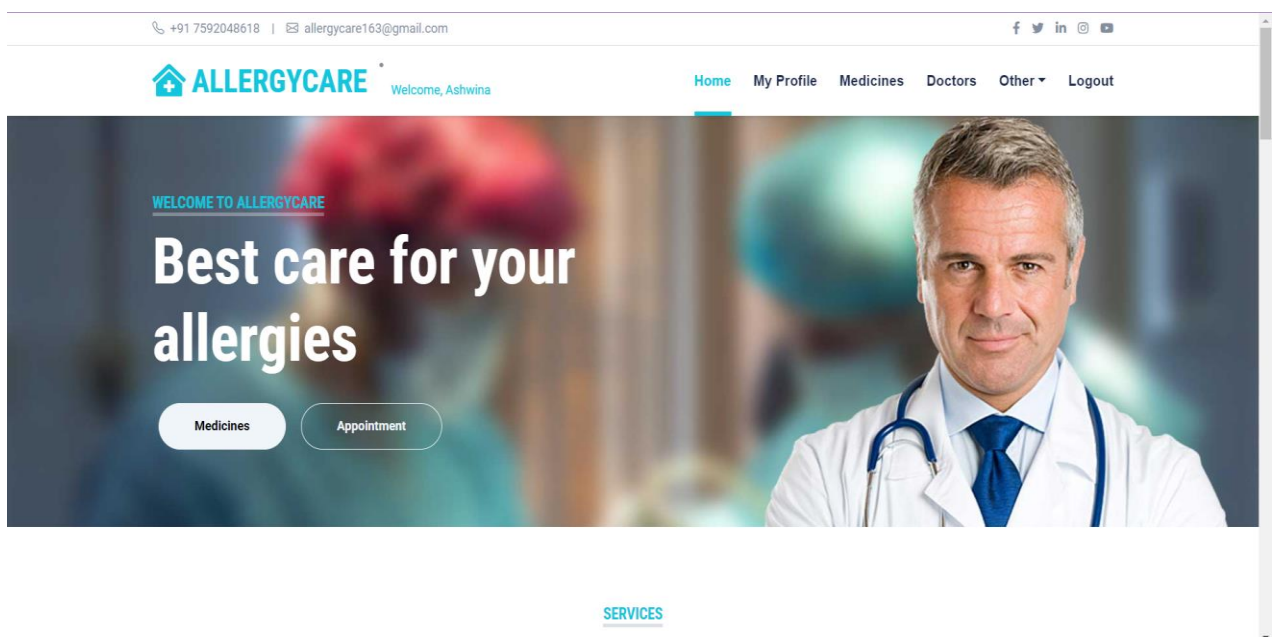


The registration form is titled "Patient Registration" and is set against a light blue background with a hexagonal pattern on the left and a medical illustration on the right. The form fields are as follows:

- Username:** A text input field.
- First Name:** A text input field.
- Last Name:** A text input field.
- Date of birth:** A date picker showing "01-01-1990".
- Email:** A text input field.
- Password:** A text input field.
- Confirm Password:** A text input field.


Below the fields is a prominent blue button labeled "REGISTER". At the bottom, there is a link: "Already have an account? [Login](#)".

Patient Home Page



The home page features a header with contact information (+91 7592048618, allergycare163@gmail.com) and social media icons. The main navigation bar includes "Home", "My Profile", "Medicines", "Doctors", "Other", and "Logout". The "ALLERGYCARE" logo is accompanied by a "Welcome, Ashwina" message. The hero section displays the text "WELCOME TO ALLERGYCARE" and "Best care for your allergies", with buttons for "Medicines" and "Appointment". A large image of a doctor is on the right. A "SERVICES" link is located at the bottom.

Patient Profile Page :



[Back](#) [My Profile](#) [Logout](#)

My Profile

Username

Ashwin

First Name

Ashwina

Last Name


S Pillai

Email

ashwinpillai230501@gm

Update Profile


Doctor List



[Back](#) [Logout](#)

List of All Doctors

Doctor Information



Name: Ashwininn Ponnappa

Email: ashwinpillai2024a@mca.ajce.in

Date of Birth: Jan. 11, 1990

ID: 40

Registration Number: 2345678

Experience: 5

Specialty: Dermatologist


Education: MBBS

Book Appointment

Submit Testimonial

Online Consult

Doctor Information



Name: Wilson John

Email: wilsonjohn@gmail.com

Date of Birth: Jan. 17, 1990

ID: 41

Registration Number: 8976453

Experience: 5

Specialty: Dermatology


Education: MBBS

Book Appointment

Submit Testimonial

Online Consult

Quiz

Back


Allergy Quiz

Do you regularly suffer from sore or watery eyes, puffy eyes, itchy eyes, runny nose, blocked nose or bouts of sneezing?

☐ Yes ☐ No

Next

Clinic Nearest Location

Back

Nearest Clinic

Nearest Clinic

The nearest clinic is **Care Hospital**

The distance to the nearest clinic is: **15.564690363472675 kilometers**

[Back to Clinic List](#)