

IrisGlow

Eye Care Hospital Management System

Project Report

Submitted by

AKKU SHAJI

Reg. No.: AJC22MCA-2009

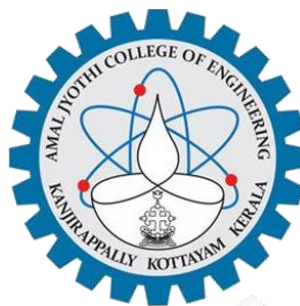
In Partial fulfillment for the Award of the Degree of

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]

2022-2024

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**IRISGLOW**” is the bona fide work of **AKKU SHAJI (Regno: AJC22MCA-2009)** 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



AMAL JYOTHI COLLEGE OF ENGINEERING

Department of Computer Applications CERTIFICATE ON PLAGIARISM CHECK

1	Name of the Scholar	Akku Shaji
2	Title of the Publication	IrisGlow
3	Name of the Guide	Ms. Lisha Varghese
4	Similar Content (%) identified	22%
5	Acceptable Maximum Limit	25%
6	Software Used	TURNITIN
8	No. of pages verified	80

*Report on plagiarism check result with % of similarity shall be attached.

Name & Signature of the Scholar: **Akku Shaji**

Name & Signature of the Guide: **Ms. Lisha Varghese**

Name & Signature of the Head of the Department: **Rev. Fr. Dr. Rubin Thottupurathu Jose**

Dept. Seal

DECLARATION

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

Date: 16/04/2024

KANJIRAPPALLY

AKKU SHAJI

Reg: AJC22MCA-2009

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 Pazhayaparambil** 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 his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my Guide **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.

AKKU SHAJI

ABSTRACT

IrisGlow stands as an innovative web-based Eye Care Hospital Management System, meticulously crafted to revolutionize the landscape of eye care services. With a primary focus on optimizing hospital efficiency and elevating patient care, this platform integrates cutting-edge technologies to offer an exceptional patient experience. Tailored exclusively for eye care facilities, IrisGlow encompasses vital functionalities like patient management, appointment scheduling, medical records, billing, and advanced eye disease self-detection, providing an all-encompassing solution for informed and seamless patient care. The system also delves into user-specific modules, allowing patients to access comprehensive information on eye health, schedule appointments, and even purchase spectacles online. The user-centric approach extends to doctors, administrators, and spectacle vendors, ensuring a holistic and enriching experience for all stakeholders. The incorporation of advanced technology for eye disease self-detection empowers patients to proactively engage in their eye health, making IrisGlow a comprehensive and user-friendly solution. Driven by a commitment to user-centric eye care, IrisGlow takes a bold step towards revolutionizing hospital operations and patient experiences.

A specialized Eye Care Hospital Management System, IrisGlow excels in functionalities such as detailed medical information, appointment scheduling, disease-specific insights, spectacles catalog and purchase, and advanced eye disease self-detection. Admins wield the power to manage user accounts, hospital settings, content, and data, while patients enjoy a seamless journey from registration and appointment booking to accessing crucial eye health information. Doctors benefit from streamlined profile and appointment management, diagnosis capabilities, and outreach participation. Spectacle vendors find a dedicated platform for catalog and purchase management. With a frontend powered by HTML, CSS, and JS, and a robust backend backed by Django and SQLite3, IrisGlow is poised to set new standards in comprehensive and accessible eye care.

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	13
3.3.1	DJANGO	13
3.3.2	SQLITE	14
4	SYSTEM DESIGN	15
4.1	INTRODUCTION	16
4.2	UML DIAGRAM	16
4.2.1	USE CASE DIAGRAM	17
4.2.2	SEQUENCE DIAGRAM	19
4.2.3	STATE CHART DIAGRAM	22
4.2.4	ACTIVITY DIAGRAM	23
4.2.5	CLASS DIAGRAM	26
4.2.6	OBJECT DIAGRAM	28
4.2.7	COMPONENT DIAGRAM	29

4.2.8	DEPLOYMENT DIAGRAM	31
4.3	USER INTERFACE DESIGN USING FIGMA	32
4.4	DATABASE DESIGN	34
4.5	TABLE DESIGN	40
5	SYSTEM TESTING	50
5.1	INTRODUCTION	51
5.2	TEST PLAN	52
5.2.1	UNIT TESTING	52
5.2.2	INTEGRATION TESTING	53
5.2.3	VALIDATION TESTING	53
5.2.4	USER ACCEPTANCE TESTING	53
5.2.5	AUTOMATION TESTING	54
5.2.6	SELENIUM TESTING	54
6	IMPLEMENTATION	65
6.1	INTRODUCTION	66
6.2	IMPLEMENTATION PROCEDURE	66
6.2.1	USER TRAINING	67
6.2.2	TRAINING ON APPLICATION SOFTWARE	67
6.2.3	SYSTEM MAINTENANCE	67
6.2.4	HOSTING	67
7	CONCLUSION & FUTURE SCOPE	70
7.1	CONCLUSION	71
7.2	FUTURE SCOPE	71
8	BIBLIOGRAPHY	72
9	APPENDIX	74
9.1	SAMPLE CODE	75
9.2	SCREEN SHOTS	86

List of Abbreviations

IDE	-	Integrated Development Environment
HTML	-	Hypertext Markup Language
CSS	-	Cascading Style Sheets
SQL	-	Structured Query Language
UML	-	Unified Modelling Language
RDBMS	-	Relational Database Management System
ORM	-	Object-Relational Mapping
MTV	-	Model-Template-View
MVC	-	Model-View-Controller
API	-	Application Programming Interface
UI	-	User Interface
BDD	-	Behaviour-Driven Development
1NF	-	First Normal Form
2NF	-	Second Normal Form
3NF	-	Third Normal Form
XSS	-	Cross-Site Scripting
CSRF	-	Cross-Site Request Forgery
CRUD	-	Create, Read, Update, Delete

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

IrisGlow emerges as an innovative web-based Eye Care Hospital Management System, strategically developed to revolutionize hospital operations and provide comprehensive eye care services. The primary objective of this system is to optimize hospital efficiency, elevate patient care, and integrate state-of-the-art technologies for an exceptional patient experience within the realm of eye care. Specifically designed for eye care hospitals, IrisGlow encompasses pivotal functionalities, including patient management, appointment scheduling, billing, and various features tailored to enhance informed patient care.

Each module within IrisGlow contributes to a specific aspect of the eye care experience. From providing detailed medical information about eye-related conditions to offering home remedies for common discomforts, the system is crafted to be a holistic resource for patients. The appointment module facilitates seamless scheduling for medical consultations, ensuring patients have convenient access to the expertise of eye care professionals. Extensive information on various eye diseases, treatments provided by the hospital, and a dedicated spectacles module further enrich the user experience.

The user-centric approach extends to different roles within the system – administrators manage user accounts and hospital settings, while patients can register, book appointments, access information. Doctors have specialized functionalities for profile and appointment management, patient consultation, and participation in outreach activities.

Administered through a user-friendly interface, IrisGlow's advanced technology empowers patients to proactively engage in their eye health. The project signifies a paradigm shift in the eye care domain, combining efficiency in hospital management with accessibility to critical eye health information, resulting in a holistic and enriching experience for both patients and medical professionals alike.

1.2 PROJECT SPECIFICATION

The proposed system, IrisGlow, is a groundbreaking online Eye Care Hospital Management System designed to optimize hospital operations and enhance patient care in the field of eye health. It offers a comprehensive range of functionalities such as patient management, appointment scheduling, medical records, billing, advanced eye disease self-detection, and a dedicated section for spectacles. The platform is specifically crafted for eye care hospitals, aiming to revolutionize their efficiency and provide patients with an exceptional and informed healthcare experience. IrisGlow utilizes

cutting-edge technologies to ensure a seamless and user-centric approach to eye care.

Admin Module:

Within IrisGlow, the Admin module serves as the central control hub, providing administrators with secure login access to manage the system. Admins have the authority to oversee user accounts, including patients, doctors, and spectacle users. They can also control and update content related to the hospital, manage hospital settings, and configure appointment scheduling. In addition, the Admin module allows the management of patient records, appointments, spectacles inventory, and chatbot interactions. This comprehensive control ensures that the hospital's operations are efficiently coordinated and optimized.

Patient Module:

The Patient module in IrisGlow is designed to offer a user-friendly experience for patients seeking eye care services. Patients can register for an account, schedule medical appointments or surgical consultations online, access information on eye diseases, and purchase spectacles. The module also facilitates eye disease self-detection through advanced image analysis techniques. Patients can interact with the chatbot for general inquiries, share testimonials, and even pledge their eyes for donation. The Patient module ensures that patients have easy access to essential eye care services and information.

Doctor Module:

In IrisGlow, the Doctor module empowers medical professionals to efficiently manage their profiles, accept or reject appointment requests, access patient medical records, and participate in hospital outreach activities. The module also enables doctors to assist patients through the chatbot for non-emergency queries, ensuring a holistic approach to patient care. With profile management and appointment handling functionalities, the Doctor module contributes to the seamless operation of the hospital's medical services.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

A critical step in system development is system analysis, which involves gathering and analyzing data to identify problems and provide solutions. During this phase, effective communication between system users and developers is essential. Any system development process should always begin with a system analysis. The system analyst performs the role of an investigator by carefully examining the current system's performance. Identification of the system's inputs and outputs as well as a connection between its processes and organizational results are all included. A range of techniques, including surveys and interviews, are used to acquire information. Understanding the system's operation, locating problem areas, and suggesting fixes to the issues the business is having are the objectives. The designer assumes the role of issue solver, and the suggested fixes are rigorously compared with the current system. The user is given the chance to accept or reject the advice after the best choice has been chosen. The procedure continues until the user is pleased after the idea has been evaluated in light of their comments. The process of acquiring and analyzing data for future system studies is called a preliminary study. Conducting a thorough preliminary study is essential to ensure the success of a system development project.

2.2 EXISTING SYSTEM

The existing system of eye care management relies on traditional, manual processes for record-keeping and appointment scheduling. These methods often lack the efficiency and accessibility needed for optimal patient care. Patients typically visit hospitals physically, limiting the flexibility of modern healthcare solutions. Unlike the innovative capabilities of IrisGlow, existing systems may face challenges in incorporating advanced technologies and providing a holistic patient experience. The IrisGlow project aims to revolutionize and overcome these limitations in traditional eye care management.

2.2.1 NATURAL SYSTEM STUDIED

The current natural system in eye care hospitals relies on manual communication methods like phone calls and emails for appointment scheduling. Patients initiate contact, inquire about availability, and coordinate appointments, often resulting in prolonged exchanges. Recognizing the limitations of this manual process, there's a need for a streamlined, technologically advanced system to optimize appointment scheduling within eye care hospitals. This study focuses on addressing these challenges and implementing an innovative solution for efficient and accessible eye care hospital management.

2.2.2 DESIGNED SYSTEM STUDIED

In response to the inefficiencies of manual processes in eye care hospital management, a modern Eye Care Appointment Management System is proposed. This system seeks to transform the way patients and eye care professionals interact by offering a streamlined, user-friendly, and efficient solution. Key features include a digital platform for patients to access eye care professionals' profiles, book appointments, and receive real-time notifications, reducing reliance on time-consuming phone calls and emails. The system's design prioritizes convenience, transparency, and the overall patient experience, addressing the limitations of the current system and harnessing the advantages of digital technology in eye care hospital management.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Inefficiency:** The manual process can be highly inefficient, often involving numerous rounds of communication between patients and doctors to secure a suitable appointment time.
- **Limited Access:** This system may limit access to services to those who are proficient in navigating the process, potentially leaving less tech-savvy individuals at a disadvantage.
- **Lack of Transparency:** Clients may not have access to comprehensive information about doctors, making it challenging to make informed choices.
- **Missed Appointments:** Without automated reminders and notifications, there is a higher likelihood of missed appointments, leading to frustration for both patients and doctors.
- **Security Concerns:** Sharing personal information over email or phone can raise security concerns, particularly when discussing sensitive therapy topics.
- **Limited Availability:** The system's reliance on manual processes can create scheduling challenges for patients with busy schedules or those requiring treatment outside of standard working hours.
- **Inconvenience:** In-person visits can be inconvenient, especially when patients and doctors are geographically distant, necessitating multiple trips for meetings.

2.4 PROPOSED SYSTEM

The envisioned technique introduces a diverse array of eye care options, allowing individuals to select the type of assistance that aligns with their specific needs. The platform ensures continuous accessibility for users, enabling them to seek assistance whenever required. Through the website, users can effortlessly schedule appointments with nearby qualified eye care professionals, emphasizing accessibility and convenience. The system incorporates machine learning technologies find eye diseases our own to offer comprehensive eye care. Its overarching goal is holistic recovery

and the enhancement of overall eye health. Additional modules encompass feedback systems, a chatbot, all aimed at streamlining the eye care experience, empowering patients, and enhancing the accessibility of eye care services.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- **Efficiency:** Streamlined processes reduce the time and effort required for both patients and hospital staff.
- **Convenience:** Patients can access eye care services from the comfort of their homes, eliminating the need for physical visits.
- **Transparency:** Patients have access to detailed information about eye care services, medical team expertise, and treatment options, enabling informed decisions.
- **Secure Payments:** Payment integration ensures secure financial transactions for eye care services and spectacles purchases.
- **Empowering Users:** The system provides a bridge between eye care education and practice, offering opportunities for interns and medical students.
- **Enhanced Scheduling:** Integration of appointment schedules allows patients to efficiently align their visitations with the availability of eye care professionals.
- **Personalized Eye Health Guidance:** The Deep Learning-driven Cataract Disease Self-Detection feature provides personalized guidance, aiding in early detection and management of cataract based on individual eye health needs and goals.
- **Chatbot for Personal Assistance:** The AI-driven chatbot feature provides personal assistance.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

This Feasibility, in the context of the IrisGlow Eye Care Hospital Management System, is defined as the practical extent to which the project can be successfully executed. To assess this feasibility, a comprehensive study has been conducted. This study aims to determine the practicality and workability of the proposed software solution to meet the specified requirements effectively. Various factors, including resource availability, software development cost estimation, post-development organizational benefits, and maintenance expenses, have been meticulously examined during this feasibility study. The culmination of this study is a report that provides recommendations on whether proceeding with the requirements engineering and system development process is justified. The primary objective of this feasibility study is to establish the rationale for developing the IrisGlow system. It should be not only acceptable to users but also adaptable to change and compliant with established standards. Several other key objectives of this study are outlined below:

- 1. Meeting Organizational Requirements:** To analyze whether the IrisGlow software aligns with the organizational requirements for efficient hospital management and enhanced patient care.
- 2. Technological Feasibility:** To determine whether the software can be successfully implemented using current technology resources and within the specified budget and timeline.
- 3. Integration Capability:** To assess whether the software can be seamlessly integrated with other existing healthcare software systems.

By harnessing available technologies strategically, the system streamlines operations, reducing manual paperwork, saving time, and increasing accuracy. IrisGlow's potential to enhance patient care quality and operational efficiency directly benefits its users while reinforcing its viability as a cutting-edge solution in the realm of eye care hospital management. The feasibility study for IrisGlow affirms its potential to revolutionize eye care services, making it a practical and worthy endeavor for further development and implementation.

3.1.1 Economical Feasibility

IrisGlow demonstrates economic feasibility through its potential revenue streams, cost effective implementation, and anticipated return on investment (ROI). The platform has the potential to generate revenue by offering services such as appointment bookings, medical record management, and spectacles sales, providing a diversified income stream. Moreover, IrisGlow's efficient use of

available technologies and resource optimization ensures cost-effectiveness throughout the development, maintenance, and operational phases. By conducting a comprehensive cost-benefit analysis that considers development costs, hardware and software expenses, maintenance, staffing, and operational costs, IrisGlow can confidently assess its financial viability. Taking into account the potential revenue streams and well-managed expenses, IrisGlow is well-positioned to calculate its expected ROI within a reasonable timeframe. This calculation further solidifies the platform's economic feasibility, demonstrating its potential as a financially sustainable solution for enhancing eye care services and improving overall hospital management efficiency.

3.1.2 Technical Feasibility

IrisGlow exhibits robust technical feasibility through a strategic utilization of available technologies, a commitment to scalability, and a seamless integration approach. The platform optimizes the utilization of existing technologies to ensure efficient performance, negating the necessity for cloud computing. With a strong focus on user-friendliness and cross platform compatibility, IrisGlow enhances accessibility for eye care professionals, patients, and administrators, enabling effortless access and utilization across various devices. The project benefits from a dedicated development team, equipped with the expertise and skills required for a streamlined development process and continuous maintenance. Thoughtful integration of advanced technologies and rigorous security measures ensures a secure and dependable environment for users. IrisGlow is dedicated to providing a comprehensive and accessible eye care management experience, and its technical foundation is well-prepared to support these objectives effectively.

3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

- Is there adequate user support?
- Will anyone be harmed by the proposed system?

Because it would accomplish the objectives after being developed and put into action, the project would be advantageous. After carefully assessing all behavioral considerations, it is determined that the project is behaviorally feasible. Users can simply use IrisGlow GUI without any training because it is user-friendly.

3.1.4 Feasibility Study Questionnaire

A. Project Overview

IrisGlow is a web-based platform designed to enhance users' eye care experiences. It offers a wide range of services, including appointment scheduling, disease information, spectacles purchase, and more. The platform's primary objective is to optimize eye care hospital operations and improve patient care by integrating cutting-edge technologies. IrisGlow caters specifically to eye care hospitals and individuals seeking specialized eye care services, promoting holistic eye health and well-being.

B. To what extend the system is proposed for?

The system's purpose is to streamline eye care hospital management and improve patient experiences. It caters to both medical and surgical patients, offering appointment scheduling, disease information, spectacles purchase, and more.

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

- Patients seeking medical or surgical eye care.
- Eye doctors and specialists.
- Administrators managing the hospital's operations.
- Spectacles users looking to purchase eyewear.

D. List the Modules included in your System?

- Registration
- Eyecare Module
- Appointment Module
- Diseases Module
- Spectacles Module

E. Identify the users in your project?

- Admin
- Patients
- Doctors
- Spects

F. Who owns the system?

Administrator

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

IrisGlow is related to the healthcare and eye care industry, aiming to enhance services in eye hospitals.

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

- Dr. Dhrumil C K (Eye Specialist)
- Arvind Eye Care System (Eye Care Hospital Website)

I. Questionnaire to collect details about the project?

1. **What are the common eye care services offered at your hospital?**
 - Answer: We provide a wide range of services, including medical treatment, surgery, and home remedies.
2. **How do you handle appointment scheduling for medical and surgical patients?**
 - Answer: We have a manual appointment booking system currently.
3. **What challenges do you face in managing eye disease information and treatment data?**
 - Answer: Maintaining accurate records and providing information to patients efficiently can be challenging.
4. **What additional features would you like to see in the system to support eye care services effectively?**
 - Answer: Integration of virtual eye disease diagnosis tools would be valuable.
5. **How can the system improve the patient experience when purchasing spectacles online?**
 - Answer: Offering a virtual try-on feature would enhance the experience.
6. **What challenges do you anticipate in transitioning from manual appointment scheduling to an online system?**
 - Answer: Ensuring a smooth transition for both staff and patients might require training and support.
7. **How can the system ensure secure and efficient handling of insurance-related information for eye care treatments?**
 - Answer: Implementing secure data encryption and automated claims processing would be essential.

8. What features or tools do you believe would enhance the overall eye care experience for patients?

- Answer: Features like online medical records access and interactive educational resources would be valuable.

9. What are your thoughts on incorporating virtual consultations for eye care services?

- Answer: Virtual consultations could improve accessibility for patients, especially for follow-up appointments.

10. How can the system contribute to improving the outreach activities and community engagement of the hospital?

- Answer: Implementing event registration and informative content sharing features would be beneficial.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel i3

RAM - 4 G B

Hard disk - 2 5 6 G B

3.2.2 Software Specification

Front End - HTML, CSS, JS, Bootstrap

Backend - Python

Database - SQLite

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, J Query, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

Django, a leading open-source web framework, epitomizes the pinnacle of modern web development with its efficiency and versatility. Built on Python, Django follows the Model-View-Controller architecture, prioritizing simplicity and flexibility. Noteworthy features include its

Object-Relational Mapping system, streamlined URL routing, and a user-friendly template engine. The built-in admin interface simplifies data management, while robust security measures and middleware support enhance application integrity. Django scales effortlessly, accommodating growing datasets and traffic demands. Its REST Framework extension further extends its utility to API development. Supported by an active community and extensive documentation, Django stands as a powerful toolkit, seamlessly shaping the development of dynamic and secure web applications for diverse purposes, from content management systems to Restful API. In essence, Django empowers developers with a comprehensive and adaptable framework for crafting sophisticated and salable web solutions.

3.3.2 SQLITE

SQLite is a lightweight, self-contained, and serverless relational database management system. Unlike traditional databases, it doesn't require a separate server but is embedded directly within the application. One of its standout features is the self-contained nature of SQLite databases; the entire database is stored in a single file, simplifying management, backups, and transfers. Despite its lightweight design, SQLite ensures data integrity by supporting transactions, making it suitable for multi-user environments.

SQLite is cross-platform and compatible with various operating systems, making it a versatile choice for applications targeting different platforms. Its small code size and minimal resource usage make it suitable for resource-constrained environments, such as mobile devices. SQLite is known for its speed and efficiency, especially in read-heavy workloads. It is commonly used in mobile applications, desktop applications, and embedded systems, where a full-fledged database server might be excessive, and portability is essential.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The development process of any intended system or product begins with design. Creative process is planning the secret to an effective system is proper planning. The process uses different methods and concepts to define a process or system in sufficient detail to enable it the physical implementation is called the "design". One way to describe it is as a work process various methods and concepts to define a device, process or system in sufficient detail enable its physical realization. Regardless of the development paradigm used, software engineering forms the technical core of software engineering. Architectural the detail required to build a system or product is developed through system design. This the program also refined all the efficiency, performance, and levels of detail, as with any systematic technique. A user-oriented document is converted into a document for developers or database workers during the design phase. The two phases of system design are logical design and physical design.

4.2 UML DIAGRAM

A standardized language known as UML used for design purposes, The development, visualization and description of software systems are created by Object management group (OMG). The initial draft of the UML 1.0 standard was published in January 1997 and since then it has been widely used as a visual language for description and development. software systems. Unlike programming languages such as Java, C and COBOL, UML is used instead of diagrams to describe the flow and interaction of system components writing code Although UML is most often used to design software systems, it can also be applied to non-software systems such as the process flow of a manufacturing facility. With use UML diagrams in easy-to-use tools can be translated into code for multiple computers Language (languages. Thanks to successful standardization, OMG recognized UML as a standard and a close relationship with object system analysis and design.

The nine different UML diagrams are:

- Class Diagram
- Object Diagram
- Use Case Diagram
- Sequence Diagram
- Activity Diagram
- State chart Diagram
- Deployment Diagram
- Component Diagram

4.2.1 Use Case Diagram

A use case diagram is a graphical representation that illustrates the relationships between the functions of various system components. It is a method of defining, outlining and organizing system requirements, such as the need for a website to order goods and services. Use case diagrams are one of the tools of the popular modeling language UML, which describes the structure and operation of systems and objects seen in the real world. Tasks which constitute the objectives of the system can range from defining general needs to acceptance hardware design, testing and troubleshooting software currently in development; create online help resources and perform tasks related to customer service. For example, in the context of a business scenario, use cases might include responding to a customer service request, product order processing, product listing maintenance and tracking payment processing. Typically, a use case diagram has four important parts

- A wall that separates the system being studied from its environment.
- Actors are often participants in the system, identified by their role.
- Actors inside and outside the system implement use cases, which are expert roles.
- Discussions and interactions between participants and use cases



Figure 1: Use Case Diagram of Admin, Patient, Doctor and Specs

4.2.2 Sequence Diagram

A sequence diagram is a powerful visual tool that effectively portrays the interaction between various objects or components in a precise sequence or order. This diagram is also commonly known as an event diagram or event scenario. The primary objective of a sequence diagram is to provide a clear and detailed representation of how different objects within a system collaborate and execute their functions in a specific order. Software developers, as well as business professionals, often leverage sequence diagrams to document, analyze, and comprehend the intricate dynamics and operational requirements of both newly designed systems and pre-existing ones. These diagrams play a vital role in enhancing communication, facilitating system design, and ensuring a comprehensive understanding of system behavior and interaction.

Sequence diagrams in UML are a powerful tool for visualizing the interactions between objects in a system. To create effective sequence diagrams, it's essential to understand the key notations and elements used:

1. **Actors:** Actors represent roles or external entities that interact with the system being modeled. These can include human users, other systems, or any external subjects. Actors are always positioned outside the system's scope in the UML diagram. In sequence diagrams, actors are depicted using a stick person notation. Multiple actors can be present in a single sequence diagram, each representing a distinct role or entity.
2. **Lifelines:** Lifelines are named elements that signify individual participants in the sequence diagram. Each instance involved in the interaction is represented by a lifeline. Lifelines are typically located at the top of the sequence diagram, and they serve as a visual reference for the objects or participants in the interaction.
3. **Messages:** Messages in a sequence diagram represent communication between objects. Messages are presented in a sequential order on the lifelines, demonstrating the flow of interactions. Messages are typically represented using arrow notations. They are at the core of a sequence diagram and play a crucial role in illustrating how objects collaborate. Messages can fall into several categories, including:
 - **Synchronous messages:** These messages denote interactions where the sender waits for a response from the receiver.
 - **Asynchronous messages:** These messages indicate interactions where the sender does not wait for an immediate response.
 - **Create message:** Used to illustrate the creation of a new object.

- **Delete message:** Indicates the destruction or deletion of an object.
 - **Self-Message:** Represents interactions within the same object.
 - **Reply Message:** Depicts responses or replies to messages.
 - **Found Message:** Used to show a message found later during the interaction.
 - **Lost Message:** Demonstrates a message that was lost during the interaction.
4. **Guards:** Guards are used in UML sequence diagrams to model conditions or constraints. They come into play when there's a need to control the flow of messages based on specific conditions. Guards are crucial in conveying any constraints or rules governing the system or a particular process. They provide valuable information to software developers about the conditions that must be met for certain interactions to occur.

Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.

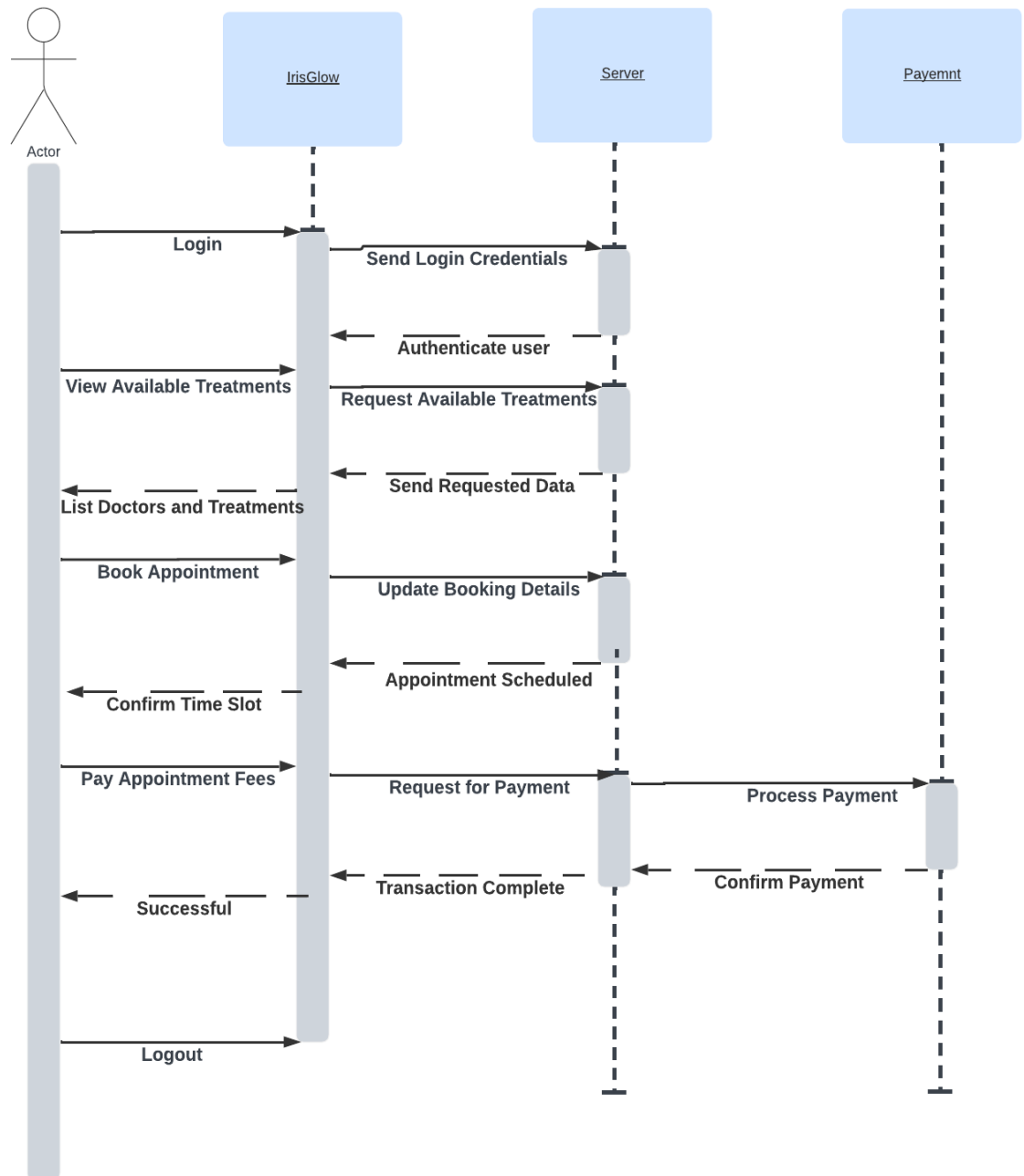


Figure 2: Sequence Diagram of IrisGlow

4.2.3 State Chart Diagram

A state machine diagram, also known as a state diagram or state transition diagram, is a visual representation that shows the sequence of states that an object passes through in a system. It provides a comprehensive view of the behavioral aspects of a software system and can be applied to model the behavior of a class, subsystem, package, or the entire system itself. State machine diagrams are particularly effective in modeling the interaction and cooperation between external entities and the system, especially in event-driven systems where object state plays a critical role. These diagrams define different states for a system component, and each object or component is associated with a specific state.

Notations of a State Machine Diagram:

The notations used in a State Machine Diagram include the following elements:

- **Initial State:** This element represents the starting point of the system's state machine and is depicted by a black filled circle.
- **Final State:** The final state signifies the endpoint of the system's state machine and is illustrated by a filled circle enclosed within another circle.
- **Decision Box:** This diamond-shaped element symbolizes decision points in the state machine, where choices are made based on evaluated guards or conditions.
- **Transition:** Transitions in a state machine diagram signify a change of control from one state to another due to the occurrence of a specific event. Transitions are represented by arrows labeled with the event that triggers the change.
- **State Box:** The state box represents the conditions or circumstances of a particular object or component within a class at a specific moment in time. It is depicted as a rectangle with rounded corners.

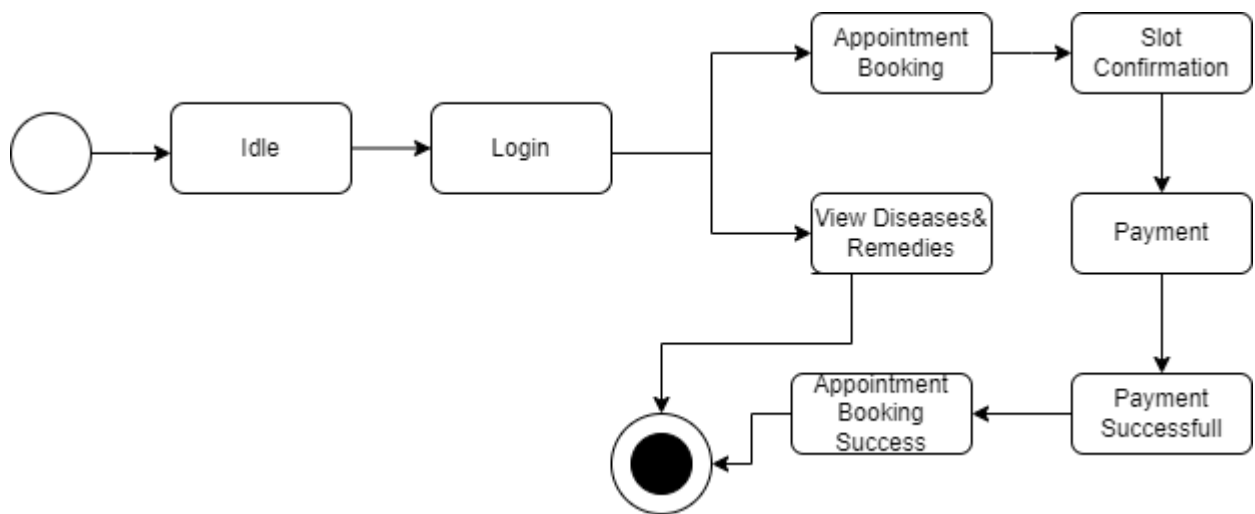


Figure 3: State Chart Diagram of IrisGlow

4.2.4 Activity Diagram

An Activity Diagram in UML serves as a visual tool to depict the flow of control within a system, with a focus on the workflow between activities rather than the detailed implementation. It is especially useful for illustrating the sequence of activities, both concurrent and sequential, and emphasizes the order and conditions that dictate the flow. Activity Diagrams are often likened to object-oriented flowcharts and are composed of a series of actions or operations to model the behavioral aspects of a system.

Components of an Activity Diagram:

- Initial Node:** This node marks the starting point of the activity diagram, indicating where the process begins.
- Activity States:** Activity states represent the various activities involved in a process and are typically depicted by rounded rectangles.
- Decision Nodes:** Decision nodes serve as points in the activity diagram where the flow can diverge into different paths based on specific conditions.
- Fork Nodes:** Fork nodes indicate parallel processing of activities, allowing multiple actions to occur concurrently.
- Join Nodes:** Join nodes are used to merge parallel processing flows back into a single flow, consolidating the outcomes of parallel activities.
- Final Nodes:** Final nodes signal the conclusion of an activity diagram, marking the endpoint of the control and object flows.
- Control Flows:** Control flows are represented by arrows connecting various nodes and depict

the order in which activities are performed.

- h) **Action States:** Action states represent specific actions or operations performed during an activity and are represented by rectangles with rounded edges.
- i) **Swimlanes:** Swimlanes are used to group activities according to their responsible parties or departments, simplifying the visualization of the flow of work.

Notation of an Activity Diagram:

The notation used in an Activity Diagram includes the following elements:

1. **Initial State:** This denotes the initial stage or starting point of a sequence of actions.
2. **Final State:** The final state signifies the point where all control flows and object flows reach their conclusion.
3. **Decision Box:** Decision boxes ensure that the control or object flow follows only one path based on evaluated conditions.
4. **Action Box:** Action boxes represent a set of actions that need to be performed during the activity and are typically depicted as rectangles.

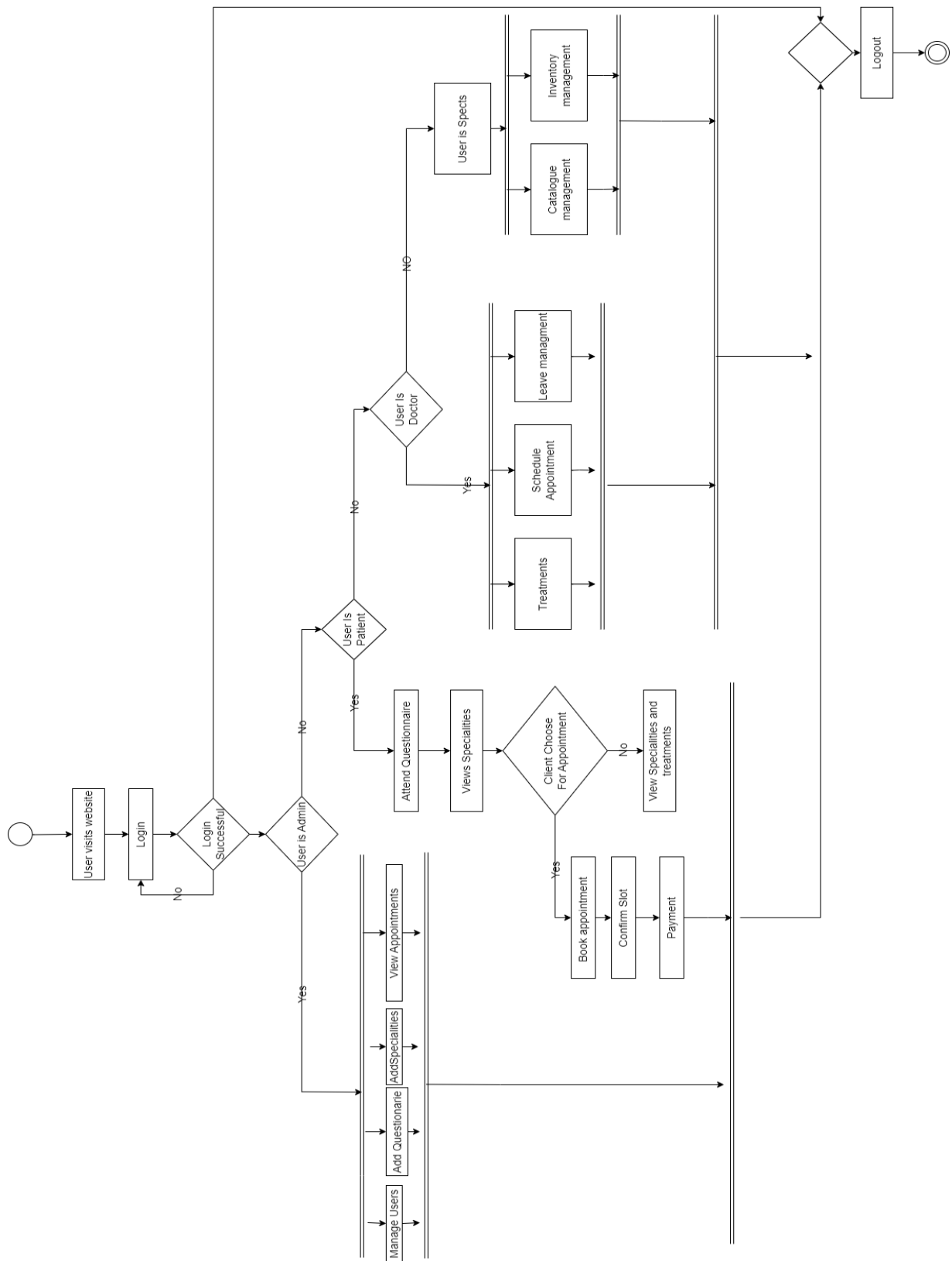


Figure 4: Activity Diagram of IrisGlow

4.2.5 Class Diagram

The static view of the program is shown in the class diagram. The many product types of the system contains and possible connections between them. Despite the fact that the class can too derived from other classes, a class is only as good as its objects. Class diagrams are used to describe, describe and document the various parts of the system. They are also used to create executable software code. A block diagram gives a general picture of the architecture of a software system by showing system classes, relationships, properties and functions. to support by creating programs for a specific domain, it helps to gather information about the class names, properties and characteristics. A Class Diagram in UML is a visual representation of the structure and organization of classes in a system, emphasizing their attributes, operations, and relationships. It consists of three main sections:

1. **Upper Section:** This section encompasses the name of the class, which represents a category of objects sharing the same attributes, operations, relationships, and semantics. When representing a class, certain rules should be followed:
 - Capitalize the initial letter of the class name.
 - Place the class name in the center of the upper section.
 - The class name must be written in bold format.
 - If the class is abstract, its name should be written in italics format.
2. **Middle Section:** The middle section of the class diagram deals with the attributes of the class. Attributes describe the qualities or characteristics of the class. Attributes in a class diagram have specific characteristics, such as:
 - Attributes are written along with their visibility factors, which can be public(+), private(-), protected (#), or package (~).
 - Visibility factors illustrate the accessibility of an attribute within the class.
 - Attributes should have meaningful names that explain their usage within the class.
3. **Lower Section:** The lower section of the class diagram contains methods or operations that define how the class interacts with data or performs specific actions. Methods are represented as a list, with each method written on a single line.

Relationships:

In UML, relationships between classes are classified into three main types:

- **Dependency:** A dependency is a semantic relationship between two or more classes where a change in one class may result in changes in another class. It represents a weaker relationship that indicates one class relies on another class, but not necessarily in an inheritance or association manner.
- **Generalization:** Generalization is a relationship between a parent class (superclass) and a child class (subclass). In this relationship, the child class inherits attributes, operations, and characteristics from the parent class. It signifies an "is-a" relationship where the child class is a specialized version of the parent class.
- **Association:** Association represents a static or physical connection between two or more objects, describing how many objects are involved in the relationship and their roles within that relationship. Associations can also include multiplicities to specify the number of instances participating in the relationship.

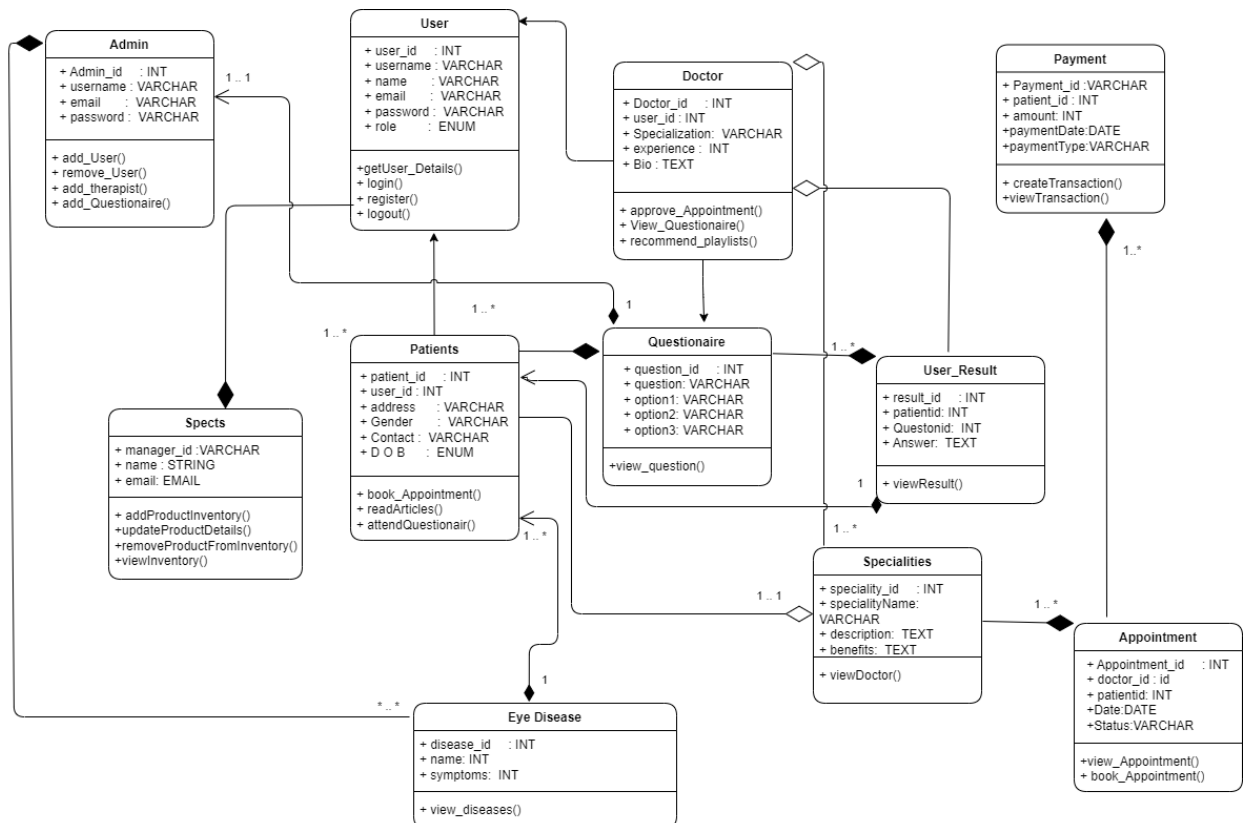


Figure 5: Class Diagram of IrisGlow

4.2.6 Object Diagram

Object diagrams are based on the class diagram because they are based on the class diagram. They describe class diagram representation by object representation. Objects provide a static representation of the object system at a given moment. Object and class diagrams are similar in a way, but the class diagram provides a conceptual overview of the system and provides a visual representation of a specific function of the system

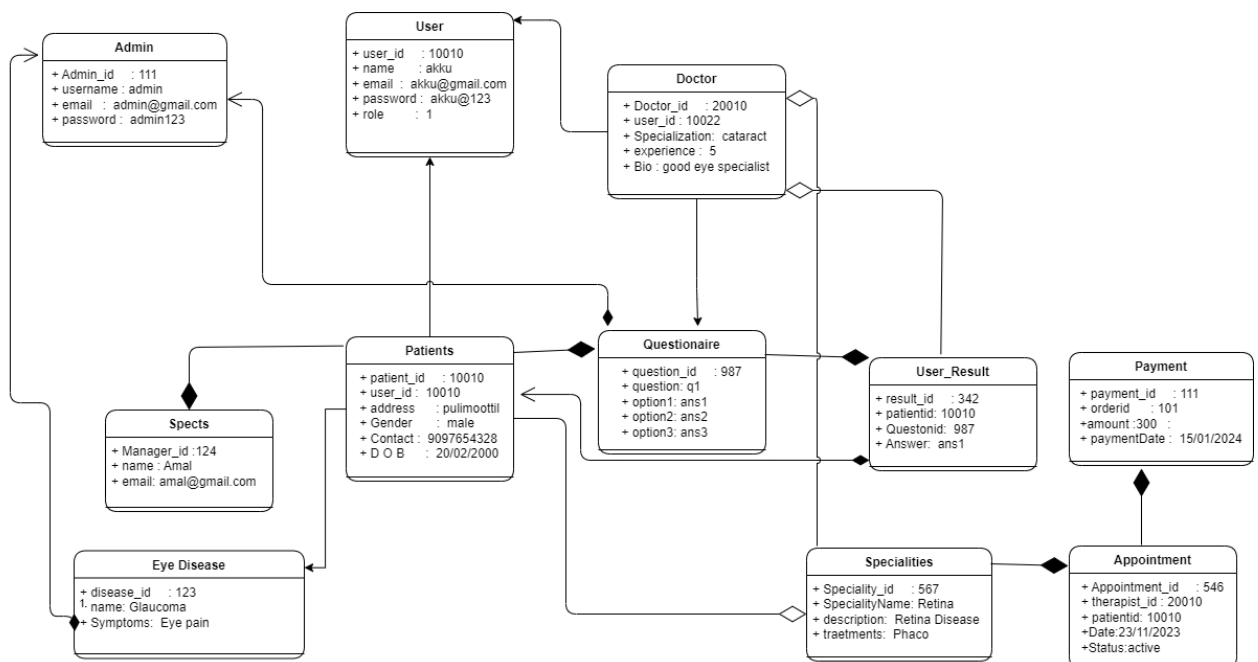


Figure 6: Object Diagram of IrisGlow

4.2.7 Component Diagram

There are smaller components to make a complex object system more understandable separated using a component diagram. It simulates the physical view of the system, including this internal node executables, files, libraries, etc. It describes existing connections and hierarchies between system components. It helps to create a working system. individual, replaceable and executable system unit is called a component. Component application information is hidden, so a user interface is required to perform the function. This acts as a "black box" where the provided and required interfaces explain its behavior.

Notation of a Component Diagram

1. A component
2. A node

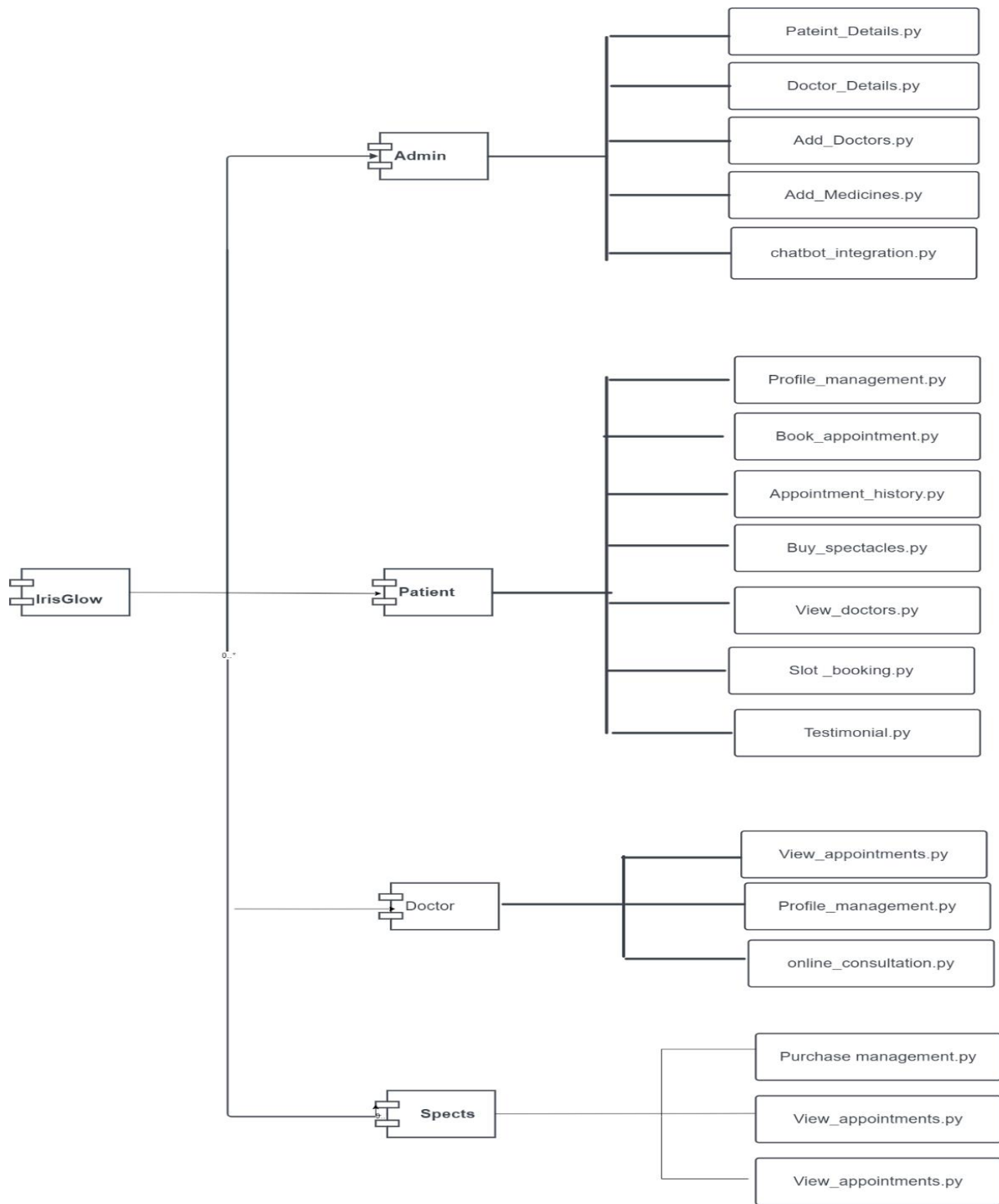


Figure 7: Component Diagram of IrisGlow

4.2.8 Deployment Diagram

An deployment diagram is a type of UML diagram that shows the physical hardware or system the architecture in which applications are deployed. This gives a static view system implementation and includes nodes and their relationships. The diagram shows how software components are distributed and deployed on physical hardware or nodes. Diagram is useful for mapping a software architecture designed to a physical system architecture where the software is launched. Communication channels are used to show relationships between people

Notation of Deployment diagram

The deployment diagram consists of the following notations:

1. A component
2. An artifact
3. An interface
4. A nod

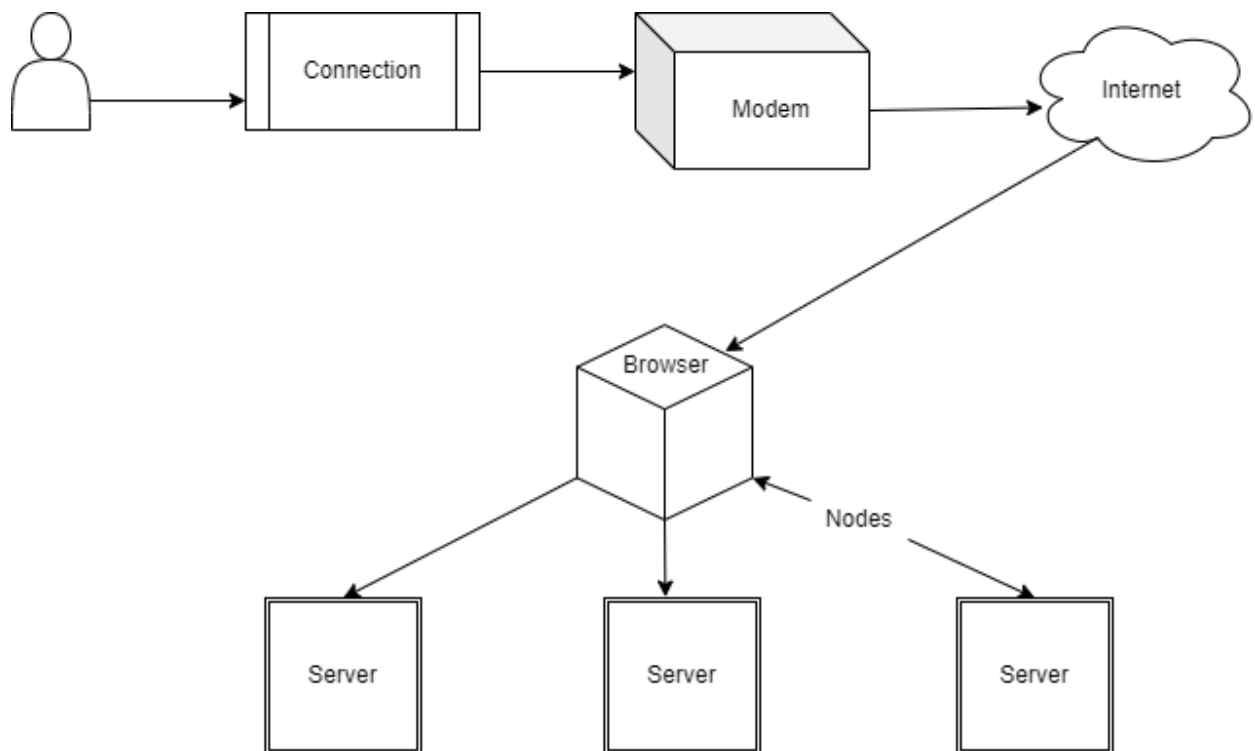
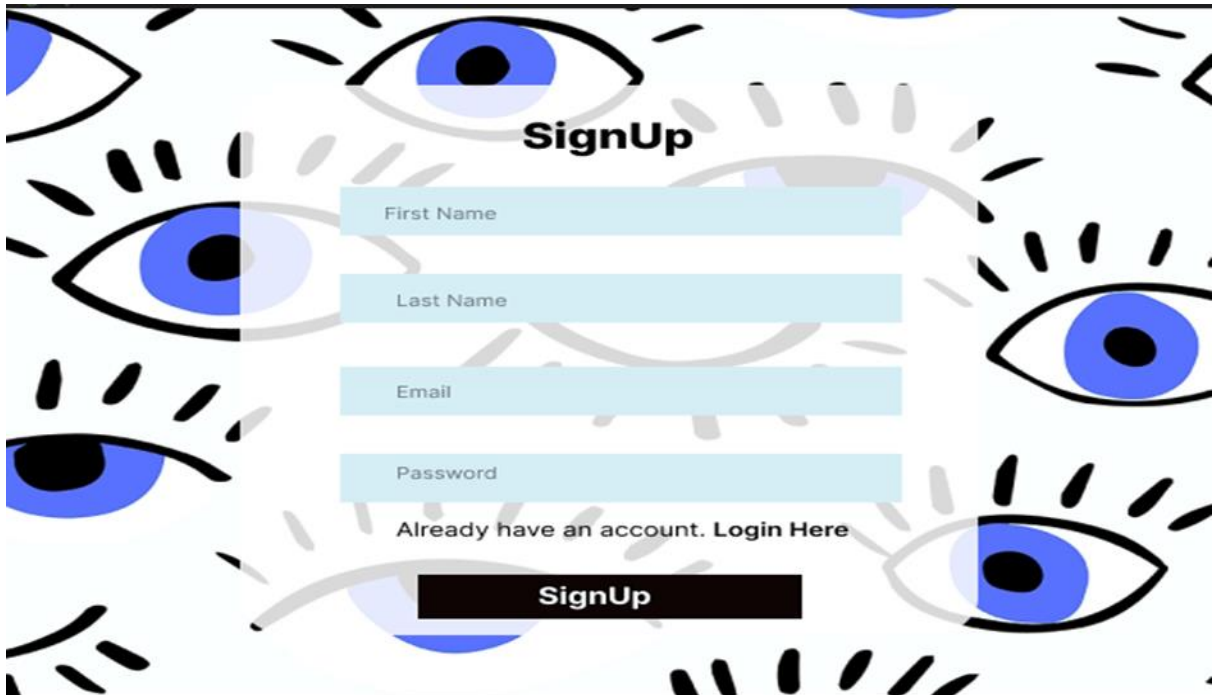


Figure 8: Deployment Diagram of IrisGlow

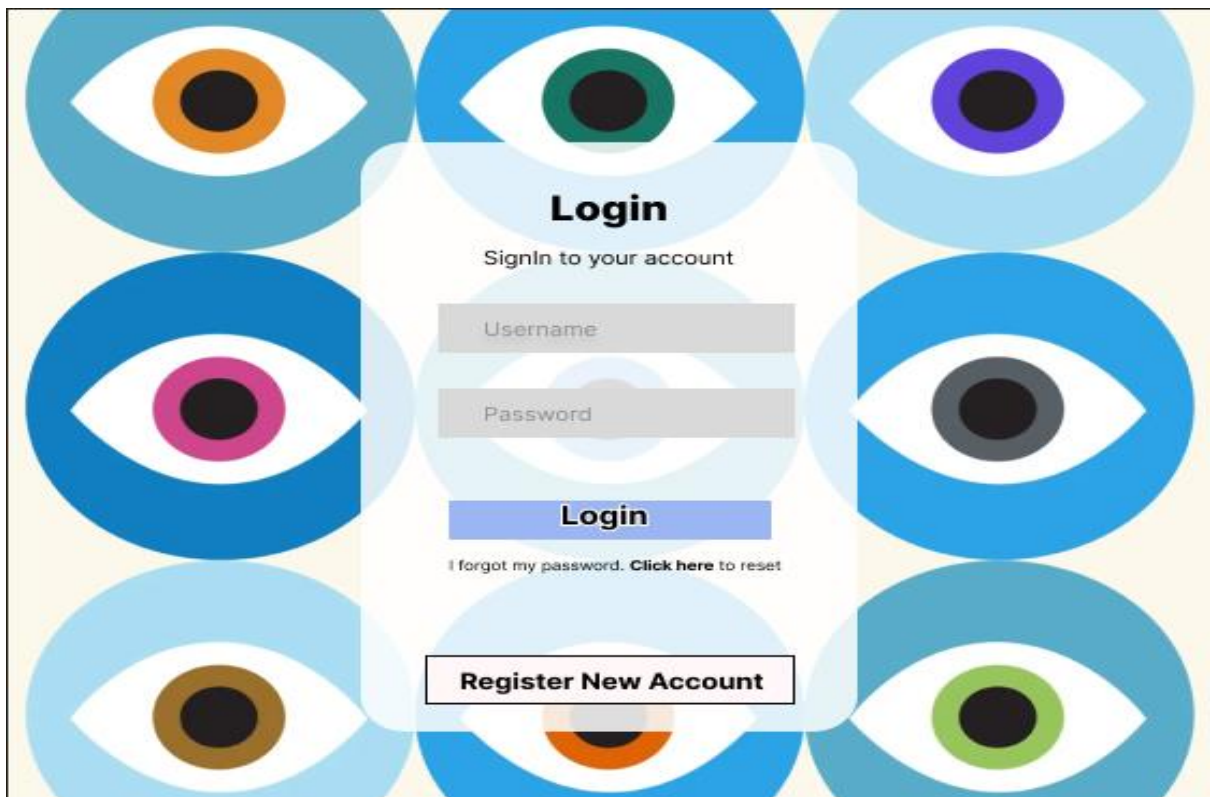
4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: User Registration



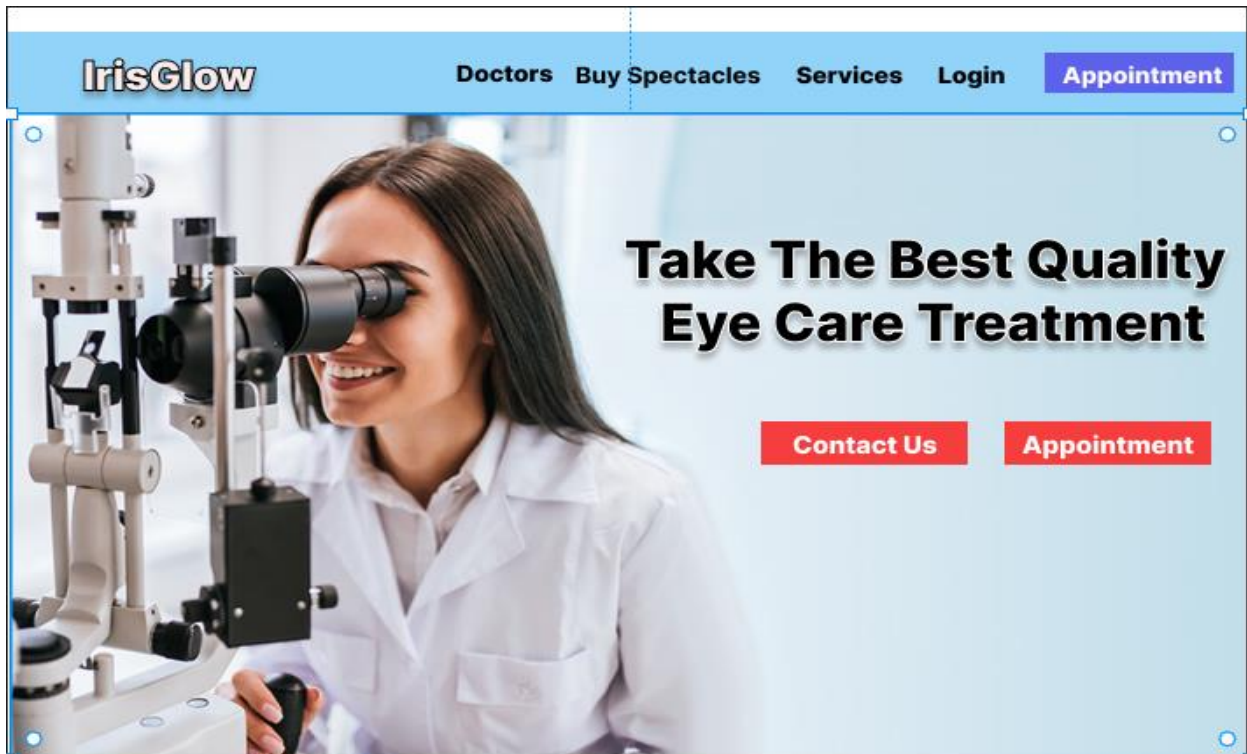
The User Registration form is centered on a light blue background with a pattern of stylized eyes. The form includes a title "SignUp" in bold black text. Below the title are four light blue input fields with placeholder text: "First Name", "Last Name", "Email", and "Password". Below these fields is a link "Already have an account. Login Here" in black text. At the bottom of the form is a black button with the text "SignUp" in white.

Form Name: User Login

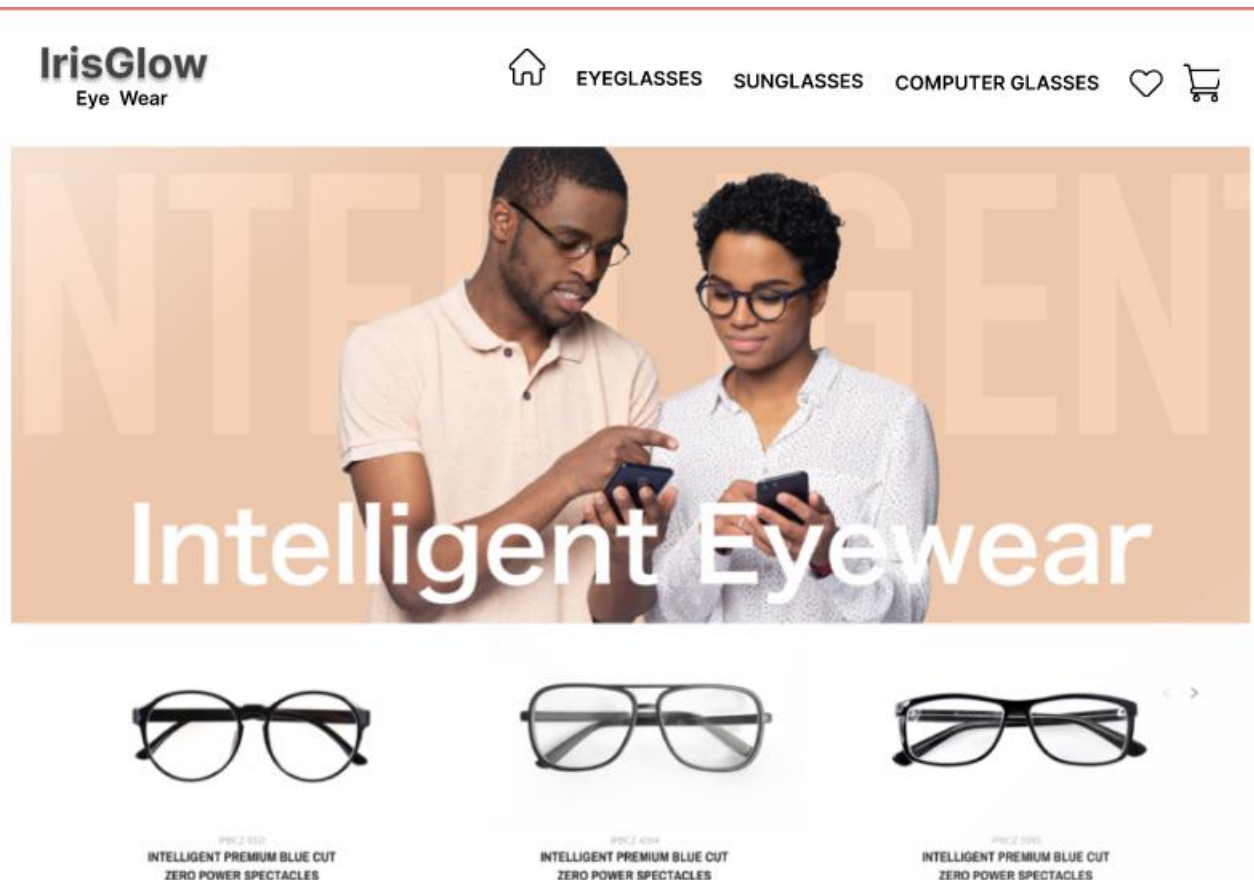


The User Login form is centered on a light yellow background with a pattern of stylized eyes. The form includes a title "Login" in bold black text. Below the title is a subtitle "SignIn to your account" in black text. Below the subtitle are two light gray input fields with placeholder text: "Username" and "Password". Below these fields is a blue button with the text "Login" in white. Below the button is a link "I forgot my password. Click here to reset" in black text. At the bottom of the form is a white button with the text "Register New Account" in black.

Form Name: Index Page



Form Name: Frames Page



4.4 DATABASE DESIGN

An organized system called a database allows users to retrieve information quickly and easily and has the capacity to retain information. The main objective of any database is its data, which need protection. The database design process is divided into two steps. In order to create a database that as clearly as possible satisfies these objectives, the user requests must first be identified. This stage, called as Information Level Design, is carried out independently from any specific DBMS. UML is a popular language utilized for modeling systems and objects found in the real world, and one of the tools it employs is use case diagrams to represent their creation and operation. The system architecture is paralleled by a database design.

The database's data layout seeks to achieve the two objectives listed below

- Data independence
- Data Integrity

4.4.1 Relational Database Management System (RDBMS)

RDBMS stands for Relational Database Management System. All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS. It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd. Data is represented in terms of tuples (rows) in RDBMS. A relational database is the most commonly used database. It contains several tables, and each table has its primary key. Due to a collection of an organized set of tables, data can be accessed easily in RDBMS.

Relations, Domains & Attributes

A table is a data structure representing a relation between entities, where the rows in a table, commonly referred to as tuples, are composed of an ordered set of n elements. These elements are organized into columns, also known as attributes, which define the characteristics of the entities. Relationships between tables in a database are established to ensure both referential and entity relationship integrity. To define the domain of a data attribute, a set of atomic values is specified, typically drawn from a particular data type. Assigning a name to the domain aids in interpreting its values and is a common practice

Relationships

- Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key is Super Key and Candidate Keys

4.4.2 Normalization

During the initial phase, data is grouped together in a basic manner to minimize the impact of future changes on data structures. The process of normalizing data structures formally aims to enhance data integrity and reduce duplication by organizing it into logical and efficient groupings. Using the normalization technique, superfluous fields are removed and a huge table is divided into several smaller ones. Anomalies in insertion, deletion, and updating are also prevented by using it. Keys and relationships are two notions used in the standard form of data modelling. In a table, each row is distinguished by a unique key, which may either be a primary key or a foreign key. A primary key is a single or multiple elements that act as an exclusive identifier for the particular row in the table, in a table that serves as a means of distinguishing between records from the same table. A column in a table known as a foreign key is used to uniquely identify records from other tables. Up to the third normal form, all tables have been normalized. It means placing things in their natural form, as the name suggests. By using normalization, the application developer aims to establish a coherent arrangement of the data into appropriate tables and columns, where names may be quickly related to the data by the user. By removing recurring groups from the data, normalization prevents data redundancy, which puts a heavy strain on the computer's resources.

These consist of

- Select appropriate table and column names.
- Normalize the data.
- Choose the proper name for the data

First Normal Form(1NF)

- A table is considered to be in the First Normal Form if it satisfies the requirement of atomicity, which means that the table's values cannot be divided into smaller, more granular parts, and each attribute or column contains only a single value. In other words, each column should hold

only one piece of information, and there should be no repeating groups or arrays of values within a single row.

- The concept of atomicity in this context denotes that a cell within a database cannot contain more than one value, and must exclusively store a single-valued attribute.
- The first normal form in database normalization places limitations on attributes that have multiple values, are composed of multiple sub-attributes, or contain a combination of both. Therefore, in order to satisfy the conditions of the first normal form., these attributes need to be modified or removed, a relation must not have multi-valued or composite attributes, and any such attributes must be split into individual attributes to form atomic values.

E.g., The table contains information pertaining to students, including their roll number, name, course of study, and age.

	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

Figure 9 : Table not in 1NF

The table containing the records of students displays a violation of the First Normal Form due to the presence of two distinct values in the course column. To ensure compliance with the First Normal Form, the table has been modified resulting in the formation of a new table.

	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

Figure 10 : Table in 1NF

The First Normal Form is applied to achieve atomicity in the system, which ensures that each column has unique values. By following this normalization process, the data is organized into individual atomic values, eliminating any redundancies or repeating groups. As a result, data integrity is maintained, and the system can efficiently handle complex data manipulations and

updates.

Second Normal Form(2NF)

To meet requirements of Second Normal Form, a table must satisfy the criteria of First Normal Form as a prerequisite. Furthermore, the table must not exhibit partial dependency, which refers to a scenario where a non-prime attribute is dependent on a proper subset of the candidate key. In other words, the table should have no attributes that are determined by only a portion of the primary key.

Now understand the Second Normal Form with the help of an example.

Consider the table Location:

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

Figure 11 : Table not in 2NF

The Location table currently has a composite primary key consisting of cust id and storied, and its non-key attribute is store location. However, the store location attribute is directly determined by the storied attribute, which is part of the primary key. As a result, this table does not meet the requirements of second normal form. To address this issue and ensure second normal form is met, it is necessary to separate the Location table into two separate tables. This will result in the creation of two distinct tables that accurately represent the relevant data and relationships: one for customer IDs and store IDs, and another for store IDs and their respective locations.:

	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

Figure 12 : Table in 2NF

After eliminating the partial functional dependency in the location table, it can be observed that the column storing the location is now entirely dependent on the primary key of the same table. In other words, the location column is fully determined by the primary key, thereby ensuring greater data consistency and integrity in the table.

Third Normal Form

A table must meet the requirements of Second Normal Form in order to be considered in Third Normal Form, and also fulfill two additional conditions. The second condition states that non-prime attributes should not rely on non-prime characteristics that are not a part of the candidate key within the same table, thus avoiding transitive dependencies. A transitive dependency arises when $A \rightarrow C$ indirectly, due to $A \rightarrow B$ and $B \rightarrow C$, where B is not functionally dependent on A . The main objective of achieving Third Normal Form is to reduce data redundancy and guarantee data integrity.

For instance, let's consider a student table that includes columns like student ID, student name, subject ID, subject name, and address of the student. To comply with the requirements for Third Normal Form, this table must first meet the standards of Second Normal Form and then ensure that there are no transitive dependencies between non-prime attributes.

	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

Figure 13 : Table not in 3NF

Now to change the table to the third normal form, you need to divide the table as shown below: Based on the given student table, it can be observed that the `stu_id` attribute determines the `sub_id` attribute, and the `sub_id` attribute determines the subject (`sub`). This implies that there is a transitive functional dependency between `stu_id` and `sub`. As a result, the table does not satisfy the criteria for the third normal form. To adhere to the third normal form, the table must be divided into separate tables where each table represents a unique entity or relationship. In this case, the table can be divided into two tables: one for the student-subject relationship (`stu_id` and `sub_id`), and another for the subject information (`sub_id` and `sub`):

	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

Figure 14 : Table in 3NF

The two tables illustrate that the non-key attributes are completely determined by and reliant on the primary key. In the first table, the columns for name, sub_id, and addresses are all exclusively linked to the stu_id. Likewise, the second table demonstrates that the sub column is entirely dependent on the sub_id.

4.4.3 Sanitization

Data Sanitization involves the secure and permanent erasure of sensitive data from datasets and media to guarantee that no residual data can be recovered even through extensive forensic analysis. Data sanitization has a wide range of applications but is mainly used for clearing out end-of-life electronic devices or for the sharing and use of large datasets that contain sensitive information. The main strategies for erasing personal data from devices are physical destruction, cryptographic erasure, and data erasure. While the term data sanitization may lead some to believe that it only includes data on electronic media, the term also broadly covers physical media, such as paper copies. These data types are termed soft for electronic files and hard for physical media paper copies. Data sanitization methods are also applied for the cleaning of sensitive data, such as through heuristic-based methods, machine-learning based methods, and k-source anonymity.

4.4.4 Indexing

Indexing is used to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. The index is a type of data structure. It is used to locate and access the data in a database table quickly.

- **Primary Index** – Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation.
- **Secondary Index** – Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

- Clustering Index – Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

4.5 TABLE DESIGN

1. Tbl_users_login

Primary key: **user_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	user_id	int(10)	Primary Key	Unique identifier for the user
2	email	varchar(50)	Unique Key	User's email address
3	first_name	varchar(50)	Not Null	User's first name
4	lastame	varchar(50)	Not Null	User's last name
4	phone	varchar(12)	Not Null	User's phone number
5	password	varchar(50)	Not Null	User's hashed password
6	role	smallint	Not Null	User's role (Patient, Doctor, Admin)
7	is_admin	boolean	Not Null	Whether the user is an admin
8	is_staff	boolean	Not Null	Whether the user is staff
9	is_active	boolean	Not Null	Whether the user is active
10	is_superuser	boolean	Not Null	Whether the user is a superadmin

2.Tbl_users_registration

Primary key: **reg_id**

Foreign key: **user_id** references table **Tbl_users_login**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	reg_id	int(20)	Primary Key	Unique identifier for the user's profile
2	user_id	int(20)	Foreign Key	References the user associated with this profile
3	profile_picture	varchar(255)	-	Path to the user's profile picture
4	address	varchar(50)	Not Null	User's address
5	addressline1	varchar(50)	Not Null	User's address line 1

6	addressline2	varchar(50)	Not Null	User's address line 2
7	country	varchar(50)	Not Null	User's country
8	state	varchar(40)	Not Null	User's state
9	city	varchar(15)	Not Null	User's city
10	pin_code	varchar(8)	Not Null	User's PIN code
11	gender	varchar(4)	Not Null	User's gender
12	dob	date	Not Null	User's date of birth
13	profile_created_at	datetime	Not Null	Date and time when the profile was created
14	profile_modified_at	datetime	Not Null	Date and time when last modified

3.Tbl_appointments

Primary key: **appointment_id**

Foreign key: **patient_id** references table **Tbl_users_login** and **doctor_id** references **Tbl_Doctor**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	appointment_id	int(20)	Primary Key	Unique identifier for the appointment
2	date	date	Not Null	Date of the appointment
3	patient_id	int(30)	Foreign Key	References the client user
4	doctor_id	int(30)	Foreign Key	References the therapist user
5	time_slot	time	Not Null	Time slot for the appointment
6	created_date	datetime	Not Null	Date and time when the appointment was created
7	modified_date	datetime	Not Null	Date and time when the appointment was last modified
8	cancelled_date	datetime	Not Null	Date and time when the appointment was cancelled
9	status	varchar(20)	Not Null	Status of the appointment (e.g., Not Paid, Pending, Scheduled, Completed, Cancelled)

10	payment_status	boolean	Not Null	Payment status of the appointment (True or False)
11	cancel_status	boolean	Not Null	Cancellation status of the appointment (True or False)

4. Tbl_Specialities

Primary key: **speciality_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	speciality_id	int(20)	Primary Key	Unique identifier for the speciality
2	speciality_name	varchar(100)	Unique Key	Name of the speciality
3	description	text	Not Null	Description of the therapy
4	symptoms	text	Not Null	Symptoms of the speciality
5	diagnosis	varchar(50)	Not Null	Daignosis of the speciality
6	treatments	text	Not Null	Treatments of the speciality

5. Tbl_Doctor

Primary key: **doctor_id**

Foreign key: **user_id** references table **Tbl_users_login**

Foreign key: **speciality_id** references table **Tbl_Specialities**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	doctor_id	int(10)	Primary Key	Unique identifier for the doctor
2	bio	text	Not Null	Bio of the doctor
3	qualification	varchar(50)	Not Null	Qualification of the doctor
4	license	varchar(50)	Unique Key	Unique identifier for the certification
5	experience	int(10)	Not Null	Experience of the doctor
6	user_id	int(10)	Foreign Key	References the user associated who are doctor
7	speciality_id	int(10)	Foreign Key	References the speciality associated with this doctor

7.Tbl_payment

Primary key: **payment_id**

Foreign key: **user_id** references table **Tbl_users_login**

Foreign key: **appointment_id** references table **Tbl_appointment**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	payment_id	int(30)	Primary Key	Unique identifier for the payment
2	user_id	int(20)	Foreign Key	References the user associated with this payment
3	razorpay_order_id	varchar(255)	Not Null	Razorpay order ID
4	payment_id	varchar(255)	Not Null	Razorpay payment ID
5	amount	decimal(8,2)	Not Null	Amount paid
6	currency	varchar(5)	Not Null	Currency code (e.g., "INR")
7	timestamp	datetime	Not Null	Timestamp of the payment
8	payment_status	varchar(20)	Not Null	Payment status (e.g., Pending, Successful, Failed)
9	appointment_id	int(20)	Foreign Key	References the appointment associated with this payment

7.Tbl_leave_request

Primary key: **request_id**

Foreign key: **doctor_id** references table **Tbl_Doctor**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	request_id	int (20)	Primary Key	Unique identifier for the leave request
2	doctor_id	int(20)	Foreign Key	References the doctor associated with this leave request
3	date	date	Not Null	Date of the leave request

4	status	varchar(10)	Not Null	Status of the leave request (e.g., Pending, Accepted, Rejected)
---	--------	-------------	----------	---

8.Tbl_appointment_history

Primary key: **history_id**

Foreign key: **appointment_id** references table **Tbl_appointment**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	history_id	int(10)	Primary Key	Unique identifier for the leave request
2	appointment_id	int(10)	Foreign Key	References the appointment table
3	date	date	Not Null	Date of the appointment
4	status	varchar(10)	Not Null	Status of the appointment

9.Tbl_doctor_day_off

Primary key: **day_off_id**

Foreign key: **doctor_id** references table **Tbl_Doctor**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	day_off_id	int (20)	Primary Key	Unique identifier for the leave request
2	doctor_id	int(20)	Foreign Key	References the doctor associated with this leave request
3	date	date	Not Null	Date of the doctors's day off

10.Tbl_frame

Primary Key: **frame_id**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	frame_id	int(10)	Primary Key	Unique identifier for the frame.
2	name	varchar(255)	Unique, Not Null	Name of the frame.

3	brand_name	varchar(255)	Not Null	Brand name of the frame.
4	category	varchar(255)	Not Null	Category of the frame (Eyeglasses, Sunglasses, etc.).
5	product_description	text	Not Null	Description of the frame's product.
6	material_description	text	Not Null	Description of the frame's material.
7	gender	varchar(1)	Not Null	Gender for which the frame is suitable.
8	price	decimal	Not Null	Price of the frame.
9	stock_quantity	PositiveInteger Field	Not Null	Quantity of the frame in stock.
10	production_date	date	Not Null	Production date of the frame.
11	color	varchar(50)	Not Null	Color of the frame.
12	size	varchar(10)	Not Null	Size of the frame (Small, Medium, Large, etc.).
13	frame_material	varchar(20)	Not Null	Material of the frame (Metal, Plastic, etc.).
14	frame_style	varchar(20)	Not Null	Style of the frame (Circle, Rectangle, etc.).
15	temple_length	varchar(20)	Not Null	Length of the frame's temple.
16	bridge_size	varchar(20)	Not Null	Size of the bridge of the frame.
17	lens_width	varchar(20)	Not Null	Width of the lens of the frame.
18	thumbnail	imagefield	Not Null	Additional image of the frame.
19	image_2	imagefield	Not Null	Additional image of the frame
20	image_3	imagefield	Not Null	Additional image of the frame
21	image_4	imagefield	Not Null	Additional image of the frame
22	image_5	imagefield	Not Null	Additional image of the frame

11.Tbl_usercart

Primary Key: **cart_id**

Foreign Key: **user_id** references table **Tbl_users_login**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	cart_id	int(10)	PrimaryKey	Unique identifier for the user cart item.
2	user_id	integer	Foreign Key	References the associated user.

3	frame_id	integer	Foreign Key	References the associated frame.
4	created_at	datetime	Auto Now Add	Date and time when the cart item was created.
5	updated_at	datetime	Auto Now	Date and time when the cart item was last updated.
6	quantity	PositiveInteger Field	NotNull	Quantity of the frame in the cart.

12.tbl_wishlist

Primary key: **wishlist_id**

Foreign key: **product_id** references table **tbl_product** and **user_id** references table **tbl_reg**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	wishlist_id	int(11)	Primary Key	Unique identifier for each wishlist item
2	product_id	int(11)	Foreign Key	References table tbl_product
3	user_id	int(11)	Foreign Key	References table tbl_reg

13.tbl_stock

Primary key: **stock_id**

Foreign key: **frame_id** references table **tbl_frame**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	stock_id	int(11)	Primary Key	Unique identifier for each stock item
2	frame_id	int(11)	Foreign Key	Reference to the product for which the size stock is being tracked
3	size	positive integer	Not Null	Size for which the stock quantity is being tracked
4	stock_quantity	int(11)	Not Null	Quantity of stock available for the specified size

14.tbl_shipping_address

Primary key: **shipping_id**

Foreign key: **user_id** references table **tbl_user_login**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	shipping_id	int(11)	Primary Key	Unique shipping ID
2	name	varchar(100)	Not Null	Name associated with the shipping address
3	email	varchar(100)	Not Null	Email address associated with the shipping address
4	phone_no	varchar(12)	Not Null	Primary phone number for the shipping address
5	phone_no2	varchar(12)	Not Null	Secondary phone number for the shipping address
6	address	textfield(200)	Not Null	Detailed address information
7	country	varchar(100)	Not Null	Country of the shipping address
8	state	varchar(100)	Not Null	State of the shipping address
9	city	varchar(100)	Not Null	District of the shipping address
10	pin_code	varchar(100)	Not Null	PIN code of the shipping address
11	land	varchar(100)	Not Null	Landmark information for the shipping address
12	user_id	int(11)	Foreign Key	References table tbl_reg

15.tbl_payment2

Primary key: **payment_id**

Foreign key: **user_id** references table **tbl_user_login** and **frame_id** references table **tbl_frame**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	payment_id	int(11)	Primary Key	Unique product ID
2	user_id	int(11)	Foreign Key	References table tbl_reg
3	frame_id	int(11)	Foreign Key	References table tbl_product
4	total_amount	decimal	Not Null	Total amount for the order
5	payment_status	boolean	Not Null	Status indicating whether the payment for the order is completed or not
6	created_at	date time field	Not Null	Date and time when the order was created

16.tbl_order

Primary key: **order_id**

Foreign key: **payment_id** references table **tbl_payment2** and **frame_id** references table **tbl_frame**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	order_id	int(11)	Primary Key	Unique order ID
2	payment_id	int(11)	Foreign Key	References table tbl_payment
3	frame_id	int(11)	Foreign Key	References table tbl_product
4	quantity	int(11)	Not Null	Quantity of the product in the order item
5	item_total	decimal	Not Null	Total cost for the order item
6	size	positive integer	Not Null	Size of the product in the order item

17.tbl_delivery

Primary key: **delivery_id**

Foreign key: **order_id** references table **tbl_order** and **address** references table **shipping_address**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1.	devivery_id	int(10)	Primary key	Delivery_id
2.	order_id	int(10)	Foreign key	References table tbl_order
3.	name	varchar(100)	Not Null	Name of customer
4.	address	int(5)	Not Null	Address of customer
5.	phone_no	varchar(12)	Not Null	Phone number of customer
6.	country	varchar(100)	Not Null	Country of the delivery address
7.	state	varchar(100)	Not Null	State of the delivery address
8.	city	varchar(100)	Not Null	District of the delivery address
9.	pin_code	varchar(100)	Not Null	PIN code of the delivery address
10.	land	varchar(100)	Not Null	Landmark information for the delivery address

18. tbl_chatbot

Primary Key: **chat_id**

Foreign Key: **user_id** reference table **Tbl_user_login**

No	Field Name	Datatype (Size)	Key Constraints	Description of the Field
1	chat_id	int(11)	Primary Key	Unique identifier for the chatbot interaction
2	user_id	int(11)	Foreign Key	Reference to the user associated with this interaction
3	timestamp	DateTimeField	Not Null	Date and time of the chatbot interaction
4	message_content	varchar(255)	Not Null	Content of the chatbot message

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is the most common way of executing programming in a controlled way, to respond to the inquiry - Does the product act as determined? Programming testing is much of the time utilized in relationship with the term's confirmation and approval. Approval is the checking or testing of things, incorporates programming, for conformance and consistency with a related detail. Programming testing is only one sort of confirmation, which additionally utilizes strategies like surveys, investigation, reviews, and walkthroughs. Approval is the most common way of making sure that what has been indicated is what the client really cared about.

Different exercises which are frequently connected with programming testing are static examination and dynamic investigation. Static examination explores the source code of programming, searching for issues and assembling measurements without really executing the code. Dynamic examination takes a gander at the way of behaving of programming while it is executing, to give data, for example, execution follows, timing profiles, and test inclusion data. Testing is a bunch of movement that can be arranged in cutting edge and directed efficiently. Testing starts at the module level and work towards the mix of whole PCs based framework. Nothing is finished without testing, as its imperative outcome of the framework testing targets, there are a few principles that can act as testing goals. They are:

Testing is a process of executing a program with the intent of finding an error.

- A successful test is one that uncovers an undiscovered error.

On the off chance that a testing is led effectively as per the targets as expressed above, it would uncover mistakes in the product. Likewise testing show that the product capability seems, by all accounts, to be working as indicated by the particular, that exhibition necessity seems to have been met. There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness is supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs

5.2 TEST PLAN

A test plan defines the steps necessary to carry out different testing approaches, as well as the tasks that must be accomplished. The task of generating computer programmers, documentation, and data structures falls under the purview of software developers. Individual testing is required to make sure that every part of the programmed operates as planned. An independent test group (ITG) is ready to give developers feedback and spot any potential problems. The test strategy must include a quantification of the testing objectives. These goals may include measurements like the mean time to failure, the cost of fixing errors, the density of remaining errors, the frequency of recurrence, and the number of hours needed for regression testing. The testing levels could include:

- Unit testing
- Integration Testing
- Validation Testing or System Testing
- Output Testing or User Acceptance Testing
- Automation Testing
- Selenium Testing

5.2.1 Unit Testing

Unit testing centers confirmation exertion around the littlest unit of programming plan - the product part or module. Involving the part level plan portrayal as an aide, significant control ways are tried to reveal mistakes inside the limit of the module. The overall intricacy of tests and uncovered scope laid out for unit testing. The unit testing is white-box situated, and step can be led in lined up for different parts. The measured point of interaction is tried to guarantee that data appropriately streams into and out of the program unit under test. The nearby information structure is inspected to guarantee that information put away briefly keeps up with its honesty during all means in a calculation's execution. Limit conditions are tried to guarantee that all assertions in a module have been executed something like once. At long last, all mistake dealing with ways are tried. Trial of information stream across a module point of interaction are expected before some other test is started. On the off chance that information don't enter and exit appropriately, any remaining tests are unsettled. Specific testing of execution ways is a fundamental errand during the unit test. Great plan directs that blunder conditions be expected and mistake dealing with ways set up to reroute or neatly end handling when a

mistake happens. Limit testing is the last undertaking of unit testing step. Programming frequently falls flat at its limits. Unit testing was finished in Sell-Delicate Framework by regarding every module as discrete substance and testing every single one of them with a wide range of test inputs. A few imperfections in the inward rationale of the modules were found and were corrected. Subsequent to coding every module is tried and run separately. All pointless code were eliminated and guaranteed that all modules are working, and gives the normal outcome.

5.2.2 Integration Testing

Integration testing is orderly procedure for developing the program structure while simultaneously directing tests to reveal blunders related with connecting. The goal is to take unit tried parts and construct a program structure that has been directed by plan. The whole program is tried as entirety. Rectification is troublesome in light of the fact that detachment of causes is confounded by immense breadth of whole program. When these blunders are revised, new ones show up and the cycle go on in an apparently unending circle. In the wake of performing unit testing in the Framework every one of the modules were coordinated to test for any irregularities in the connection points. Besides contrasts in program structures were eliminated and an extraordinary program structure was developed.

5.2.3 Validation Testing or System Testing

This is the last move toward testing. In this the whole framework was tried in general with all structures, code, modules and class modules. This type of testing is famously known as Discovery testing or Framework tests. Black Box testing technique centers around the practical prerequisites of the product. That is, Black Box testing empowers the computer programmer to infer sets of info conditions that will completely practice all utilitarian prerequisites for a program. Black Box testing endeavors to track down mistakes in the accompanying classes; erroneous or missing capabilities, interface blunders, mistakes in information designs or outside information access, execution mistakes and instatement blunders and end mistakes.

5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

- Input Screen Designs

- Output Screen Designs

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

5.2.5 Automation Testing

An experiment suite is executed utilizing specific robotized testing programming devices as a component of the product testing method known as mechanization testing. The test stages are fastidiously completed by a human performing manual testing while situated before a PC. Moreover, the robotization testing programming might produce intensive test reports, think about expected and genuine discoveries, and enter test information into the Framework Under Test. Programming test computerization requires huge monetary and material data sources. Rehashed execution of a similar test suite will be vital during ensuing improvement cycles. This test suite can be recorded and replayed depending on the situation utilizing a test robotization device. No further human contribution is required once the test suite has been mechanized.

5.2.6 Selenium Testing

Selenium is a widely used open-source automation testing suite for web UI. Originally developed by Jason Huggins in 2004, Selenium supports automation across different browsers, platforms, and programming languages. It can be deployed on Windows, Linux, Solaris, Macintosh, and even supports mobile OS such as iOS, Windows Mobile, and Android. Selenium supports various programming languages through drivers specific to each language including C#, Java, Perl, PHP, Python, and Ruby. Selenium Web Driver is the most popular with Java and C#. Test scripts can be coded in any supported language and run directly in modern web browsers such as Internet Explorer, Mozilla Firefox, Google Chrome, and Safari. Selenium is not only used for functional tests but can also be integrated with automation test tools like Maven, Jenkins, and Docker for continuous testing. Furthermore, it can be integrated with TestNG and JUnit for managing test cases and generating reports.

Test Case 1: Login Test Case

Code:

```
package Defenition;

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.marionette", "C:\\Users\\AKKU SHAJI\\eclipse-workspace\\");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("user is on login page")
    public void user_is_on_login_page() throws Exception {
        driver.navigate().to("http://127.0.0.1:8000/login/");
        Thread.sleep(2000);
    }

    @When("user enters email and password")
    public void the_user_enters_credentials() {
        driver.findElement(By.id("email")).sendKeys("anjaliraj0308@gmail.com");
        driver.findElement(By.id("password")).sendKeys("Anjali@123");
    }

    @And("User clicks on login")
    public void user_clicks_on_login() {
        driver.findElement(By.id("submit")).click();
    }

    @Then("user is navigated to the home page")
    public void navigate_to_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:

```

Apr 18, 2024 12:40:23 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
  Given browser is open # Definition.loginsteps.browser_is_open()
  And user is on login page # Definition.loginsteps.user_is_on_login_page()
  When user enters email and password # Definition.loginsteps.the_user_enters_credentials()
  And User clicks on login # Definition.loginsteps.user_clicks_on_login()
Test Passed: User is on the home page
  Then user is navigated to the home page # Definition.loginsteps.navigate_to_home_page()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m8.803s

```

Test Report:

Project Name : IrisGlow					
Login Test Case					
Test Case ID: 1			Test Designed By: Akku Shaji		
Test Priority (Low/Medium/High): High			Test Designed Date: 10-04-2024		
Module Name: Login Page			Test Executed By: Ms. Lisha Varghese		
Test Title: Verify login with valid email and password			Test Execution Date: 16-04-2023		
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 displayed	Login page displayed	Pass
2	Provide Valid email	User Name: anjaliraj0308@gmail.com	User should be able to Login	User Logged in and navigated to the dashboard with records	
3	Provide Valid Password	Password: Anjali@123			
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: Profile Update Test Case

Code

```
package Defenition;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
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;

public class editprofile {

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

    @Given("the browser is open3")
    public void browser_is_open() {
        System.setProperty("webdriver.gecko.marionette", "C:\\\\Users\\\\AKKU SHAJI\\\\eclipse-workspace\\");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("the user is on the login page3")
    public void user_is_on_login_page() throws Exception {
        driver.navigate().to("http://127.0.0.1:8000/login/");
        Thread.sleep(2000);
    }

    @When("the user enters their email and password3")
    public void the_user_enters_credentials() {
        driver.findElement(By.id("email")).sendKeys("anjaliraj0308@gmail.com");
        driver.findElement(By.id("password")).sendKeys("Anjali@123");
    }

    @And("the user clicks on the login button3")
    public void user_clicks_on_login() {
        driver.findElement(By.id("submit")).click();
    }

    @Then("the user should be navigated to the home page3")
    public void navigate_to_home_page() throws Exception {
        // Thread.sleep(1000);
        //driver.navigate().to("http://127.0.0.1:8000/profile/");
        driver.findElement(By.id("myprofile")).click();

        //driver.findElement(By.id("edit")).click();
        WebElement editButton = driver.findElement(By.id("edit"));
        JavascriptExecutor js = (JavascriptExecutor) driver;
        js.executeScript("arguments[0].click();", editButton);

        // Wait for some time (you may replace this with an explicit wait as mentioned earlier)
        Thread.sleep(2000);

        // Don't forget to close the WebDriver when done.
        WebElement address = driver.findElement(By.id("address"));
        WebElement address1 = driver.findElement(By.id("address1"));
        WebElement address2 = driver.findElement(By.id("address2"));

        address.clear();
        address1.clear();
        address2.clear();

        // Enter new data
        address.sendKeys("Kizhakkedam");
        address1.sendKeys("Koruthod");
        address2.sendKeys("Kottayam");

        Thread.sleep(2000);
        WebElement saveButton = driver.findElement(By.id("save"));
        JavascriptExecutor js1 = (JavascriptExecutor) driver;
        js1.executeScript("arguments[0].click();", saveButton);
    }
}
```

Output

Apr 18, 2024 2:25:45 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/editprofile.feature:3
  Given the browser is open3                               # Definition.editprofile.browser_is_open()
  And the user is on the login page3                       # Definition.editprofile.user_is_on_login_page()
  When the user enters their email and password3           # Definition.editprofile.the_user_enters_credentials()
  And the user clicks on the login button3                 # Definition.editprofile.user_clicks_on_login()
  Then the user should be navigated to the home page3      # Definition.editprofile.navigate_to_home_page()
```

1 Scenarios (1 passed)

5 Steps (5 passed)

0m15.311s

Test report :

Project Name : IrisGlow					
Profile Update Test Case					
Test Case ID: 2			Test Designed By: Akku Shaji		
Test Priority (Low/Medium/High): High			Test Designed Date: 10-04-2024		
Module Name: Update profile			Test Executed By: Ms. Lisha Varghese		
Test Title: Verify update profile			Test Execution Date: 16-04-2024		
Description: Testing the update profile page					
Pre-Condition: Require valid profile fields and update button					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login page		Login Page should be displayed	Login page displayed	Pass
2	Navigation to Profile Page		Profile page should be displayed	Profile page displayed	Pass
3	Provide Address line1	Pulimoottil	User should be able to update profile	Profile updated and message should be displayed.	Pass
4	Provide Address line2	Korthudu			
5	Provide Address line3	Kottayam			
6	Click on save button				
Post-Condition: User is inserted into the database and successfully registered. Profile details updated and inserted successfully into database.					

Test Case 3: Search Test Case

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 search {
    WebDriver driver;
    boolean testPassed = true;
    @Given("the user is logged in and on the home page")
    public void the_user_is_logged_in_and_on_the_home_page() {
        // Implement code to log in and navigate to the home page
        System.setProperty("webdriver.gecko.marionette", "C:\\Users\\AKKU SHAJI\\eclipse-workspace");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
        // Log in logic here
        driver.get("http://127.0.0.1:8000/login/");
        driver.findElement(By.id("email")).sendKeys("anandhu686513@gmail.com");
        driver.findElement(By.id("password")).sendKeys("Amal@123");
        driver.findElement(By.id("submit")).click();
    }

    @When("the user clicks on the \"Our Doctors\" link in the navigation bar")
    public void the_user_clicks_on_doctors_link() {
        // Implement code to click on the "Doctors" link
        driver.findElement(By.LinkText("Our Doctors")).click();
    }

    @And("the user searches for the doctor {string} in the search input")
    public void the_user_searches_for_the_doctor_in_the_search_input(String doctorName) {
        // Implement code to enter the doctor name in the search input
        WebElement searchInput = driver.findElement(By.id("search-input"));
        searchInput.sendKeys(doctorName);

        // Implement code to trigger the search (if needed)
        WebElement searchButton = driver.findElement(By.id("search"));
        searchButton.click();
    }

    @Then("the search results for {string} should be displayed")
    public void the_search_results_for_should_be_displayed(String doctorName) {
        // Implement code to verify that search results for "Amal" are displayed
        WebElement searchResultsContainer = driver.findElement(By.id("search-results"));
        WebElement doctorCard = searchResultsContainer.findElement(By.xpath("//h2[text()='\" + doct
        Assert.assertTrue(doctorCard.isDisplayed());
    }
}
```

Output

```
Apr 18, 2024 3:09:28 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 doctor named Amal # src/test/resources/Features/search.feature:3
  Given the user is logged in and on the home page # Definition.search.the_user_is_logged_in_and_on_the_home_page()
  When the user clicks on the "Our Doctors" link in the navigation bar # Definition.search.the_user_clicks_on_doctors_link()
  And the user searches for the doctor "Amal" in the search input # Definition.search.the_user_searches_for_the_doctor_in_the_search_input(java.lang.:
org.openqa.selenium.ElementNotInteractableException: Element <i id="search" class="bi bi-search"> could not be scrolled into view
Build info: version: '4.15.0', revision: '1d14b5521b'
System info: os.name: 'Windows 10', os.arch: 'amd64', os.version: '10.0', java.version: '17.0.1'
Driver info: org.openqa.selenium.firefox.FirefoxDriver
Command: [1e11df57-0277-4de1-aa22-07ca62ec095a, clickElement {id=0c7baa16-6579-4605-a57d-43fb606de9c8}]
Capabilities {acceptInsecureCerts: true, browserName: firefox, browserVersion: 124.0.2, moz:accessibilityChecks: false, moz:buildID: 20240401114208, moz:dc
Element: [[FirefoxDriver: firefox on windows (1e11df57-0277-4de1-aa22-07ca62ec095a)] -> id: search]
Session ID: 1e11df57-0277-4de1-aa22-07ca62ec095a
at java.base/jdk.internal.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at java.base/jdk.internal.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:77)
at java.base/jdk.internal.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
at java.base/java.lang.reflect.Constructor.newInstanceWithCaller(Constructor.java:499)
at java.base/java.lang.reflect.Constructor.newInstance(Constructor.java:480)
at org.openqa.selenium.remote.codec.w3c.W3CHttpResponseCodec.createException(W3CHttpResponseCodec.java:200)
```

Test Report:

Project Name : IrisGlow					
Search Test Case					
Test Case ID: 3			Test Designed By: Akku Shaji		
Test Priority (Low/Medium/High): High			Test Designed Date: 10-04-2024		
Module Name: Search			Test Executed By: Ms. Lisha Varghese		
Test Title: Verify Search Function			Test Execution Date: 16-04-2024		
Description: Testing Search Function					
Pre-Condition: Require valid Search data					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to login page		Login Page should be displayed	Login page displayed	Pass
2	Navigation to home page		Nav bar should shown with search bar	Nav bar shown with search bar	Pass
3	Provide query to be searched	Amal	User should be able input in search bar	User was able input in search bar and search results are shown	Pass
4	Results are shown				
Post-Condition: Results for query are shown in the page.					

Test Case 4: Add to Cart Test Case

Code:

```
package Defenition;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class cart {
    WebDriver driver;
    @Given("the user is logged in and on the home page")
    public void the_user_is_logged_in_and_on_the_home_page() {
        // Implement code to log in and navigate to the home page
        System.setProperty("webdriver.gecko.marionette", "C:\\Users\\AKKU SHAJI\\eclipse-workspace\\");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
        // Log in logic here
        driver.get("http://127.0.0.1:8000/login/");
        driver.findElement(By.id("email")).sendKeys("anjaliraj0308@gmail.com");
        driver.findElement(By.id("password")).sendKeys("Anjali@123");
        driver.findElement(By.id("submit")).click();
    }

    @When("the user clicks on the \"Buy Frames\" link in the navigation bar")
    public void the_user_clicks_on_the_buy_frames_link() {
        // Implement code to click on the "Buy Frames" link
        driver.findElement(By.linkText("Buy Frames")).click();
    }

    @When("the user clicks on the \"Eyeglasses\" link in the navigation bar")
    public void the_user_clicks_on_the_eyeglasses_link() {
        // Implement code to click on the "Eyeglasses" link
        driver.findElement(By.linkText("Eyeglasses")).click();
    }

    @When("the user clicks on the \"View Details\" button")
    public void the_user_clicks_on_the_view_details_button() {
        // Implement code to click on the "View Details" button
        WebElement viewDetailsButton = driver.findElement(By.linkText("View Details"));
        viewDetailsButton.click();
    }

    @Then("the user clicks on the \"Add to Cart\" button")
    public void the_user_clicks_on_the_add_to_cart_button() {
        // Implement code to click on the "Add to Cart" button
        WebElement addToCartButton = driver.findElement(By.xpath("//button[contains(text(), 'Add to Cart')]"));
        addToCartButton.click();
    }
}
```

Output:

```
Apr 18, 2024 3:22:07 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: User adds a frame to the cart # src/test/resources/Features/search.feature:3
  Given the user is logged in and on the home page # Defenition.search.the_user_is_logged_in_and_on_the_home_page()
  When the user clicks on the "Buy Frames" link in the navigation bar # Defenition.search.the_user_clicks_on_the_buy_frames_link()
  And the user clicks on the "Eyeglasses" link in the navigation bar # Defenition.search.the_user_clicks_on_the_eyeglasses_link()
  And the user clicks on the "View Details" button # Defenition.search.the_user_clicks_on_the_view_details_button()
  Then the user clicks on the "Add to Cart" button # Defenition.search.the_user_clicks_on_the_add_to_cart_button()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m9.104s
```

Test Report:

Project Name : IrisGlow					
Add to Cart Test Case					
Test Case ID: 4			Test Designed By: Akku Shaji		
Test Priority (Low/Medium/High): High			Test Designed Date: 10-04-2024		
Module Name: Add to cart			Test Executed By: Ms. Lisha Varghese		
Test Title: Verify add to cart functionality			Test Execution Date: 16-04-2024		
Description: Testing the add to cart page					
Pre-Condition: Require valid profile fields and update button					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login page displayed	Pass
2	Navigation to home page		Home page should be displayed	Home page displayed	Pass
3	Navigate to Buy Frames link		Pages should be displayed	Pages displayed	Pass
4	Navigate to Eyeglasses				
5	A frame is selected				
6	Click on the Add to Cart button				
Post-Condition: Results for query are shown in the page.					

Test Case 5: Apply Filter Test Case

Code:

```
package Definition;

import org.openqa.selenium.By;

public class filter {

    WebDriver driver;

    @Given("the user is logged in and on the home page")
    public void the_user_is_logged_in_and_on_the_home_page() {
        // Implement code to log in and navigate to the home page
        System.setProperty("webdriver.gecko.marionette", "C:\\Users\\AKKU SHAJI\\eclipse-workspace\\
        driver = new FirefoxDriver();
        driver.manage().window().maximize();

        // Log in logic here
        driver.get("http://127.0.0.1:8000/login/");
        driver.findElement(By.id("email")).sendKeys("anjaliraj0308@gmail.com");
        driver.findElement(By.id("password")).sendKeys("Anjali@123");
        driver.findElement(By.id("submit")).click();
    }

    @When("the user clicks on the \"Buy Frames\" link in the navigation bar")
    public void the_user_clicks_on_the_buy_frames_link() {
        // Implement code to click on the "Buy Frames" link
        driver.findElement(By.linkText("Buy Frames")).click();
    }

    @When("the user clicks on the \"Eyeglasses\" link in the navigation bar")
    public void the_user_clicks_on_the_eyeglasses_link() {
        // Implement code to click on the "Eyeglasses" link
        driver.findElement(By.linkText("Eyeglasses")).click();
    }

    @When("the user selects {string} from the gender filter dropdown")
    public void the_user_selects_from_the_gender_filter_dropdown(String gender) {
        // Implement code to select the specified gender from the dropdown
        WebElement genderDropdown = driver.findElement(By.name("gender"));
        Select select = new Select(genderDropdown);
        select.selectByVisibleText(gender);
    }

    @When("the user clicks on the \"Apply Filter\" button")
    public void the_user_clicks_on_the_apply_filter_button() {
        // Implement code to click on the "Apply Filter" button
        driver.findElement(By.cssSelector("button.btn.btn-primary.mb-2")).click();
    }

    @Then("the filtered results should only display frames for {string}")
    public void the_filtered_results_should_only_display_frames_for(String gender) {
        // Implement code to verify that only frames for the specified gender are displayed
        List<WebElement> products = driver.findElements(By.className("product"));
        for (WebElement product : products) {
            // Check if the product's gender matches the expected gender
            String productGender = product.findElement(By.className("gender")).getText();
            Assert.assertEquals(gender, productGender);
        }
    }
}
```

Output:

```
Apr 18, 2024 9:08:18 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: User filters frames by gender # src/test/resources/Features/filter.feature:3
  Given the user is logged in and on the home page # Definition.filter.the_user_is_logged_in_and_on_the_home_page()
  When the user clicks on the "Buy Frames" link in the navigation bar # Definition.filter.the_user_clicks_on_the_buy_frames_link()
  And the user clicks on the "Eyeglasses" link in the navigation bar # Definition.filter.the_user_clicks_on_the_eyeglasses_link()
  And the user selects "Female" from the gender filter dropdown # Definition.filter.the_user_selects_from_the_gender_filter_dropdown
  And the user clicks on the "Apply Filter" button # Definition.filter.the_user_clicks_on_the_apply_filter_button()
  Then the filtered results should only display frames for "Female" # Definition.filter.the_filtered_results_should_only_display_frames

1 Scenarios (1 passed)
6 Steps (6 passed)
0m9.066s
```


Test Report:

Project Name : IrisGlow					
Filter Search Test Case					
Test Case ID: 5			Test Designed By: Akku Shaji		
Test Priority (Low/Medium/High): High			Test Designed Date: 10-04-2024		
Module Name: Filter search			Test Executed By: Ms. Lisha Varghese		
Test Title: Verify filter search functionality			Test Execution Date: 16-04-2024		
Description: Testing the filter search					
Pre-Condition: Require valid profile fields and update button					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login page displayed	Pass
2	Navigation to home page		Home page should be displayed	Home page displayed	Pass
3	Navigate to Buy Frames link		Pages should be displayed	Pages displayed	Pass
4	Navigate to Eyeglasses				
5	Select the category to be filtered				
6	Click on the Filter button				
Post-Condition: Results for query are shown in the page.					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the phase of the task where the hypothetical plan is transformed into a functioning framework. It very well may be viewed as the most urgent stage in accomplishing a fruitful new framework acquiring the client's certainty that the new framework will work and will be compelling and exact. It is basically worried about client preparing and documentation. Change for the most part happens about a similar time the client is being prepared or later. Execution essentially implies meeting another framework plan into activity, which is the most common way of changing over another reconsidered framework plan into a functional one. At this stage the principal responsibility, the best disturbance and the significant effect on the current framework movements to the client office. In the event that the execution isn't painstakingly arranged or controlled, it can make disarray and disarray.

Execution incorporates that multitude of exercises that occur to change over from the current framework to the new framework. The new framework might be an absolutely new, supplanting a current manual or computerized framework or it could be a change to a current framework. Legitimate execution is fundamental to give a dependable framework to meet association necessities. The most common way of placing the created framework in genuine use is called framework execution. This incorporates that multitude of exercises that happen to change over from the old framework to the new framework. The framework can be carried out solely after through testing is finished and assuming that it is viewed as working as per the details. The framework work force actually looks at the practicality of the framework. The more perplexing the framework being executed, the more elaborate will be the framework examination and plan exertion expected to carry out the three principal perspectives: instruction and preparing, framework testing and changeover.

6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the last establishment of the bundle in its genuine climate, as per the general inclination of the expected purposes and the activity of the framework. In numerous associations somebody who won't be working it, will commission the product improvement project. In the underlying stage individual uncertainty about the product yet we need to guarantee that the opposition doesn't develop, as one needs to ensure that: The dynamic client should know about the advantages of utilizing the new system. Their trust in the software is developed. Legitimate direction is granted to the client with the goal that he is agreeable in utilizing the application. Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process

won't take place.

6.2.1 User Training

The goal of user training is to give users the confidence to test and alter computer-based systems in order to finally accomplish the intended goals. Training becomes more important as systems get more complicated. As part of the user training process, participants are exposed to a variety of crucial tasks like data entering, handling error alerts, querying databases, calling up routines to generate reports, among others.

6.2.2 Training on the Application Software

The new application software requires users to first complete a basic computer literacy training course before utilising it. They should be shown how to access help resources, traverse the software panels, correct entry errors, and update data that has already been entered. Specific topics pertaining to the user group and their function in using the system or its components should also be covered in the training. The training for the programme should be adapted to the requirements of various user groups and hierarchical levels.

6.2.3 System Maintenance

The software maintenance phase is an essential part of the software development life cycle and occurs after a software product has been successfully implemented and is performing useful work. Maintenance is necessary to ensure that the system remains adaptable to changes in the system environment. While finding mistakes is a part of software maintenance, it is not the only task involved. There are many other activities involved in maintenance, such as updating documentation, fixing bugs, enhancing existing features, and adding new features. Effective maintenance can help ensure that the software remains functional, reliable, and efficient over time.

6.2.4 Hosting

Web hosting is the process of giving websites on the internet access and storage space. For a website to be accessible online, it must be kept on a server that is always linked to the internet. A web hosting company provides this service, allowing individuals and businesses to store the data, software, and files for their websites on a server that is maintained and managed by the hosting company.

Amazon Web Services (AWS)

Amazon Web Services, is a cloud computing platform offered by Amazon and it provides EC2, called Elastic Computer Cloud. EC2 provides scalable compute capacity in the cloud and allows users to manage virtual machines on the cloud.

Procedure for hosting a website on AWS using Amazon EC2 instance:

1. Create an AWS Account:
 - Go to the AWS website and click on the "Create an AWS Account" button.
 - Follow the instructions to create a new AWS account by providing your email address, password, and other required information.
 - Verify your email address and set up your billing information.
2. Set Up AWS EC2 Instance:
 - Log in to your AWS Management Console.
 - Navigate to the EC2 dashboard and launch a new instance.
 - Choose the "t2.micro" instance type and configure the instance settings.
 - Review and launch the instance.
3. Check Instance Status:
 - Once the instance is launched, check its status in the EC2 dashboard.
 - Verify that the instance has passed both 2/2 status checks, indicating it is running properly.
4. Set Inbound Rule in Security Group:
 - In the EC2 dashboard, navigate to the "Security Groups" section.
 - Select the security group associated with your EC2 instance.
 - Click on the "Edit inbound rules" button.
 - Add a new rule to allow inbound traffic on port 8000, either by selecting "Custom TCP Rule" from the "Type" dropdown menu or by manually specifying port 8000 and the source IP range (0.0.0.0/0 for allowing access from any IP).
 - Save the changes to apply the new inbound rule.
5. SSH into the Instance:
 - In the EC2 dashboard, select the newly created instance.
 - Click on the "Connect" button at the top of the dashboard to get instructions on how to connect to the instance using SSH.
 - Use the provided SSH instructions to connect to the EC2 instance using your preferred SSH client.
6. Install Python 3, pip, and Project Dependencies:

- Update the package list on the instance using the command: `sudo apt-get update`.
- Install Python 3 and pip with: `sudo apt-get install python3 python3-pip -y`.
- Navigate to the project directory where your Django project resides: `cd django-on-ec2`.
- Install the project dependencies listed in the requirements.txt file using pip: `pip install -r requirements.txt`.

7. Prepare the Database:

- Run database migrations to set up the database schema: `python3 manage.py makemigrations` followed by `python3 manage.py migrate`.

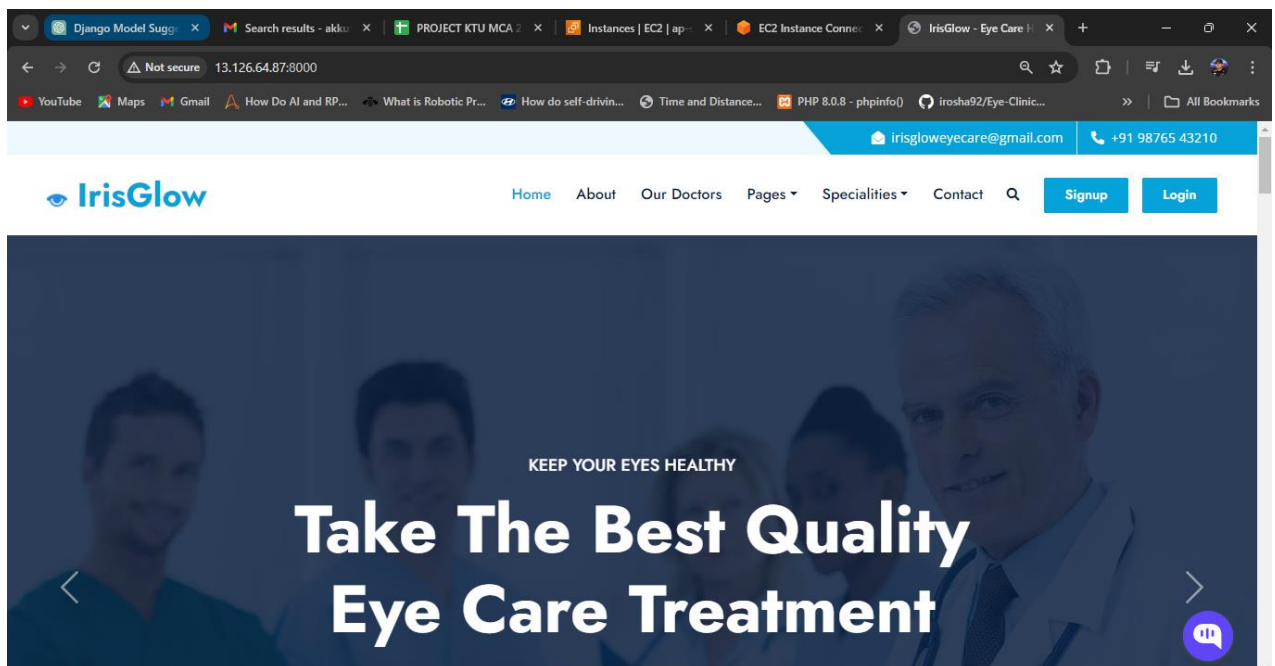
8. Host the App:

- Start the Django development server and bind it to all available network interfaces using 0.0.0.0: `python3 manage.py runserver 0.0.0.0:8000`.
- Access your Django app via a web browser using the public IP address of your EC2 instance followed by :8000.

Hosted Website:

Hosted Link: <http://13.126.64.87:8000/>

Screenshot:



CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In the realm of evolving healthcare technologies, IrisGlow emerges as a groundbreaking innovation in eye care hospital management. Departing from traditional methodologies, IrisGlow embodies efficiency and user-centric design, marking a significant advancement in the eye care sector. The project's primary aim is to establish a cutting-edge web platform that transforms the patient experience by simplifying access to eye care services and leveraging advanced technologies. Administrators gain efficient control over eye care offerings, ensuring the platform's currency and security. Users experience an intuitive and informative system, where they can delve into detailed eye care information, explore the expertise of medical professionals, and effortlessly schedule appointments. Rigorous testing guarantees a seamless and reliable user experience, promoting eye health and elevating the overall quality of care. IrisGlow aspires to be a holistic eye care platform, utilizing automation and personalization to propel individuals toward their eye health goals, fostering comprehensive well-being across the spectrum of eye care.

7.2 FUTURE SCOPE

Looking forward, IrisGlow is poised for expansive growth and innovation in eye care hospital management. The platform can diversify its offerings, incorporating additional eye care modalities beyond the outlined functionalities in the abstract. Integration of wearables data and health tracking capabilities can enhance diagnostic insights. IrisGlow has the potential to evolve into a broader healthcare marketplace, adding nutritionists, fitness experts, and life coaches to its network.

To boost personalization, features like sentiment analysis during patient consultations can be integrated. Strategic partnerships with hospitals could drive clinical use cases, facilitating doctor consultations and remote patient monitoring. Introducing a subscription model offers bundled eye care packages for users. Investing in emerging technologies like Virtual Reality can make eye care therapies more engaging. Through an expanded network, deeper personalization, and immersive technologies, IrisGlow aims to make comprehensive eye care accessible to all.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009
- Roger S Pressman, “*Software Engineering*”, 1994.
- Pankaj Jalote, “*Software engineering: a precise approach*”, 2006
- EEE Std 1016 Recommended Practice for Software Design Descriptions

WEBSITES:

- <https://stackoverflow.com/>
- www.w3schools.com
- www.agilemodeling.com/artifacts/useCaseDiagram.html

CHAPTER 9

APPENDIX

9.1 Sample Code

Views.py

```
import dataclasses
from decimal import Decimal
import json
from random import sample
from django.shortcuts import get_object_or_404,
render
from django.contrib.auth import login as auth_login
,authenticate, logout
from django.shortcuts import render, redirect
from DoctorApp.models import Appointments,
Specialities
from .forms import BootstrapDateInput,
BootstrapSelect, BootstrapTextInput,
CustomUserForm, SpecialityForm, UserProfileForm
from .models import CustomUser, PaymentP
from .decorators import user_not_authenticated
from .models import CustomUser, UserProfile
from django.contrib.auth import get_user_model
from django.urls import reverse
from django.http import HttpResponseRedirect
from django.contrib.auth import authenticate, login
from django.contrib import messages
from django.contrib.auth.decorators import
login_required
from django.contrib.auth import
update_session_auth_hash
from django.db.models import Q
from django.contrib.auth.decorators import
login_required
from django.views.decorators.cache import
never_cache
from social_django.models import UserSocialAuth
from social_django.utils import psa
from django.contrib.auth.models import User
from django.views.decorators.csrf import csrf_protect
```

```
User = get_user_model()
```

1. Login

```
def login_view(request):
    if request.user.is_authenticated:
        user=request.user
        if user.role == CustomUser.ADMIN:
            messages.success(request, 'Login Success!!')
            return redirect(reverse('admindashboard'))
```

```

elif user.role == CustomUser.PATIENT:
    messages.warning(request, 'Login Success!!')
    return redirect(reverse('index'))
elif user.role == CustomUser.DOCTOR:
    messages.success(request, 'Login Success!!')
    return redirect(reverse('doctordashboard'))
else:
    return redirect('/')
if request.method == 'POST':
    email = request.POST.get('email')
    password = request.POST.get('password')
    print(email,password)
    if email and password:
        user = authenticate(request, email=email,
password=password)
        print("authenticated")
        if user is not None:
            auth_login(request, user)
            # Redirect based on user_type
            if user.role == CustomUser.ADMIN:
                return redirect(reverse('admindashboard'))
            elif user.role == CustomUser.PATIENT:
                return redirect(reverse('index'))
            elif user.role == CustomUser.DOCTOR:
                return redirect(reverse('doctordashboard'))
            else:
                return redirect('/')
        else:
            return HttpResponseRedirect(reverse('login')
+ '?alert=invalid_credentials')
    else:
        return HttpResponseRedirect(reverse('login') +
'?alert=fill_fields')
    return render(request, 'login.html')

```

2. Register

```

@csrf_protect
def register(request):
    if request.user.is_authenticated:
        messages.warning(request, 'You are already logged in!')
        return redirect('index')
    elif request.method == 'POST':
        first_name = request.POST.get('first_name', None)
        last_name = request.POST.get('last_name', None)
        email = request.POST.get('email', None)
        phone = request.POST.get('phone', None)

```

```

password = request.POST.get('password', None)
confirmPassword = request.POST.get('confirmPassword', None)
role = CustomUser.PATIENT
if first_name and last_name and email and phone and password and role:
    if User.objects.filter(email=email).exists():
        return HttpResponseRedirect(reverse('register') + '?alert=email_is_already_registered')
    elif User.objects.filter(phone=phone).exists():
        return HttpResponseRedirect(reverse('register') +
'?alert=phone_no_is_already_registered')
    elif password != confirmPassword:
        return HttpResponseRedirect(reverse('register') + '?alert=passwords_do_not_match')
    else:
        user = User(first_name=first_name, last_name=last_name, email=email, phone=phone,
role=role)
        user.set_password(password) # Set the password securely
        user.save()
        user_profile = UserProfile(user=user)
        user_profile.save()
        return HttpResponseRedirect(reverse('login') + '?alert=registered')
return render(request, 'register.html')

```

3. Search

```

def search_doctor(request):
    query = request.GET.get('query')
    speciality = request.GET.get('speciality')
    experience = request.GET.get('experience')
    # Use Q objects to filter doctors based on first name, last name, or speciality name
    doctors = Doctor.objects.filter(
        Q(user__first_name__icontains=query) |
        Q(user__last_name__icontains=query)
    )
    if speciality:
        doctors = doctors.filter(speciality_name__speciality_name__icontains=speciality)
    if experience:
        doctors = doctors.filter(experience__gte=experience)
    # Create a list of dictionaries containing doctor information
    doctors_data = [
        {
            'id': doctor.user.id,
            'first_name': doctor.user.first_name,
            'last_name': doctor.user.last_name,
            'speciality_name': doctor.speciality_name.speciality_name,

```

```

        'profile_picture': doctor.user.userprofile.profile_picture.url,
        'qualification': doctor.qualification,
        'experience': doctor.experience,
    }
    for doctor in doctors
]
return JsonResponse({'doctors': doctors_data})
def doctor_search(request):
    return render(request, 'doctor_search.html')

```

4. Appointment

```

@login_required
def appointment(request, t_id):
    therapist = get_object_or_404(CustomUser, id=t_id)
    context = None
    if request.method == 'POST':
        date_str = request.POST.get('date')
        time_str = request.POST.get('time_slot')
        # Use datetime.strptime from the correct module
        date = datetime.strptime(date_str, '%Y-%m-%d').date()
        # Parse the date and time strings to datetime.date and datetime.time objects
        date = datetime.strptime(date_str, '%Y-%m-%d').date()
        time_parts = time_str.split(':')
        time_slot = time(int(time_parts[0]), int(time_parts[1]))
        # Check if the selected date is a day when the therapist has taken a day off
        if DoctorDayOff.objects.filter(doctor=therapist, date=date).exists():
            messages.error(request, 'The therapist is on leave on the selected date. Please choose a
different date.')
            return redirect('appointment', t_id=t_id)
        # Rest of your view code...
        existing_appointment = Appointments.objects.filter(client=request.user, date=date,
time_slot=time_slot).first()
        if existing_appointment:
            context = {
                'error': 'You have already scheduled an appointment for the selected Date and Time Slot',
                'therapist': therapist,
            }
        elif Appointments.objects.filter(client=request.user, date=date).exists():

```

```
        context = {
            'error': 'You have already scheduled an appointment for the selected Date.',
            'therapist': therapist,
        }
    else:
        is_time_slot_available = is_time_slot_available_for_doctor(therapist, date, time_slot)
        if is_time_slot_available:
            form = AppointmentForm(request.POST)
            form.instance.client = request.user
            form.instance.therapist = therapist
            if form.is_valid():
                # appointment = form.save()
                appointment1 = form.save()
                appointment_fee=300
                if appointment1.id is not None:
                    return redirect('payment', appointment_id=appointment1.id,
t_fees=appointment_fee)
            else:
                # Handle the case where the appointment instance is not saved
                print('Failed to save the appointment. Please try again.')
                else:
                    context = {
                        'error': 'The selected time slot is not available. Please choose a different time slot.',
                        'therapist': therapist,
                    }
        else:
            user = request.user
            initial_data = {
                'client': user,
                'client_name': user.first_name,
                'client_phone': user.phone,
                'therapist': therapist.id,
                'therapist_name': therapist.first_name,
            }
            appointment_form = AppointmentForm(initial=initial_data)
            user_form = CurrentUserForm(instance=user)
            context = {'appointment_form': appointment_form, 'user_form': user_form, 'therapist':
therapist}
```



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

def is_time_slot_available_for_doctor(therapist, date, time_slot):
    # Convert the selected time slot to a timezone-aware datetime object
    selected_time = timezone.datetime.combine(date, time_slot)
    # Check if the time slot is available for the specific doctor and date
    existing_appointment = Appointments.objects.filter(therapist=therapist, date=date,
time_slot=selected_time).first()
    return existing_appointment is None

def get_available_time_slots(request):
    therapist_id = request.GET.get('therapist_id')
    therapist = get_object_or_404(CustomUser, id=therapist_id)
    date = request.GET.get('date')

    # Fetch existing appointments for the selected date and therapist
    existing_appointments = Appointments.objects.filter(therapist=therapist, date=date)
    # Create a list of all available time slots
    all_time_slots = [time(9, 0), time(11, 0), time(13, 0), time(15, 0), time(17, 0)]
    # Initialize a dictionary to store the availability of time slots
    time_slot_availability = {time_slot: True for time_slot in all_time_slots}
    # Mark time slots as unavailable if they are already booked
    for appointment in existing_appointments:
        if appointment.time_slot in time_slot_availability:
            time_slot_availability[appointment.time_slot] = False
    # Filter the available time slots
    available_time_slots = [time_slot.strftime('%I:%M %p') for time_slot, is_available in
time_slot_availability.items() if is_available]
    return JsonResponse({'available_time_slots': available_time_slots})
```

5. Payment

```
def payment(request, appointment_id, t_fees):
    print(t_fees)
    t_fees = float(request.resolver_match.kwargs['t_fees'])
    print(t_fees)
    appointments = Appointments.objects.all()
    current_appointment = Appointments.objects.get(pk=appointment_id)
    # For Razorpay integration
    currency = 'INR'
    amount = t_fees # Get the subscription price
    amount_in_paise = int(amount * 100) # Convert to paise
    print(amount_in_paise)
```

```

# Create a Razorpay Order
razorpay_order = razorpay_client.order.create(dict(
    amount=amount_in_paise,
    currency=currency,
    payment_capture='0'
))
# Order ID of the newly created order
razorpay_order_id = razorpay_order['id']
callback_url = reverse('paymenthandler', args=[appointment_id]) # Define your callback URL
here
phone=current_appointment.client.phone
print(phone)
payment = Payment.objects.create(
    user=request.user,
    razorpay_order_id=razorpay_order_id,
    payment_id="",
    amount=amount,
    currency=currency,
    payment_status=Payment.PaymentStatusChoices.PENDING,
    appointment=current_appointment
)
appointment=current_appointment
# Prepare the context data
context = {
    'user': request.user,
    'appointment':appointment,
    'razorpay_order_id': razorpay_order_id,
    'razorpay_merchant_key': settings.RAZOR_KEY_ID,
    'razorpay_amount': amount_in_paise,
    'currency': currency,
    'amount': amount_in_paise / 100,
    'callback_url': callback_url,
    'phone':phone,
}
return render(request, 'client/razorpay_payment.html', context)

```

6. Payment Handler

```

@csrf_exempt
def paymenthandler(request, appointment_id):
    # Only accept POST requests.
    if request.method == "POST":
        # Get the required parameters from the POST request.
        payment_id = request.POST.get('razorpay_payment_id', "")
        razorpay_order_id = request.POST.get('razorpay_order_id', "")
        signature = request.POST.get('razorpay_signature', "")
        params_dict = {

```

```
'razorpay_order_id': razorpay_order_id,
'razorpay_payment_id': payment_id,
'razorpay_signature': signature
}
# Verify the payment signature.
result = razorpay_client.utility.verify_payment_signature(params_dict)

payment = Payment.objects.get(razorpay_order_id=razorpay_order_id)
amount = int(payment.amount * 100) # Convert Decimal to paise

# Capture the payment.
razorpay_client.payment.capture(payment_id, amount)
payment = Payment.objects.get(razorpay_order_id=razorpay_order_id)

# Update the order with payment ID and change status to "Successful."
payment.payment_id = payment_id
payment.payment_status = Payment.PaymentStatusChoices.SUCCESSFUL
payment.save()

try:
    update_appointment = Appointments.objects.get(id=appointment_id)
    print(update_appointment)
    update_appointment.status = 'pending'
    update_appointment.save()
    pay_amt= payment.amount
    payee = update_appointment.client.first_name
    email = update_appointment.client.email
    ap_date=update_appointment.date
    ap_time=update_appointment.time_slot
    therapist = update_appointment.therapist.first_name
    appointment_email(email, payee, ap_date, ap_time, pay_amt,therapist,payment)
except Appointments.DoesNotExist:
    # Handle the case where the appointment with the given ID does not exist
    return HttpResponseBadRequest("Invalid appointment ID")

# Render the success page on successful capture of payment.
return render(request, 'payment_confirmation.html',{'appointment':update_appointment})

else:
    update_appointment = Appointments.objects.get(id=appointment_id)
    update_appointment.payment_status = False
    update_appointment.save()

# If other than POST request is made.
return HttpResponseBadRequest()
```

8. Appointment.html

```

<!-- Appointment Start -->
<div class="container-fluid py-5">
  <div class="dcontainer">
    <div class="row gx-5">
      <div class="col-lg-6"></div>
      <div class="col-lg-6 mx-auto ccontainer">

        <h1 class="no-slots text-center mb-3" id="available-slots-description">
          Select a Date to Check Slot Availability
        </h1>
        <ul class="list-inline time-slots" id="available-slots">
          <!-- Time slots will be dynamically generated here -->
        </ul>
        <p class="text-success text-center" id="navslot"></p>

        <p class="no-slots " id="available-slots-description"></p>

      </div>
    </div>
    <div class="row gx-5">
      <div class="col-lg-6 mb-5 mb-lg-0">
        
      </div>
      <div class="col-lg-6 pt-5 mt-5">
        <div class="bg-light rounded p-5">
          <h1 class="mb-4 text-center">Book An Appointment</h1>
          <form method="post" id="appointmentForm">
            { % if error % }
            <span style="color: red">{ {error}} </span><br />
            { % endif % }
            <div class="row g-3 d-flex">
              { % csrf_token % }
              <div class="col-lg-6">{ {appointment_form.as_p}} </div>
              <div class="col-lg-6">{ {user_form.as_p}} </div>

              <div class="col-12">
                <button class="btn btn-primary w-100 py-3" type="submit">
                  Make An Appointment
                </button>
              </div>

            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>

```

```
</div>
<script>
    $(document).ready(function () {

        // Function to update available slots based on selected date
        function updateAvailableSlots() {
            // Get the selected date from the date input field
            var selectedDate = $("#date").val();
            var therapistId = "{ { therapist.id } }"; // Replace with the therapist's ID

            // Validate the selected date format
            var datePattern = /^\d{4}-\d{2}-\d{2}$/;
            if (!datePattern.test(selectedDate)) {
                // Invalid date format, do not make the AJAX request
                return;
            }

            // Send an AJAX request to your Django view to get available slots
            $.ajax({
                type: "GET",
                url: "{% url 'get-available-time-slots' %}",
                data: { date: selectedDate, therapist_id: therapistId },
                dataType: "json",
                success: function (data) {
                    // Update the available slots in the HTML
                    var availableSlots = data.available_time_slots;
                    var slotsList = $("#available-slots");
                    var description = $("#available-slots-description");
                    var navslot = $("#navslot");

                    slotsList.empty(); // Clear the existing list
                    if (availableSlots.length > 0) {
                        description.text("The available slots for the selected date are:");
                        $.each(availableSlots, function (index, slot) {
                            // Create a button element with the class "slot-button"
                            var slotButton = $("<button></button>")
                                .addClass("slot-button")
                                .text(slot);

                            // Always set the button's class to "available"
                            slotButton.addClass("available");

                            // Append the button to the list
                            slotsList.append(slotButton);
                            slotButton.css("margin-right", "10px");
                        });
                        navslot.text("Select your Desired Time from the Dropdown in the Appointment
Form");
                    } else {
                        description.text("No available slots for the selected date.");
                        slotsList.append("<p>No available slots</p>");
                    }
                }
            });
        }
    });
}
```

```
    }
  },
  error: function (xhr, status, error) {
    // Handle errors if needed
    console.error(xhr.responseText);
  },
});
}

// Listen for changes in the date input field
$("#date").on("change", function () {
  updateAvailableSlots();
});

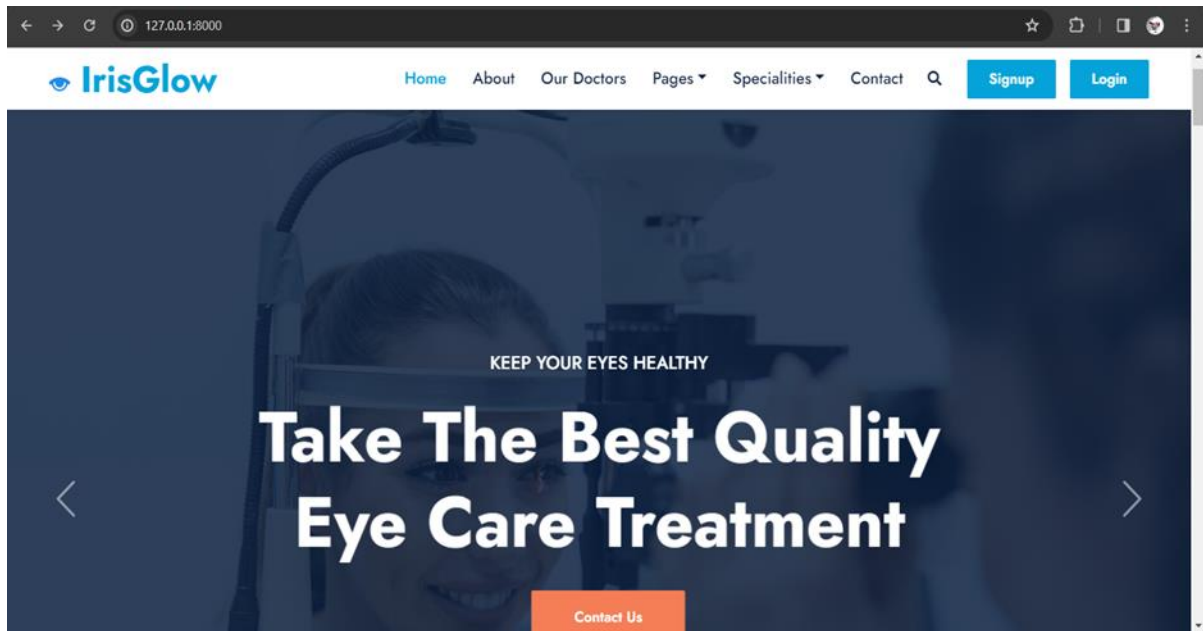
// Initialize available slots when the page loads
updateAvailableSlots();

// Function to handle cancellation of an appointment
function cancelAppointment(appointmentId) {
  if (confirm("Are you sure you want to cancel this appointment?")) {
    fetch(`/cancel_appointment/${appointmentId}`)
      .then(response => response.text())
      .then(data => {
        alert(data);
        // Reload the available slots after cancellation
        updateAvailableSlots();
      })
      .catch(error => console.error('Error:', error));
  }
}

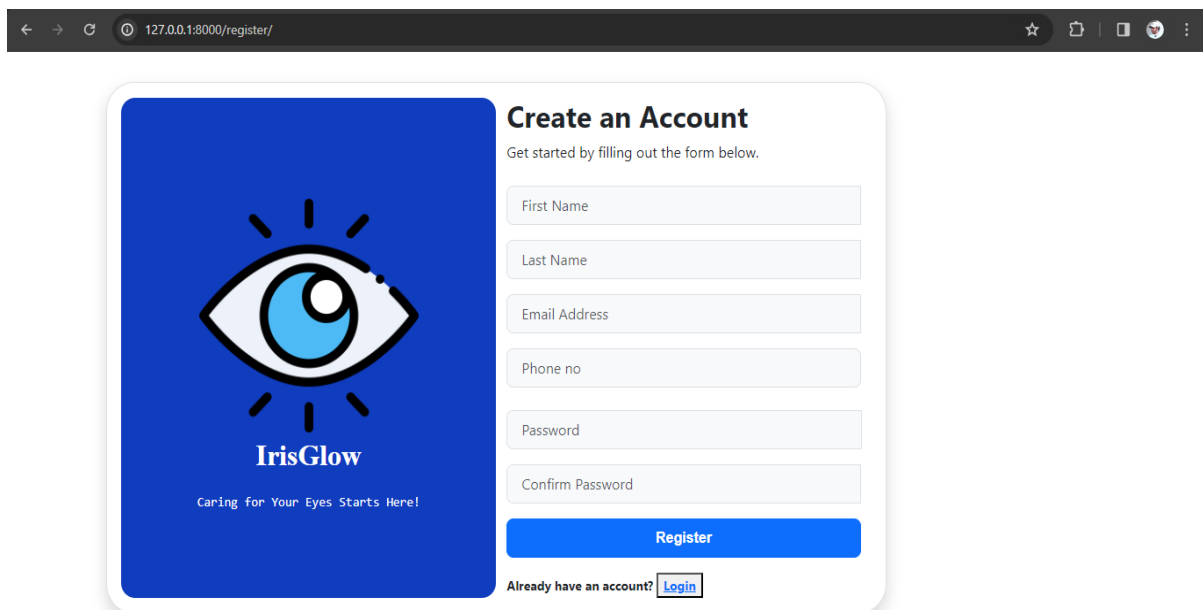
// Attach event listener to handle cancellation button clicks
$(document).on("click", ".cancel-appointment-button", function () {
  var appointmentId = $(this).data("appointment-id");
  cancelAppointment(appointmentId);
});
</script>
```

9.2 Screen Shots

Landing Page:



Registartion Page:

A screenshot of the IrisGlow registration page. The browser's address bar shows the URL '127.0.0.1:8000/register/'. The page is divided into two main sections. On the left is a blue square containing a white eye icon with a blue iris, the 'IrisGlow' logo, and the tagline 'Caring for Your Eyes Starts Here!'. On the right is a white box titled 'Create an Account' with the instruction 'Get started by filling out the form below.' Below this are six input fields: 'First Name', 'Last Name', 'Email Address', 'Phone no', 'Password', and 'Confirm Password'. A blue 'Register' button is positioned below the fields. At the bottom of the white box, it says 'Already have an account?' followed by a blue 'Login' button.


Profile Page:

← → ↻ 127.0.0.1:8000/profile/ 🔍 ☆ 📄 🌐 ⋮

IrisGlow

Home View Appointment History Welcome, Virat

My Profile



Change

Change Password

New password *

Confirm Password *

First Name : Virat

Last Name : Kohli

Email : viratkohli@gmail.com

Phone : +91 98765 41235

Address : Virat House, Virat Nagar, Shimla, Himachal

Country : India

State : Himachal Pradesh

City : Shimla

Pin Code : 789654

Gender : M

Date of Birth : July 22, 1991

Edit profile

View Doctors Page:


← → ↻ 127.0.0.1:8000/team/ ☆ 📄 🌐 ⋮

IrisGlow

Home About Service Pages Specialities Contact My Profile 🔍 Welcome, Virat Logout

Search by first name, last name, or speciality


Search



Dr. Alfiya PS


MBBS, MS (Ophthalmology)

Cataract



Dr. Amal Raj

MBBS, DOMS (AMU), DNB (Ophthalmology)



Dr. Abdul Roof

MBBS, BJMC

Glaucoma

Amal Jyothi College of Engineering, Kanjirappally

Department of Computer Applications

Book Appointment Page:

The available slots for the selected date are:
09:00 AM 11:00 AM 01:00 PM 03:00 PM 05:00 PM
Select your Desired Time from the Dropdown in the Appointment Form

Book An Appointment

Therapist name: Dr. Amal
First name: Virat
Date: 19-12-2023
Time slot: [dropdown]
Email: viratkohli@gmail.com
Phone: 98765 41235

Make An Appointment

Frames Page:

IrisGlow Eyewear Home Eyeglasses Sunglasses Computer Glasses Welcome, Anjali Logout

SUNGLASSES
Starting ₹899
SHOP NOW

Explore More Frames 🔍

