

V-CARE -The Healthcare App

Project Report Submitted By

ASHISH WILSON

Reg. No :AJC20MCA-2028

In Partial fulfillment for the Award of the Degree Of

**MASTER OF COMPUTER APPLICATIONS (2 Year)
(MCA)**

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



**AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY**

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

2020-2022

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**V-Care -The healthcare App**” is the bonafide work of **ASHISH WILSON (Reg.No:AJC20MCA-2028)** 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 2021-22.

Mr. Ajith G.S
Internal Guide

Ms. Nimmy Francis
Coordinator

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

External Examiner

DECLARATION

I hereby declare that the project report “**V-CARE -THE HEALTHCARE APP**” is a bonafided work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

Date:21/07/2022

KANJIRAPPALLY

ASHISH WILSON

Reg. No: AJC20MCA-2028

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev.Fr.Dr.MathewPaikatt** 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. Nimmy Francis** for their valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide, **Mr. Ajith G.S** for his inspiration and helping hand.

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

ASHISH WILSON

ABSTRACT

The recent days have been hard for everyone due to the COVID-19 pandemic and a lot of people are scared to even step out of their homes due to this. Hence, people are less likely to visit a doctor these days since they have to go out of their homes and visit the doctor. The doctor then might suggest another specialist depending on the severity and the type of the disease.

Hence, a more accessible method is required for people to get checked if they have a specific disease. In India, especially, there has been a surge of internet usage in recent times and a lot more people now have access to smartphones and other devices through which they can access the internet easily.

Therefore, we will be creating a system that helps the people know what kind of disease they might have instead of directly using search engines which could generally give out wrong data due to inappropriate searches of symptoms by people. After prediction of the disease a doctor will approve the predicted disease and give suitable remarks to users which help users to redirect to doctors which are available. User can book appointment and also chat with doctors in realtime this will help user to get to know about their condition and we can detect disease in early stages.

This application will assess the information given by the user by asking specific questions and their severity and then predict what disease they might have.

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	6
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
3.2	SYSTEM SPECIFICATION	11
3.2.1	HARDWARE SPECIFICATION	11
3.2.2	SOFTWARE SPECIFICATION	11
3.3	SOFTWARE DESCRIPTION	11
3.3.1	ANDROID	11
3.3.2	JAVA	12
3.3.3	TENSORFLOW LITE	12
3.3.4	FIREBASE	13
3.3.5	PYTHON	14
3.3.6	GOOGLE COLLAB	14
3.3.7	KERAS	14
4	SYSTEM DESIGN	16
4.1	INTRODUCTION	17
4.2	UML DIAGRAM	17
4.2.1	USE CASE DIAGRAM	18
4.2.2	SEQUENCE DIAGRAM	21
4.2.3	STATE CHART DIAGRAM	23
4.2.4	ACTIVITY DIAGRAM	26

4.2.5	CLASS DIAGRAM	27
4.2.6	OBJECT DIAGRAM	28
4.2.7	COMPONENT DIAGRAM	29
4.2.8	DEPLOYMENT DIAGRAM	29
4.5	USER INTERFACE DESIGN	30
4.6	DATA BASE DESIGN	33
5	SYSTEM TESTING	39
5.1	INTRODUCTION	40
5.2	TEST PLAN	41
5.2.1	UNIT TESTING	41
5.2.2	INTEGRATION TESTING	47
5.2.3	VALIDATION TESTING	47
5.2.4	USER ACCEPTANCE TESTING	47
6	IMPLEMENTATION	48
6.1	INTRODUCTION	49
6.2	IMPLEMENTATION PROCEDURE	49
6.2.1	USER TRAINING	50
6.2.2	TRAINING ON APPLICATION SOFTWARE	50
6.2.3	SYSTEM MAINTENANCE	50
7	CONCLUSION & FUTURE SCOPE	51
7.1	CONCLUSION	52
7.2	FUTURE SCOPE	52
8	BIBLIOGRAPHY	53
9	APPENDIX	55
9.1	SAMPLE CODE	56
9.2	SCREEN SHOTS	90
9.3	PLAGIARISM REPORT	99

List of Abbreviation

IDE	-	Integrated Development Environment
XML	-	Extensible Markup Language.
NoSQL	-	Not only Structured Query Language
UML	-	Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Most of the people tend to search on search engines like Google to find out what disease they might have. This could be effective but sometimes the search might show a different conclusion since the severity of some symptoms is not mentioned by the user. Therefore, this outcome could be inaccurate, it could show something very severe or minor, both of which could be harmful for the user and might cause him/her to panic.

The project focuses on the prediction of disease basis on the symptoms entered by the users and also help them by giving precautions. The prediction is based on the input that is entered by the users. After the disease predicted then the user can take doctor appointment and will get proper prescription of disease. Also doctor can chat with users to know about the health condition of the users.

1.2 PROJECT SPECIFICATION

The proposed system is an android app which is simple for users to get to know in which the users are able to enter the symptoms and get to know about the disease what the user is having. After the prediction any doctor for that specialization of predicted disease can approve the predicted disease and give remarks based on the symptoms entered by the users. After the disease prediction the system will also recommend user ,a doctor on specialization of predicted disease. User can take appointment with doctor if doctor approved the appointment .Also user can chat with doctor about his condition. The Modules in the system are as follows:

1. Admin

Admin must have a login for logging into this system. He has the overall control of the system. Admin will manage user data etc. He can view all the registered users. Admin in this system add doctors after checking details doctors will get a mail after admin added to the system and can disable a doctor and also verify a doctor if he updated details. Admin is able to view appointments and feedbacks of users about the doctors.

2. User

User will register their name and their details in this app via registration page. They can sign in the login page on successful registration. The user has right to edit and view his/her profile details. User can provide their symptoms and based on the symptoms ,

the disease will be predicted using machine learning algorithms. Also user can do appointment booking and chat with doctor if it is needed.

3.Doctor

Doctor is added by admin after proper verification with doctor as user of the application. Also doctor will be able to login with default credentials given by the admin. Before login as doctor, the doctor needs to change password of login details. Doctor can add slots depending upon their availability. The doctor has privileges to verify the appointment and prediction is done properly.

4.Appointment

The doctor can add the slot when he can attend the patients. So if the slots are not booked, the user or patient can book the appointment. The appointment will be completed only if the payment is done. For each disease there are different speciality doctors available. Based on the experience and speciality type the appointment will be booked. Once the appointment is booked, the user will book the slots and wait for the approval from doctor.

5.Chat

The user can chat with doctor if they need to know about disease details or to get more information. Chat will be enabled to users after successfully appointment then chat is real-time and chat we can share examination report and so all with help of chat we can get proper advice.

6. Disease Prediction

In our system we use various machine learning algorithms so it works based on the symptoms entered by the users and disease will be predicted and after prediction a doctor can check the symptoms and approve the prediction and suggest some remark.

7. Prescriptions

The doctor can enter the prescription and doctors can see the previous prescription provided to the user. Patient or users before seeing the prescriptions users can rate the doctor based on the appointment with that doctor.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis is a process of gathering and interpreting the facts, diagnosing health problems and the information or inputs to recommend improvements on the system. It is a problem solving activity that requires intensive or deep communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to every minute's detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the inputs to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming problem awareness, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made with the help of various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is understanding of how the system functions or works and this system is called the existing system. Now the existing system is subjected to close study and areas of problems are identified. The designer now functions as a problem solver and tries to sort out the difficulties of that enterprise faces. The solutions are given as many proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

2.2 EXISTING SYSTEM

Most of the people tend to search on search engines like Google to find out what disease they might have. This could be effective, but sometimes, the search engine might show a different conclusion since the some symptoms are not related and may lead to incorrect disease. Therefore, this outcome could be inaccurate, it could show something very severe or minor, both of which could be harmful for the user and might cause him/her to panic. There is no proper disease prediction system for users.

2.3 DRAWBACKS OF EXISTING SYSTEM

- No proper disease prediction system .
- Human effort is needed.
- It is difficult for old people to visit hospitals for minor diseases .
- After Disease prediction the user is not having proper guidance from doctor.
- No doctor recommendation after disease prediction.

2.4 PROPOSED SYSTEM

The proposed V-Care the healthcare app would help a lot more people self-diagnose so that they can take appropriate steps to counter the situations. For example, once the application gives an output mentioning a disease, the user might directly contact a specialist if needed instead of going to a general physician and can book a doctor on that speciality. In our disease prediction system user can get remarks and approval from the doctor about their predicted disease and help user . In later versions, we could also use this data to make this system better and also help the users to chat with doctors about

The system helps users by giving a predicted disease with precautions about the disease help users with the predictions also it gives a clear picture of his/her conditions. After getting know about the predictions user is able to get recommendation about the doctor of specific specialization and user is able to book appointment with doctors.

The prediction of disease is based on the symptoms entered by the users and it will giving appropriate prediction based on values on the training. The prediction is completely

depends on the symptoms entered by the users. User in our system can chat with doctors

and give feedback about the doctors also user and doctors can read live news about health and it help users to get updated about the health industry.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and easy to implement. The system requires very low system resources, and the system will work in almost all configurations. It has got following features:

➤ More secure: -

For data to remain secure some measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction that will compromise security. The system security problem can be divided into four related issues: security, integrity, privacy and confidentiality. Username and password requirement to sign in ensures security. It will provide data security as we are using the secured databases for maintaining the documents.

➤ Ensure data accuracy: -

The proposed system eliminates the manual errors while entering the details of the users during the registration.

➤ Better service: -

The product will avoid the burden of hard copy storage. We can also conserve the time and human resources for doing the same task. The data can be maintained for longer period with no loss of data.

➤ Prediction of disease: -

Using Disease prediction system diseases will be predicted with the help of symptoms entered by the user and give a prediction on basis of symptoms also the system will help users with precautions.

➤ Appointment booking Service: -

Appointment can be booked and it is easy for user to book slot and it is easily managable by user and doctors this help users to book .

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility study is done to see if the project after completion will fulfill the organization's objective for the work, effort and time spent on it. Feasibility study allows the developer to foresee the future and usefulness of the project. The feasibility study of a system proposal is based on its viability, which is the impact on the organization, the ability to meet their user needs, and the effective use of resources. So, when a new application is proposed, it goes through a feasibility study before being approved for development.

The document provides the feasibility of designing the project and lists the technical and economic feasibility in various areas which were carefully considered during the feasibility study of the project.. The following are its features: -

3.1.1 Economical Feasibility

The evolving system must be justified by cost and benefit. Criteria to ensure that effort is focused on the project that will yield the best and fastest returns. One of the factors affecting the development of a new system is its cost.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

The cost of project, V-care was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open source software

3.1.2 Technical Feasibility

The system must first be evaluated from a technical standpoint. This feasibility assessment should be based on the system requirement design in terms of input, output, programs and procedures. After an outline system has been identified, once the system has been designed, the investigation must continue to suggest the type of equipment to operate the system and the methodology required to develop the system.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed in such a way that the required functions and performance are achieved within the constraints. The project requires high-resolution scanning equipment and uses cryptographic techniques. Technology may become obsolete after some time, and the system may still be used because newer versions of the same software support older versions. So there are minimum restrictions involved in this scheme. The system is developed using Android, with Java on the front end and Firebase on the server at the back end, making the project technically feasible for development. This system is developed using Android. The System used was also of good performance of Processor Intel i3 core; RAM 4GB and, Hard disk 1TB

3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project will be beneficial as it fulfills the objectives when developed and installed. All behavioral aspects are carefully considered and the project is concluded to be behaviorally feasible..

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel core i5 8th Gen

RAM - 8 GB

Hard disk - 1 TB

3.2.2 Software Specification

Front End - XML

Backend - Java, Python

Database - Firebase

Client on PC - Windows 8 and above.

Technologies used - Java, Python, TensorFlow lite, Android

3.3 SOFTWARE DESCRIPTION

3.3.1 Android

Android OS is a mobile operating system with a Linux foundation that mostly powers smartphones and tablets. An operating system built on the Linux kernel, a GUI, a web browser, and end-user applications that can be downloaded are all included in the Android platform. The operating system was created to run on reasonably priced handsets with conventional numeric keypads, despite the fact that the early Android demonstrations used a basic QWERTY smartphone with a huge VGA screen. The Apache v2 open source licence, which was used for the distribution of Android, enables the creation of several OS variants for different hardware, including game consoles and digital cameras. Android is an open source software. Writing programmers that run on Android devices is possible with Java and the Android SDK.

3.3.2 Java

The Java programming language is used to create Android applications. That's really your only choice right now for native applications. Sun Microsystems created the incredibly successful programming language called Java (now owned by Oracle). Java, which was created much later than C and C++, integrates many of the powerful aspects of both languages while resolving some of its shortcomings. However, the strength of a programming language depends on its libraries. These libraries are available to aid programmers in creating applications.

The following are some of the key core aspects of Java:

- It's simple to learn and comprehend.
- It is intended to be safe and independent of platforms by utilizing
- simulated machines
- It is goal-oriented.

These Java foundational concepts are crucial to Android.

These Java foundational concepts are crucial to Android. The Android SDK offers a tonne of conventional Java libraries, including data structure libraries, math libraries, graphics libraries, networking libraries, and everything else you could possibly need, in addition to unique Android libraries that will aid you in creating fantastic Android applications.

3.3.3 Tensorflow Lite

TensorFlow Lite is a collection of technologies that makes it possible for developers to run their models on mobile, embedded, and edge devices, enabling on-device machine learning.

Key attributes:

- Optimized for on-device machine learning by addressing 5 main constraints: size (reduced model and binary size), battery consumption, privacy (no personal data leaves the device), and latency (there is no round-trip to a server) (efficient inference and a lack of network connections). Support for multiple platforms, including microcontrollers, embedded Linux, and Android and iOS smartphones. Support for a variety of languages, including Java, Swift, Objective-C, C++, and Python.
- High performance, with model optimization and hardware acceleration.
- End-to-end examples for typical machine learning problems like text

categorization, pose estimation, object detection, and question answering.

3.3.4 Firebase

A backend-as-a-service is Firebase (Baas). It offers a range of tools and services to developers so they can create high-quality apps, expand their user base, and make money. It is built using Google's technical framework.

A NoSQL database application, Firebase stores data in documents that resemble JSON.

- **Realtime Database:**

A NoSQL database housed in the cloud, the Firebase Realtime Database enables real-time syncing and storing amongst your users. In reality, the Realtime Database is essentially a sizable JSON object that the developers can control in real time. A Tree of Values => Real-Time Database The Firebase database gives your app access to both the data's current value and any updates to it using a single API. Your users will find it simple to access their data using any web-enabled or mobile device thanks to real-time syncing. Additionally, real-time databases facilitate user collaboration. Realtime Database also comes with web and mobile SDKs, enabling you to create apps without the use of servers, which is an incredible advantage. The Realtime Database SDKs employ local caching on the device to serve and save changes when your users go offline. The local data is automatically synchronised once the device is online.

- **Firebase Authentication**

To authenticate users for your project, Firebase Authentication offers ready-made UI libraries, simple SDKs, and backend services. Normally, setting up your own authentication system would take months. The system would need to be maintained by a dedicated team even after that. But if you utilise Firebase, you can set up the complete system—which will handle everything for you, even complex activities like account merging—in less than 10 lines of code. You can use the following techniques to verify users of your app: Phone numbers, Email and password, Facebook, Twitter, and more websites! Building safe authentication systems is made simpler by using Firebase Authentication, which also enhances the sign-in and onboarding experience for end users.

- **Cloud Storage for Firebase**

For app developers that need to store and serve user-generated material, such images or videos, Cloud Storage for Firebase was created. A robust, user-friendly, and reasonably priced object storage service designed for Google scale is Cloud Storage for Firebase. Regardless of network condition, the Firebase SDKs for Cloud Storage give Google security to file uploads and downloads for your Firebase apps.

Our SDKs can be used to store user-generated material such as pictures, audio, and video. You can access the same files on the server using Google Cloud Storage APIs.

3.3.5 Python

Python which is one of the most well-known general-purpose programming languages also it is high-level, dynamic programming language. Software developers ,data analysts ,network engineers, students, and accountants all use it, and it has one of the greatest growth rates of any language in the world.

It is an object-oriented, high-level, interpreted programming language. Because its source code is compiled into bytecode, which is then interpreted, it is known as an interpreted language. Before analysing Python code, CPython normally converts it to bytecode. Python can be used for machine learning as there are number of inbuilt methods in it.

3.3.6 Google Collab

Collab, to be precise, is a cloud-based, free Jupyter notebook environment. The best part is that there is no setup needed, and your team members can simultaneously edit the notes you create, just like you can with Google Docs projects. You may quickly load numerous well-known machine learning libraries supported by Collab into your notebook.

3.3.7 Keras

On top of Keras , open source machine libraries like TensorFlow, Theano, or Cognitive Toolkit are constructed (CNTK). For rapid numerical and computational tasks, Python's Theano package is used. TensorFlow is the most well-known symbolic math toolkit for creating neural networks and deep learning models. TensorFlow offers distributed

computing flexibility. The deep learning framework known as CNTK was developed by Microsoft. It uses C#, Python, and C++ libraries together with standalone machine learning toolkits. Theano and TensorFlow are extremely complex libraries for creating neural networks, despite their incredible capability. The basic structure of Keras makes it simple and quick to build TensorFlow or Theano-based deep learning models. Deep learning models may be quickly defined with Keras.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Any engineering system or product development process begins with design. Design is a creative process. The secret to an efficient system is a decent design. The process of using different methods and concepts to specify a process or a system in sufficient detail to allow its physical implementation is called "design". One way to describe it is the process of using different methods and concepts to specify a device, a process, or a system in sufficient detail to allow its physical reality. Regardless of the development model used, software design forms the technical core of the software engineering process. System design creates the necessary architectural details. To create a system or item. The best possible design phase for this program optimizes all efficiency, performance and accuracy, as is the case with any systematic technique. A user-oriented document is converted into a document for programmers or database staff throughout the design phase. The two phases of system design development are logical design and physical design.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for Unified Modeling Language. UML is different from other common programming languages like C++, Java, COBOL etc. UML is a pictorial language used to create software blueprints. UML can be described as a general purpose visual modeling language for visualizing, specifying, constructing and documenting software systems. Although UML is commonly used to model software systems, it is not restricted within this scope. It is also used to model non-software systems. For example, process flow in a manufacturing unit etc. UML is not a programming language, but tools can be used to generate code in various languages using UML diagrams. UML is directly related to object-oriented analysis and design. After some standardization, UML became an OMG standard. All elements and relationships are used to construct a complete UML diagram, and the diagram represents a system. The visual effect of the UML diagram is the most important part of the whole process. All the other elements are used to make it complete. UML includes the following nine diagrams.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- Statechart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a graphic representation of the interactions between the components of a system. A use case is a methodology used in systems analysis to identify, specify, and organize system requirements. In this context, the term "system" refers to something that is developed or operated, such as a mail-order product sales and service website. Use case diagrams are used in UML (Unified Modeling Language), a standard notation for modeling real-world objects and systems.

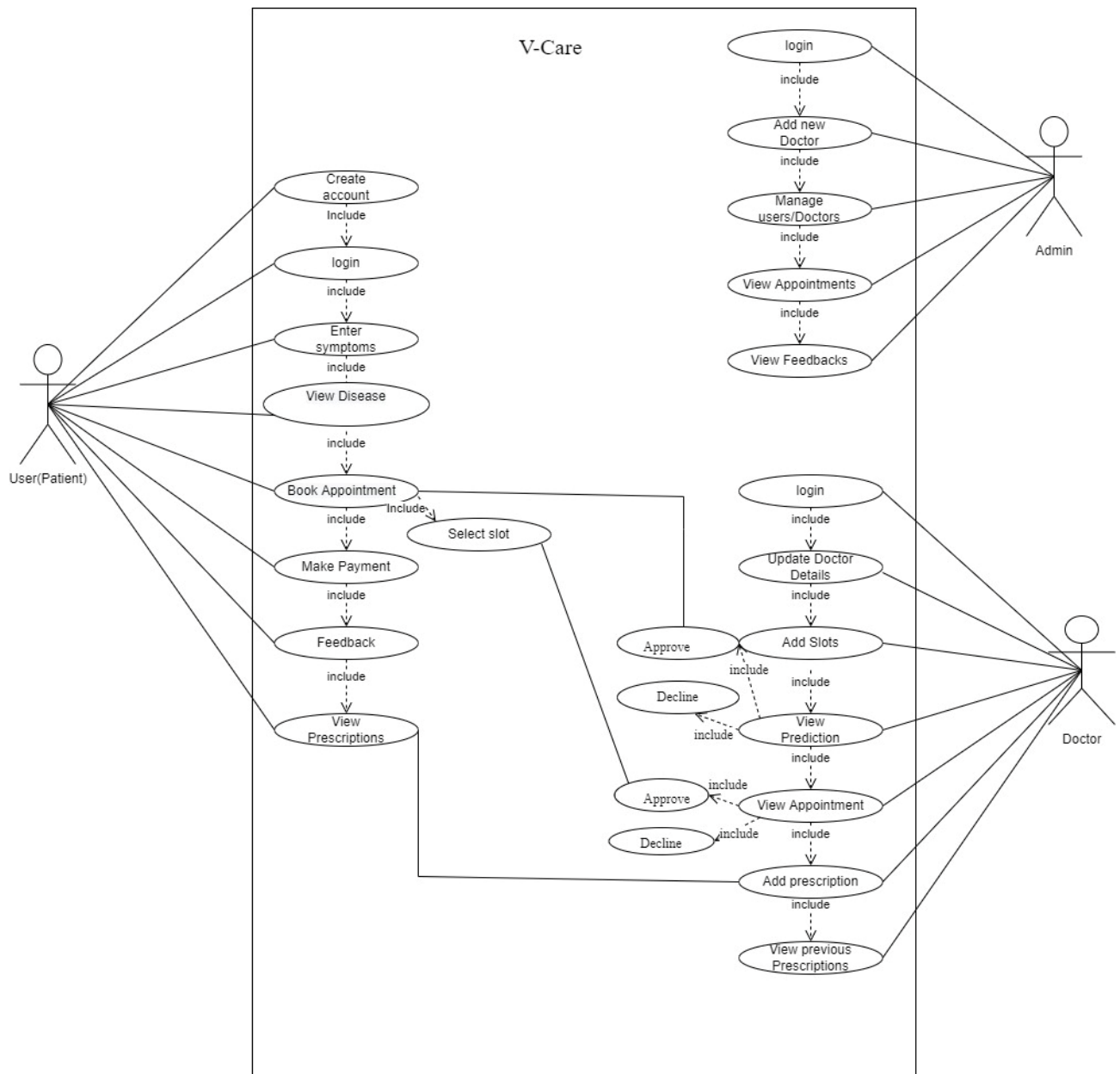
System objectives may include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a customer-service-oriented task. For example, use cases in a product sales environment include item ordering, catalog updating, payment processing, and customer relations. A use case diagram consists of four elements.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram. The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.

-
- Give a suitable name for actors.
 - Show relationships and dependencies clearly in the diagram.
 - Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
 - Use notes whenever required to clarify some important points.

Use Case Diagram



4.2.1 USE CASE DIAGRAM

4.2.2 SEQUENCE DIAGRAM

A sequence diagram depicts the interactions between objects in a sequence, i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system work. These diagrams are widely used by businessmen and software developers to document and understand the requirements of new and existing systems..

Sequence Diagram Notations –

- i. **Actors** – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.
- ii. **Lifelines** – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram
- iii. **Messages** – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

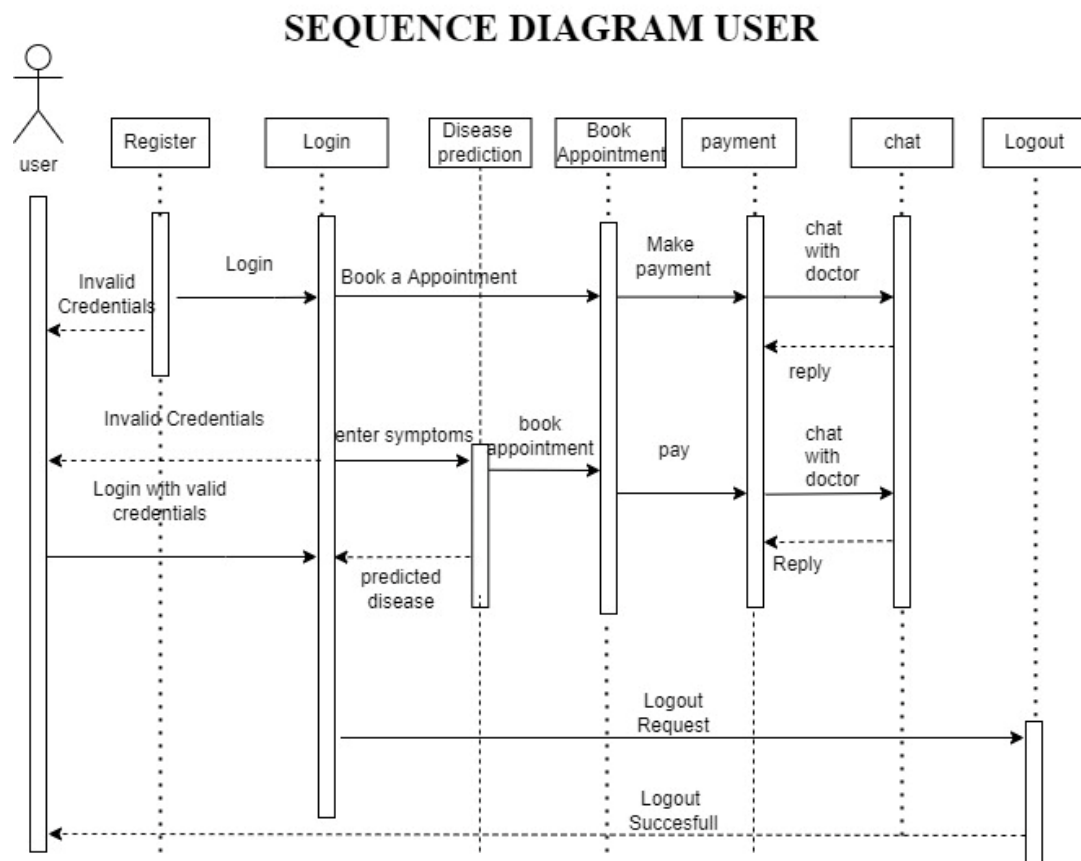
- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message

- Reply Message
- Found Message
- Lost Message

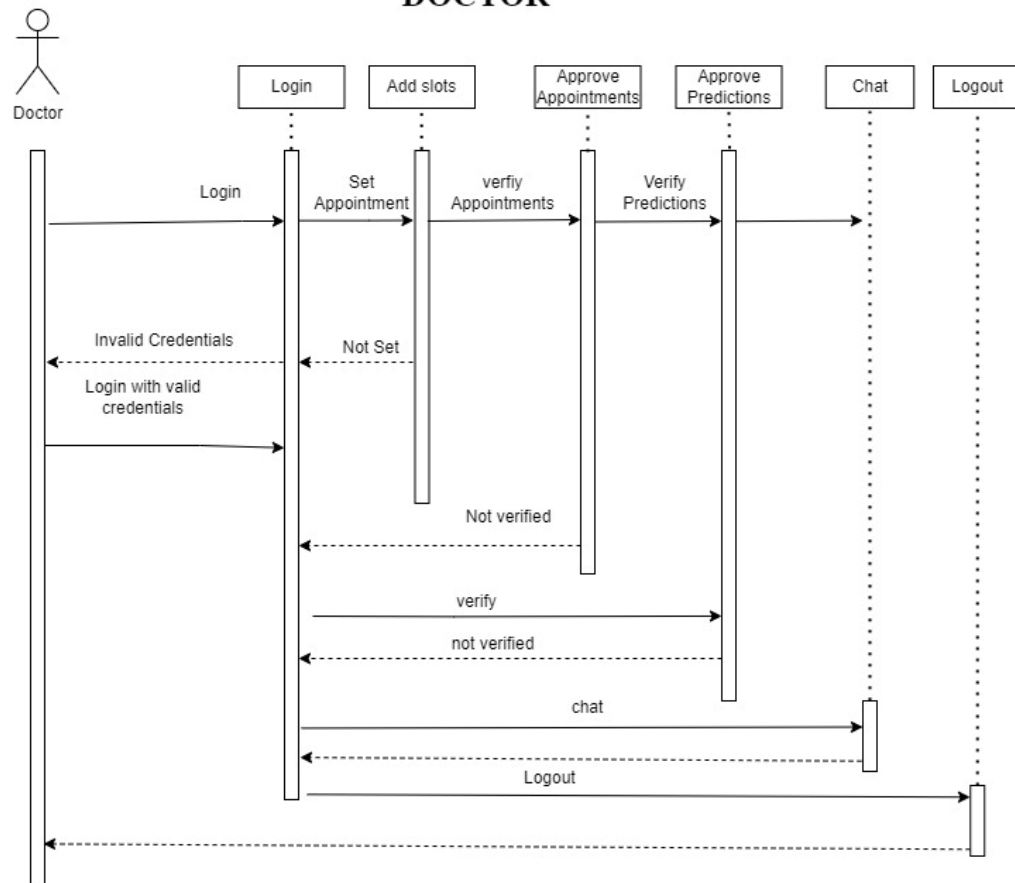
iv. Guards – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

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.



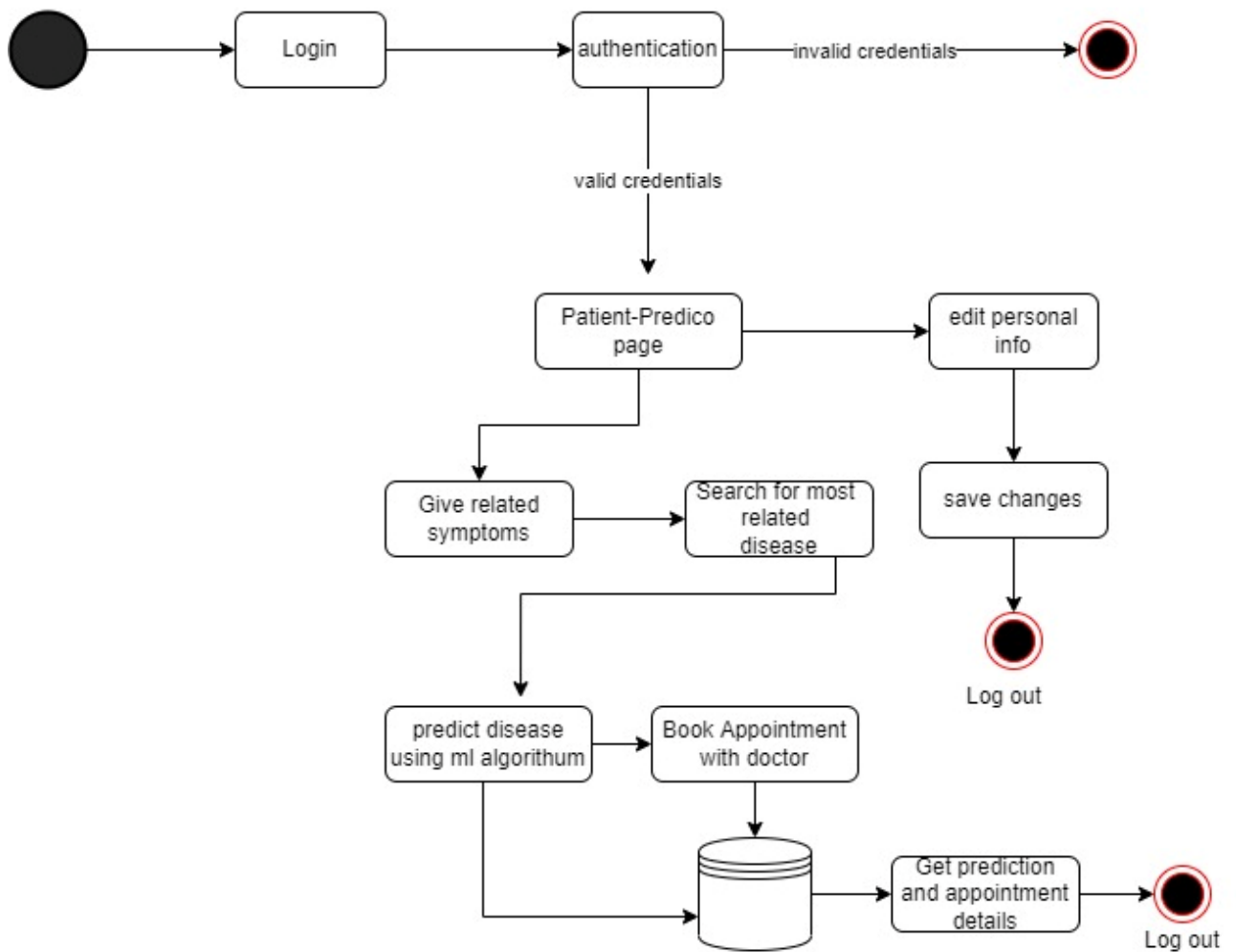
SEQUENCE DIAGRAM DOCTOR



4.2.3 STATE CHART DIAGRAM

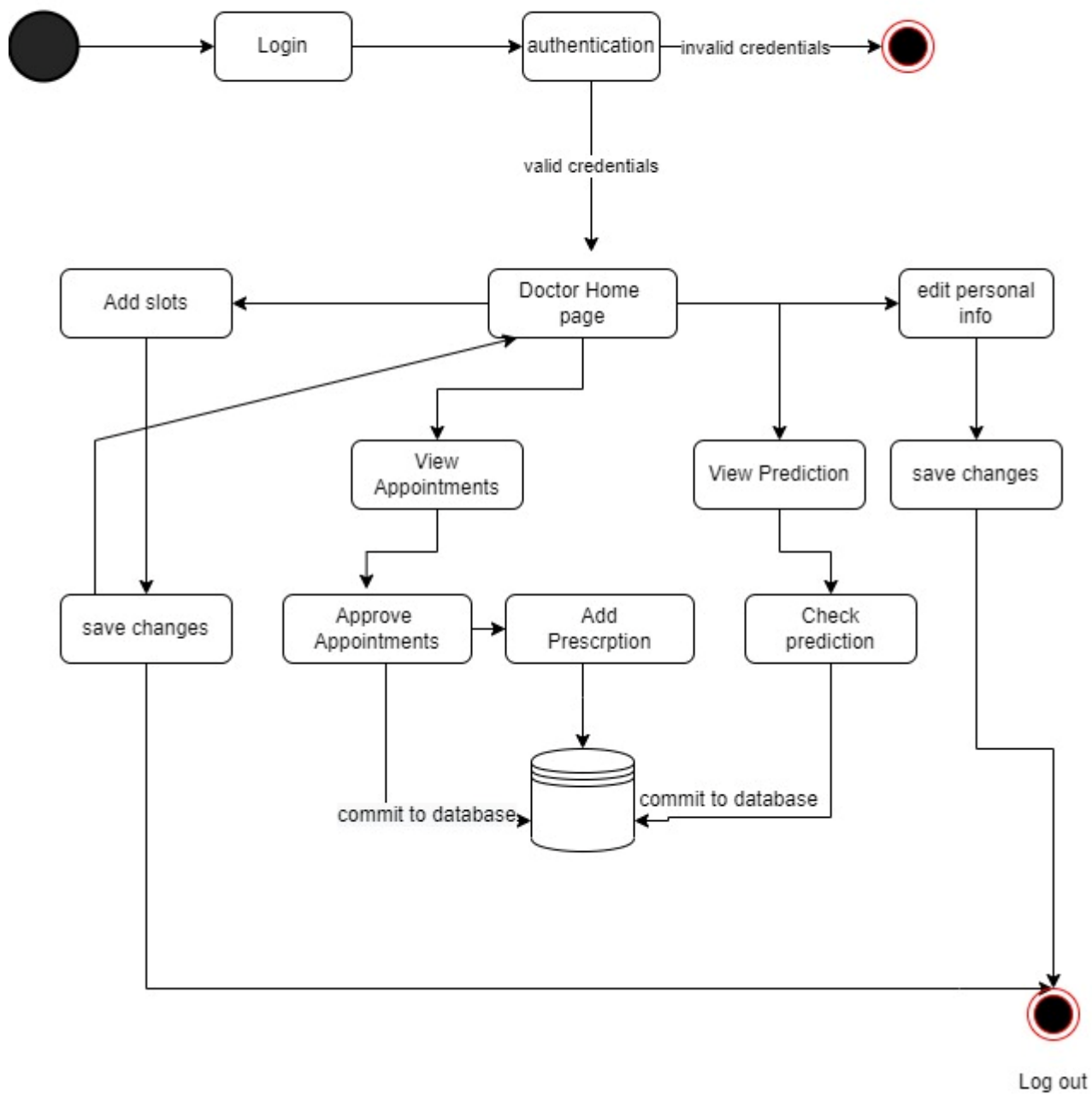
A state diagram is used to capture the behavior of a software system. UML state machine diagrams can be used to model the behavior of a class, a subsystem, a package, or an entire system. It is also called a state chart or state transition diagram. Statechart diagrams provide an efficient way to model the interactions or communication that occurs within a system, both external entities and within a system. These diagrams are used to model event-based systems. The state of an object is controlled with the help of an event. State chart diagrams are used to describe the various states of an entity within the application system.

V-Care State chart for patient



4.2.3 State Chart Diagram for user

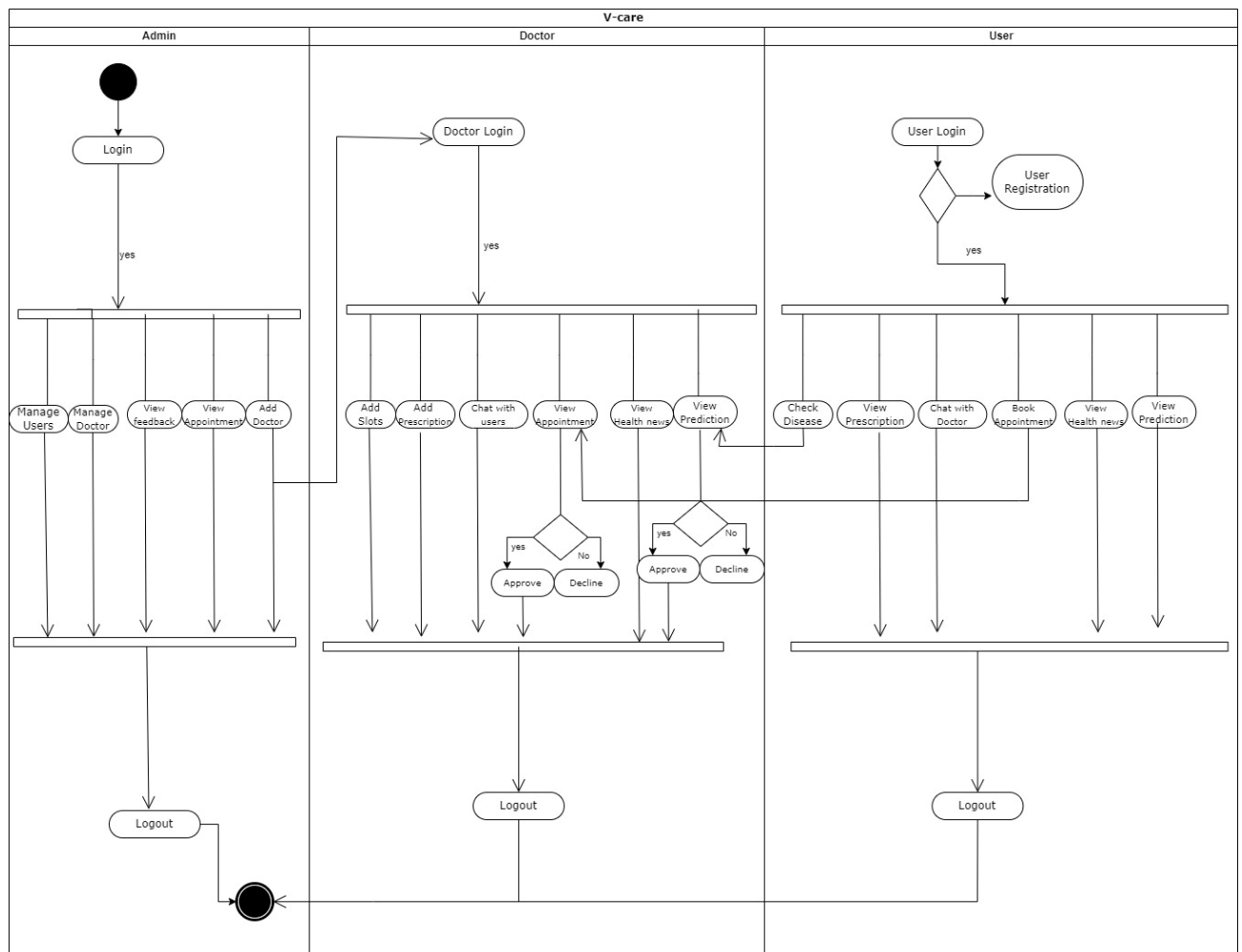
V-Care State chart for Doctor



4.2.3 State Chart Diagram for Doctor

4.2.4 ACTIVITY DIAGRAM

Activity diagrams describe how activities are coordinated to provide a service at different levels of abstraction. Typically, an event needs to be achieved by some activity, especially an activity intended to achieve several different things that require coordination, or how events in a single use case are related to each other, in particular, use cases in which activities are performed. May overlap and may require coordination. It is also ideal for modeling how a collection of use cases is coordinated to represent business workflows.

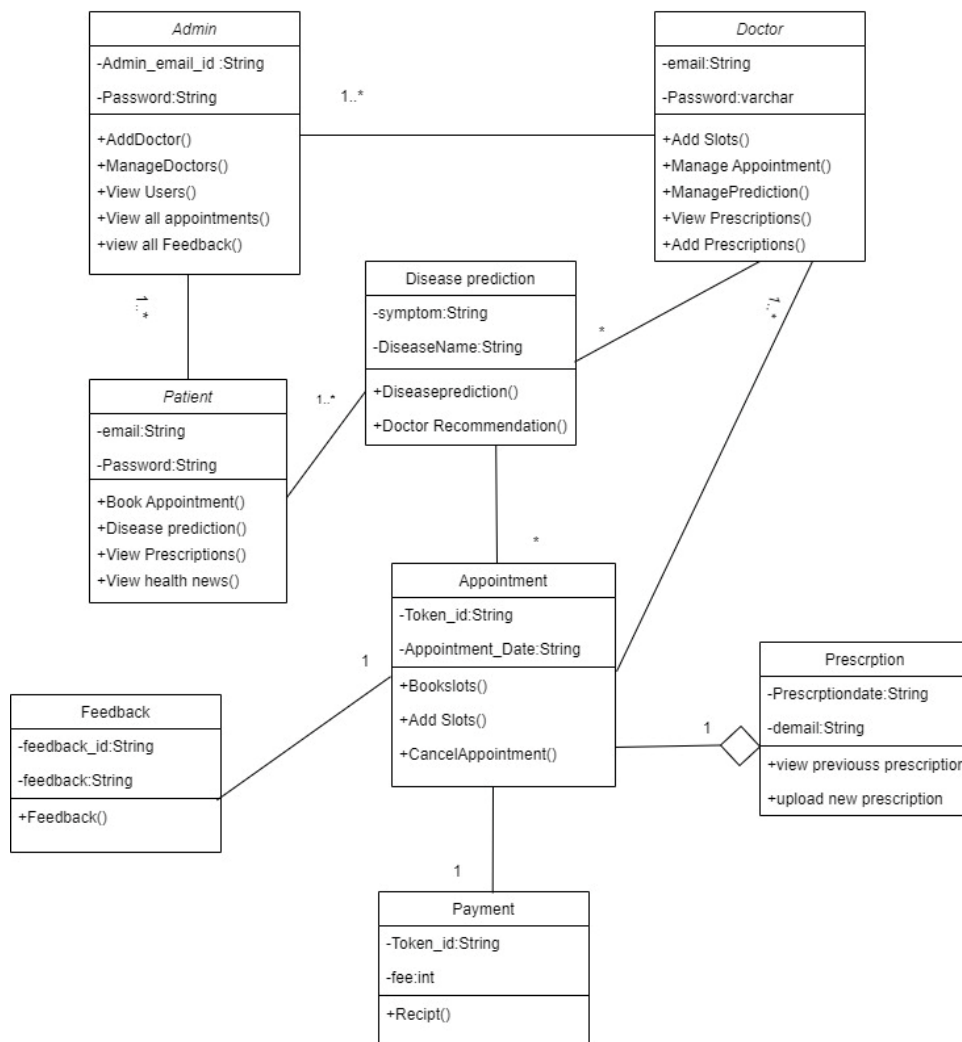


4.2.4 Activity Diagram

4.2.5 CLASS DIAGRAM

A class diagram is a static diagram. It represents a static view of an application. A class diagram is used not only to visualize, describe and document various aspects of a system, but also to construct the executable code of a software application. A class diagram describes the attributes and functions of a class and the constraints imposed on the system. Class diagrams are widely used in modeling object-oriented systems because they are UML diagrams that can be directly mapped using object-oriented languages. A class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as structural diagram.

Class Diagram

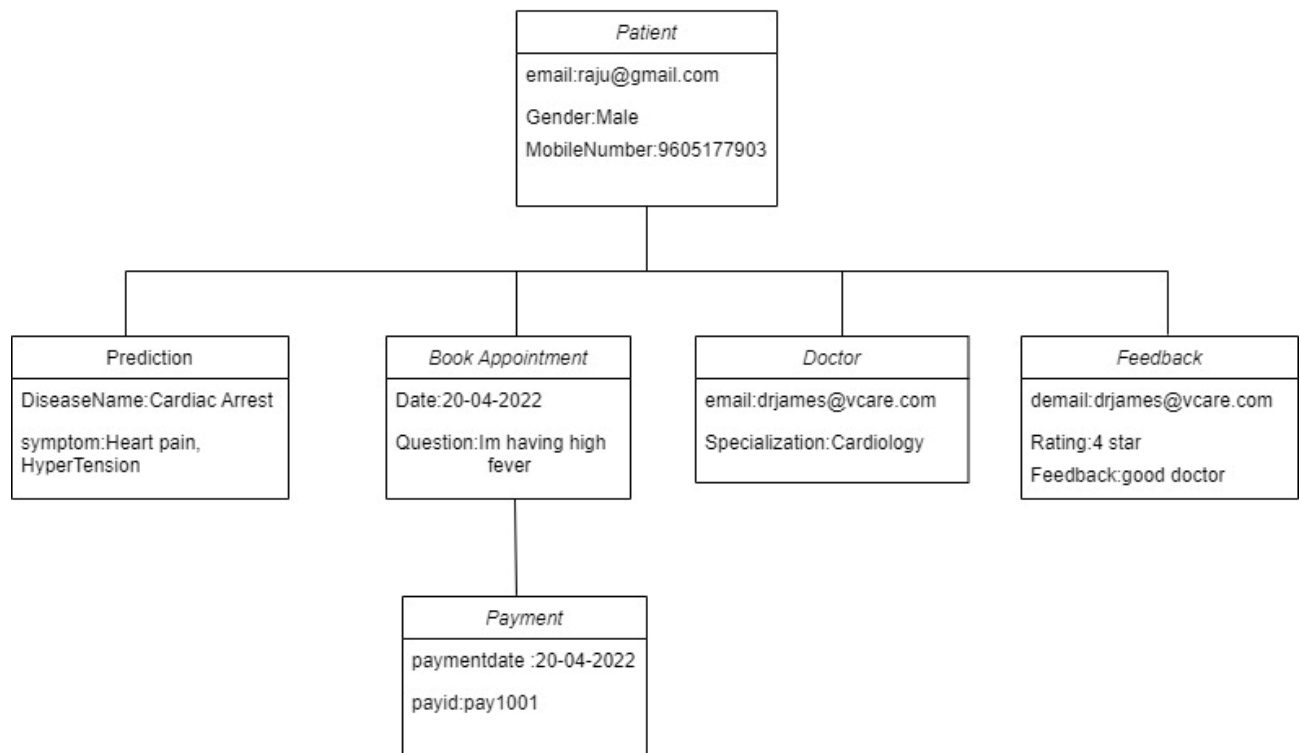


4.2.5 Class Diagram

4.2.6 OBJECT DIAGRAM

Object diagrams are derived from class diagrams, so object diagrams depend on class diagrams. Object diagrams represent an example of a class diagram. The basic concepts are similar to those of class diagrams and object diagrams. Object diagrams represent a static view of a system, but this static view is a snapshot of the system at a particular moment in time. Object diagrams are used to represent a set of objects and their relationships as an example.

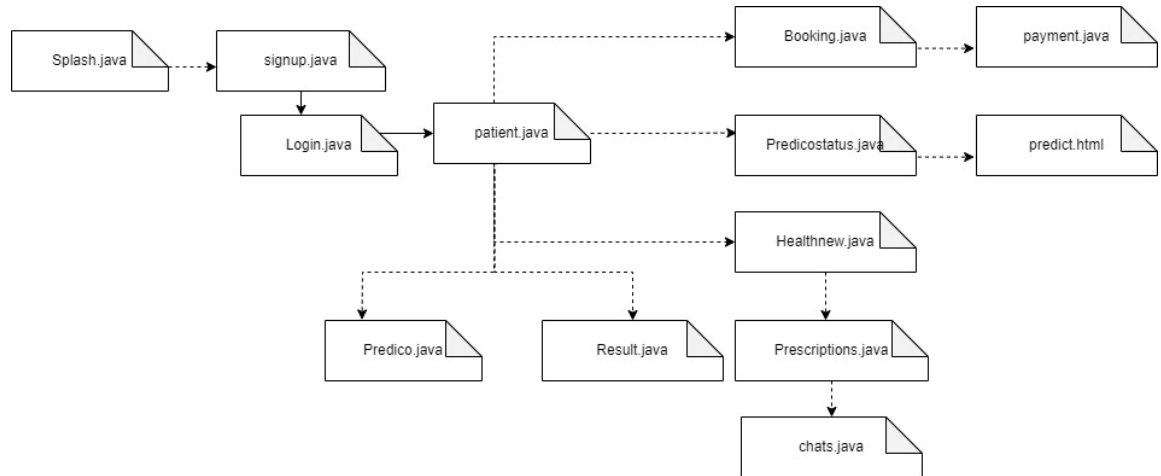
Object Diagram



4.2.6 Object Diagram

4.2.7 COMPONENT DIAGRAM

Component diagrams are different in nature and behavior. Component diagrams are used to model the physical aspects of a system. Physical aspects are components such as executables, libraries, files, and documents that reside on a node. Component diagrams are used to visualize the organization and relationships between components in a system. These diagrams are also used to build executable systems.



4.2.7 Component Diagram

4.2.8 Deployment Diagram

Deployment diagrams are used to depict the topology of a system's physical components, as well as the locations of software components. Deployment diagrams are used to describe a system's static deployment view. Nodes and their relationships are shown in deployment diagrams.

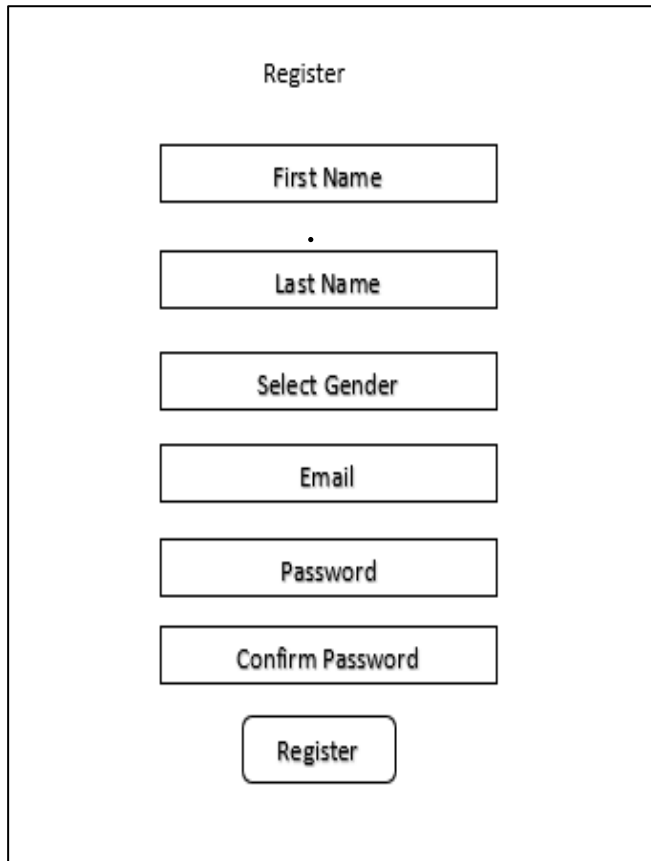


4.2.8 Deployment Diagram

4.5 USER INTERFACE DESIGN

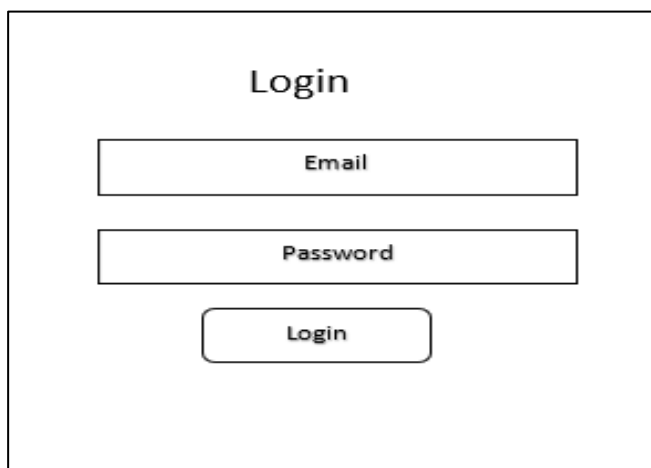
4.5.1-INPUT DESIGN

Form Name : User Registration



The User Registration form is a vertical layout within a rectangular frame. At the top, the title "Register" is centered. Below it, there are six input fields stacked vertically: "First Name", "Last Name", "Select Gender", "Email", "Password", and "Confirm Password". Each field is a simple rectangle with its label inside. At the bottom of the form is a rounded rectangular button labeled "Register".

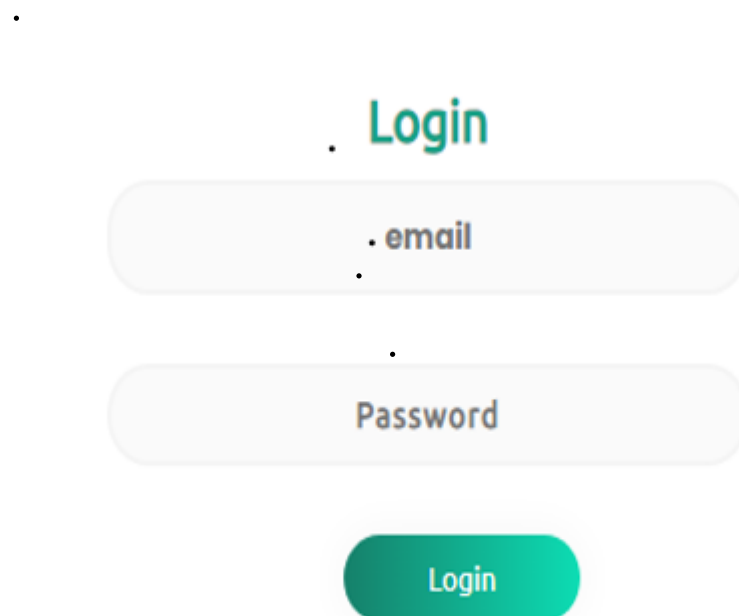
Form Name : Login



The Login form is a vertical layout within a rectangular frame. At the top, the title "Login" is centered. Below it, there are two input fields stacked vertically: "Email" and "Password". Each field is a simple rectangle with its label inside. At the bottom of the form is a rounded rectangular button labeled "Login".

4.5.2 OUTPUT DESIGN

Login



A login form with a light gray background. At the top, the word "Login" is written in a teal color. Below it, there are two light gray rounded rectangular input fields. The first field contains the text "email" in a dark gray font, and the second field contains the text "Password" in a dark gray font. Below these fields is a teal rounded rectangular button with the word "Login" in white text.

User Registration

Register



4.6 DATABASE DESIGN

A database is an organized system capable of storing information from which a user can efficiently and effectively retrieve the stored information. The purpose of any database is data, which must be preserved.

Database design is a two-level process. In the first step, user requirements are gathered together and a database is designed that meets these requirements as clearly as possible. This step is called information level design and is independent of any individual DBMS.

In a second step, this information level design is converted into a design for the specific DBMS that will be used to implement the system in question. This step is called physical level design and is concerned with the specifications of the specific DBMS to be used. A database design works in parallel with system design. The organization of data in the database aims to achieve the following two main objectives.

- Data Integrity
- Data independence

4.6.1 Non Relational Database Management System

A non-relational database is one that does not use the tabular schema of rows and columns of common standard database systems. Non-relational databases, on the other hand, use a storage model tailored to the specific needs of the data being stored. For example, data may be stored as direct key/value pairs or as JSON documents or as a network of edges and vertices. Investigates weighted connections between entities. Neither format works well for handling transactional data in general.

Data stores that do not use SQL queries are called NoSQL. Instead, data stores query data using various computer languages and techniques. Despite the fact that many of these databases enable SQL-compatible queries, "NoSQL" is generally used to refer to "non-relational databases".

Although the fundamental query execution approach is frequently significantly different from how the identical SQL query would be handled by a conventional RDBMS. Every value in a relation is atomic, that is not decomposable.

NoSQL

Unlike relational databases, NoSQL databases store data in documents. Accordingly, we categorize them as "not just SQL" and divide them by various flexible data models. Pure document databases, key-value stores, wide-column databases, and graph databases are some examples of NoSQL database types. NoSQL databases are designed from the ground up for a growing number of modern enterprises.

Instead of the columns and rows that relational databases use to store data, NoSQL database technology stores data in JSON documents. Strictly speaking, NoSQL does not mean "no SQL" but "not only SQL". This implies that a NoSQL JSON database "doesn't need to use SQL" to store and retrieve data. Or, for the best of both worlds, you can combine the adaptability of JSON with the power of SQL. For this reason, NoSQL databases are designed to be flexible, scalable, and fast to meet the data management needs of contemporary enterprises. The four most common types of NoSQL databases are described as follows:

- **Document databases:**The main purpose of document databases is to store data as documents, including but not limited to JSON documents. XML documents can also be stored using these technologies.
- **Key-value stores :**For quick retrieval, key-value storing organises group-related data into collections of records with unique keys. Key-value stores provide the advantages of NoSQL while yet having just enough structure to replicate the value of relational databases.
- **Wide-column databases:**Even within the same table, wide-column databases allow for a great deal of variation in how data is titled and presented in each row. Wide-column databases, like key-value stores, provide some fundamental structure while simultaneously retaining a great deal of flexibility.
- **Graph databases:**The relationships between the data points recorded in graph databases are defined by the graph structures. For finding patterns in unstructured and semi-structured data, graph databases are helpful.

JSON Tree

The data in the Firebase Realtime Database is all kept as JSON objects. The database can be compared to a JSON tree maintained in the cloud. There are no tables or records, in contrast to a SQL database. When you add data to the JSON tree, it is transformed into a node with a key in the already-existing JSON structure.

TABLE DESIGN**Users**

Field Name	Datatype	Description
email	String	Email of user
name	String	Name of user
phMain	String	Phone number of user
status	String	Status of user(offline ,online)
u_active	String	Field for showing the user is active or not
user_type	String	Describes user type of user(admin,doctor,user)
verf	String	Field for showing the doctor is verified or not.

User_data

Field Name	Datatype	Description
email	String	Email of user
name	String	Name of user
phMain	String	Phone number of user
user_pic	String	Image of doctor

Doctor_data

Field Name	Datatype	Description
email	String	Email of user
desc	String	Description of user
doc_pic	String	Image of doctor
experience	String	Experience of doctor
fees	String	Fees for doctor consultation
gender	String	Gender of the doctor
name	String	Name of the doctor
type	String	Speciality of doctor
verification	String	Field shows the verification of doctor

Patient_Details

Field Name	Datatype	Description
pemail	String	Email of user
name	String	Name of the patient

Chats

Field Name	Datatype	Description
receiver	String	Email of receiver user
sender	String	Email of sender user
seen	Boolean	Description about seen true or false
time	String	Time of message send
type	String	Type of message sen text or image

Doctor_Chosen_Slots

Field Name	Datatype	Description
email	String	Email of doctor
date	String	Date of slot for booking
time	String	Time of slot about date
pemail	String	Email of user
value	Int	Value 0 or 1

Doctors_Feedback

Field Name	Datatype	Description
email	String	Email of doctor
pemail	String	Email of user
date	String	Date of feedback
time	String	Time of feedback
feedback	String	Feedback about doctor
rating	Int	Rating about doctor

Patient_Chosen_Slots

Field Name	Datatype	Description
Patient_name	String	Email of user
question	String	Description of user
flag	int	Description of flag value
pemail	String	Email of user
name	String	Description of user

Prescription_by_doctor

Field Name	Datatype	Description
age	String	Age of user
consultationType	String	Description of user
<u>consultation_date</u>	String	Description of consultation date
doctorAddress	String	Email of user
name	String	Description of user name or patient
prescription	String	Description of prescription for patient by doctor

Doctor_Appointments

Field Name	Datatype	Description
pname	String	Name of patient
question	String	Description of question of patient
<u>consultation_date</u>	String	Description of consultation date
time	String	Time of appointment
token	String	Description of token given to patient
prescription	String	Description of prescription for patient by doctor

Disease_prediction

Field Name	Datatype	Description
did	int	Prediction id
age	String	Email of user
approvalstatus	int	Description of approval 0 and 1
discategory	String	Description of disease category
pemail	String	Email of user
Predicted_disease	String	Description of predicted disease of patient
symptoms	String	Symptoms entered by the users

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is the process of executing software in a controlled manner to answer the question – Does the software work as specified? Software testing is often used in conjunction with the terms verification and validation. Validation is the examination or examination of items comprising software for conformance and consistency to a related specification. Software testing is just one type of verification that also uses techniques such as reviews, analysis, tests, and walkthroughs. Validation is the process of checking what the user actually wants.

Other activities often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the software's source code, looks for problems, and collects metrics without actually executing the code. Dynamic analysis looks at the behavior of software as it executes to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activities that can be extensively planned and systematically carried out. Testing starts at the module level and works towards integration of the entire computer-based system. Nothing is complete without testing, as it is the vital achievement of system testing objectives, there are many rules that serve as testing objectives. They are:

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

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a testing is successfully performed according to the above objectives, it will detect errors in the software. Testing demonstrates that the software functions according to the specification and that the performance requirements are seen to be met. There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

5.2 TEST PLAN

A test plan refers to a series of activities to be followed in implementing various testing methods. The test plan acts as a blueprint of the action to follow. Software engineers create a computer program, its documentation, and associated data structures. Software developers are always responsible for testing individual units of programs to ensure that each one performs the function it was designed to do. There is an Independent Test Group (ITG) which removes the inherent problems associated with allowing the builder to test the built thing. The specific objectives of the test should be stated in measurable terms. Therefore, the mean time to failure, the cost of finding and fixing defects, the density or frequency of occurrence of remaining defects, and the work time in a regression test should all be specified in the test plan.

The levels of testing include:

- ❖ Unit testing
- ❖ Integration Testing
- ❖ Data validation Testing
- ❖ Output Testing

5.2.1 Unit Testing

Unit testing focuses the verification effort on the smallest unit of software design, the software component or module. Using the component-level design description as a guide, critical control paths are examined to detect errors within the module's boundaries. Relative complexity of tests and uncovered scope established for unit testing. Unit testing is white-box oriented and can be phased in parallel for multiple components. The modular interface is tested to ensure that information flows correctly to and from the program unit under test. The local data structure is checked to ensure that the stored data temporally maintains its integrity at every step in the execution of an algorithm. Boundary conditions are checked to ensure that all statements in a module are executed at least once. Finally, all error handling paths are tested. Tests of data flow across the module interface are required before starting any other test. If data is not entered and exited correctly, all other checks are problematic. Selective testing of execution paths is an essential task during unit testing. Good design suggests anticipating error conditions and setting up error handling paths to

reorder or cleanly terminate processing when an error occurs. The final task of the unit testing phase is boundary testing. Software often fails at its limits.

Unit testing in the Cell-Soft system was performed by treating each module as a separate entity and testing each of them with a broad spectrum of test inputs. Some errors in the internal logic of the modules were detected and corrected. Each module is individually tested and run after being coded. All unnecessary code is removed and all modules are made to work and deliver the expected results.

Test Case 1

Project Name: V-Care					
Registration Test Case					
Testcase ID: Testcase1			Test Designed By: Ashish Wilson		
Test Priority(Low/Medium/High):High			Test Designed Date: 18-07-2022		
Module Name: Register page			Test Executed By :Mr Ajith G.S		
Test Title : Register the user with email and password			Test Execution Date: 18-07-2022		
Description: Test the Registration Page					
Pre-Condition :User has valid email id and contact number and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate To Register page		Register page should be displayed	Register page displayed	Pass
2	Provide Valid Email Id	Email: ashishvwilson@gmail.com	User should be able to Register and go to login page	User Registered and went to login page	Pass
3	Provide 10 digit phone number	Number: 9878945156			
4	Provide Valid Password	Password: 12345678 Confirm Password: 12345678			
5	Click on Sign Up button				

6	Provide Invalid Email Id or password or contact number	Email Id: ashishvwilson@gmail.com Password: 123456 Confirm password: 123456 Number: 9495493948	User shouldn't register And redirect back to register page.	User didn't register as email already exists from previous registration. Invalid email, number etc can also cause error messages	Pass
7	Provide Null Email, Password, Name, Number	Email Id: null Password: null	User shouldn't register And redirect back to register page.	User didn't register as email already exists from previous registration. Invalid email, number etc can also cause error messages	Pass
8	Click on LogIn button				
Post-Condition: User is validated and successfully registered and can now login.					

Code:

```
package com.example.vcare;

import org.junit.Assert;
import org.junit.Test;
import com.example.vcare.register.Signup;

public class registerTest {
    @Test
    public void testExampleWithCorrectValues() {
        String lname = "anu"; String emailMain = "ashish123@gmail.com"; String
        phMain = "9878945156"; String passwordMain = "user1234"; String cpasswordMain
        = "user1234";
        boolean response =
        Signup.validate(lname, emailMain, phMain, passwordMain, cpasswordMain);
        Assert.assertEquals(true, response);
    }
}
```

```

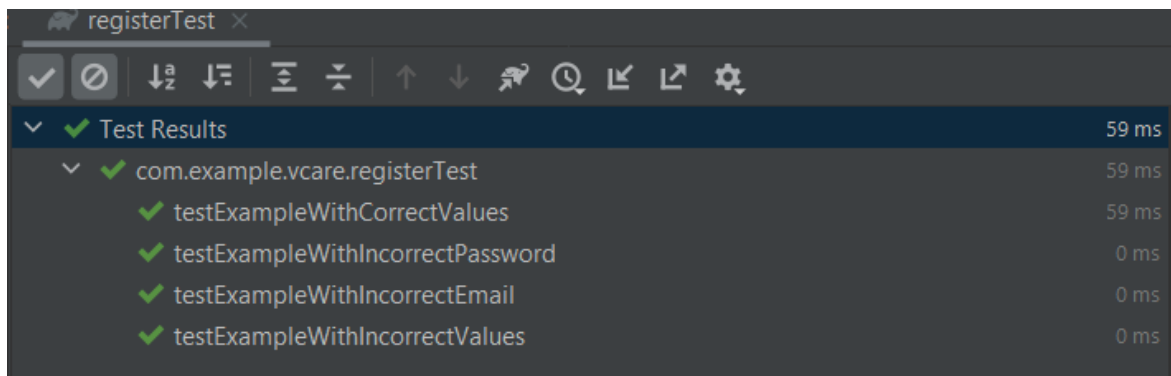
@Test
public void testExampleWithIncorrectEmail() {
    String lname = ""; String emailMain = "user@gmail.com"; String phMain = ""; String
passwordMain = "user"; String cpasswordMain = "user";
    boolean response =
Signup.validate(lname, emailMain, phMain, passwordMain, cpasswordMain);
    Assert.assertEquals(false, response);
}

@Test
public void testExampleWithIncorrectPassword() {
    String lname = "anu"; String emailMain = "user@gmail.com"; String
phMain = "9878945156"; String passwordMain = ""; String cpasswordMain = "";
    boolean response =
Signup.validate(lname, emailMain, phMain, passwordMain, cpasswordMain);
    Assert.assertEquals(false, response);
}

@Test
public void testExampleWithIncorrectValues() {
    String lname = "anu"; String emailMain = "use123r@gmail.com"; String
phMain = "9878945156"; String passwordMain = ""; String cpasswordMain = "";
    boolean response =
Signup.validate(lname, emailMain, phMain, passwordMain, cpasswordMain);
    Assert.assertEquals(false, response);
}
}

```

Test Result



✓ Tests passed: 4 of 4 tests – 59 ms

Executing tasks: [:app:testDebugUnitTest] in project [D:\Mainproject_2022](#)

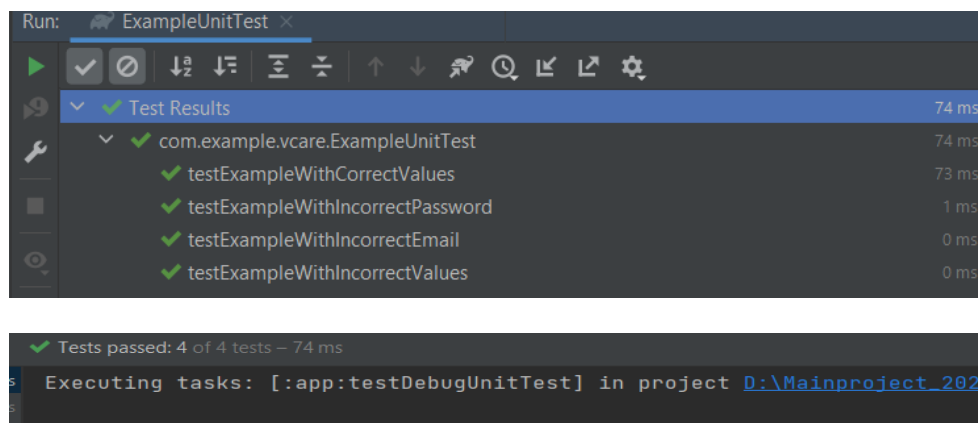
TestCase 2

Project Name: V-Care					
Login Test Case					
Test Case ID: Test_2			Test Designed By: Ashish Wilson		
Test Priority(Low/Medium/High):High			Test Designed Date: 18-07-2022		
Module Name: Login			Test Executed By :Mr Ajith G.S		
Test Title : Login with email and password			Test Execution Date: 18-07-2022		
Description: Test the Login Page					
Pre-Condition :User has valid email id and contact number and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to Login page		Login page should be displayed	Login page displayed	Pass
2	Provide Valid Email Id	Email: ashishvwilson@gmail.com	User should be able to login	User logged in and navigation to dashboard	Pass
5	Provide Valid Password	Password: 12345678			
6	Click on Log In button				
5	Provide Invalid Email Id or password	Email Id: ashishvwilson@gmail.com Password: 12345678	User shouldn't Login And redirect back to Login page.	Message for invalid password and email (Toast message)	Pass
6	Provide Null Email, Password, Name,Number	Email Id: null Password:null			
7	Click on Sign In button				
Post-Condition: User has been validated and successfully and logged into the account.					

Code:

```
package com.example.vcare;
import org.junit.Assert;
import org.junit.Test;
import static org.junit.Assert.*;
import com.example.vcare.register.Login;
public class ExampleUnitTest {
    @Test
    public void testExampleWithCorrectValues() {
        String validEmail = "ashish123@gmail.com";
        String validPassword = "user123456";
        boolean response = Login.validate(validEmail,validPassword);
        Assert.assertEquals(true, response);
    }
    @Test
    public void testExampleWithIncorrectEmail() {
        String invalidEmail = "email1";
        String validPassword = "password";
        boolean response = Login.validate(invalidEmail,validPassword);
        Assert.assertEquals(false, response);
    }
    @Test
    public void testExampleWithIncorrectPassword() {
        String validEmail = "email";
        String invalidPassword = "password1";
        boolean response = Login.validate(validEmail,invalidPassword);
        Assert.assertEquals(false, response);
    }
    @Test
    public void testExampleWithIncorrectValues() {
        String invalidEmail = "email1";
        String invalidPassword = "password1";
        boolean response = Login.validate(invalidEmail,invalidPassword);
        Assert.assertEquals(false, response);
    }
}
```

TestResult:



5.2.2 Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

5.2.3 Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests. Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

5.2.1 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.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to

ensure that the resistance does not build up, as one has to make sure that:

- The active user must be aware of the benefits of using the new system.
- Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using 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

User training is designed to prepare the user to test and convert the system. In order to achieve the goal and benefits expected from the computer-based system, it is essential that the people involved have confidence about their role in the new system. The more complex the system, the more important the need for training. Through user training, the user knows how to enter data, respond to error messages, query the database, produce reports, and perform routine call ups to perform other required functions..

6.2.2 Training on the Application Software

After imparting the necessary basic training in computer awareness, the user will have to be trained in the new application software. It will provide the basic philosophy of using the new system like screen flow, screen design type of help on screen, type of errors in data entry, validation check associated with each entry and ways to correct date. Entered It should cover the information required by the specific user/group to use the system or part of the system while training the program in the application. This training may be different for different user groups and at different levels of hierarchy.

6.2.3 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In this project, we predict the disease by considering the symptoms given by the user. In this project, prediction of disease based on user and user given symptoms can book appointment with a doctor under that particular disease. After booking the appointment the user can chat with the doctors and the doctor can upload the prescription to the user. Although we have tested our system for viral and routine-based diseases, it can be extended to larger settings. This app has a huge scope as it has the following features:

- Automation of Disease Prediction
- To save the environment by using paper free work
- To increase the accuracy and efficiency so that patients can get direct help.
- Management of disease related data
- It will be useful in urgent cases where patient is unable to reach doctor, during late night emergencies
- It is easy to handle
- Booking of appointments after disease prediction
- We can ask queries with doctors with the help of real-time chat
- Both users and doctors can read latest health news

7.2 FUTURE SCOPE

Future work can be focused on considering different types of symptoms in predicting more symptoms with help of a body map which can be helpful to users so it will be more user friendly and users are not required to enter symptoms which they are not sure. Test results for various medical conditions will help improve the reliability of the system. Also by helping users with top rated doctors user can see review of the doctors and book appointment with them also videochat can also be enabled so user can chat with doctors in realtime.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009.
- Roger S Pressman, “*Software Engineering*”, 1994.
- PankajJalote, “*Software engineering: a precise approach*”, 2006.
- “*Native Mobile Development: A Cross-Reference for IOS and Android Book*” by Mike Dunn and Shaun Lewis,2015
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- www.w3schools.com
- <https://developer.android.com/training/basics>
- <http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf>
- www.agilemodeling.com/artifacts/useCaseDiagram.html

CHAPTER 9

APPENDIX

9.1 Sample Code

Login.java

```
package com.example.vcare.register;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Patterns;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.example.vcare.R;
import com.example.vcare.admin.Admin_Dashboard;
import com.example.vcare.doctor.Doctor_Email_Id;
import com.example.vcare.doctor.Doctors;
import com.example.vcare.doctor.Doctors;
import com.example.vcare.doctor.Session_Mangement;
import com.example.vcare.model.Patient_email_id;
import com.example.vcare.patient.Patient;
import com.example.vcare.patient.Patient_Session_Management;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.EmailAuthProvider;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
```



```
public class Login extends AppCompatActivity implements View.OnClickListener {
    private TextView register, forgotPassword, newuser;
    private EditText editTextEmailMain, editTextPasswordMain;
    private Button signIn;
    private int flag = 0;
    private FirebaseAuth myAuth;
    private String encoded_email;
    private ProgressBar progressBar;
    private DatabaseReference databaseReference;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        signIn = (Button) findViewById(R.id.loginButton);
        signIn.setOnClickListener(this);
        editTextEmailMain = (EditText) findViewById(R.id.emailMain);
        editTextPasswordMain = (EditText) findViewById(R.id.passwordMain);
        progressBar = (ProgressBar) findViewById(R.id.progressbarlogin);
        myAuth = FirebaseAuth.getInstance();
        forgotPassword = (TextView) findViewById(R.id.forgotPasswordText);
        forgotPassword.setOnClickListener(this);
        newuser = (TextView) findViewById(R.id.newuser);
        newuser.setOnClickListener(this);
        databaseReference = FirebaseDatabase.getInstance("https://vcare-healthapp-default-
rtdb.asia-southeast1.firebaseio.com").getReference("Users");
        checkDoctorSession();
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.loginButton:
                userLogin();
                break;
            case R.id.forgotPasswordText:
                startActivity(new Intent(this, ForgotPassword.class));
                break;
            case R.id.newuser:
                startActivity(new Intent(this, Signup.class));
                break;
        }
    }
    private void userLogin() {
        String emailMain = editTextEmailMain.getText().toString().trim();
```

```

String passwordMain = editTextPasswordMain.getText().toString().trim();
encoded_email = EncodeString(emailMain);
if (emailMain.isEmpty()) {
    editTextEmailMain.setError("Email is a required field !");
    editTextEmailMain.requestFocus();
    return;
}
if (!Patterns.EMAIL_ADDRESS.matcher(emailMain).matches()) {
    editTextEmailMain.setError("Please provide Valid Email !");
    editTextEmailMain.requestFocus();
    return;
}
if (passwordMain.isEmpty()) {
    editTextPasswordMain.setError("Password is a required field !");
    editTextPasswordMain.requestFocus();
    return;
}
if (passwordMain.length() < 6) {
    editTextPasswordMain.setError("Minimum length of password should be 6
characters !");
    editTextPasswordMain.requestFocus();
    return;
}
progressBar.setVisibility(View.VISIBLE);
myAuth.signInWithEmailAndPassword(emailMain,
passwordMain).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {

        if (task.isSuccessful()) {
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
            if (user.isEmailVerified()) {
                databaseReference.child(encoded_email).addValueEventListener(new
ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot dataSnapshot) {
                        if (dataSnapshot.exists()) {
                            String status = dataSnapshot.child("u_active").getValue(String.class);
                            if (user != null) {
                                if (status.equals("1"))
                                    {
                                        String usertype =
dataSnapshot.child("user_type").getValue(String.class);
                                        if (usertype.equals("Doctor")) {

```

```

        flag=0;
        /*if (flag == 0) {
            showChangePasswordDialog();
            progressBar.setVisibility(View.INVISIBLE);
        }*/
        Doctor_Email_Id doctor_email_id = new
Doctor_Email_Id(emailMain, "Doctor");
        Session_Mangement _session_mangement = new
Session_Mangement(Login.this);
        _session_mangement.saveDoctorSession(doctor_email_id);
        startActivity(new Intent(Login.this, Doctors.class));
        progressBar.setVisibility(View.INVISIBLE);
    } else if (usertype.equals("user")) {

        Doctor_Email_Id doctor_email_id = new
Doctor_Email_Id(emailMain, "user");
        Session_Mangement _session_mangement = new
Session_Mangement(Login.this);
        _session_mangement.saveDoctorSession(doctor_email_id);
        Patient_email_id patient = new
Patient_email_id(encoded_email);
        Patient_Session_Management session_management = new
Patient_Session_Management(Login.this);
        session_management.saveSession(patient);
        startActivity(new Intent(Login.this, Patient.class));
        progressBar.setVisibility(View.INVISIBLE);
    } else {
        Doctor_Email_Id doctor_email_id = new
Doctor_Email_Id(emailMain, "Admin");
        Session_Mangement _session_mangement = new
Session_Mangement(Login.this);
        _session_mangement.saveDoctorSession(doctor_email_id);
        startActivity(new Intent(Login.this, Admin_Dashboard.class));
        progressBar.setVisibility(View.INVISIBLE);
    }
}
else
{
    Toast.makeText(Login.this, " Account Disabled ",
Toast.LENGTH_SHORT).show();
}
}
}

```

```
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
    }
});
progressBar.setVisibility(View.INVISIBLE);

} else if (flag == 0) {
    databaseReference.child(encoded_email).addValueEventListener(new
ValueEventListener() {

        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            String usertype = snapshot.child("user_type").getValue(String.class);
            if(usertype.equalsIgnoreCase("Doctor")) {
                showChangePasswordDialog();
                progressBar.setVisibility(View.INVISIBLE);
            } else {
                user.sendEmailVerification();
                flag = 3;
                progressBar.setVisibility(View.INVISIBLE);
            }
        }

    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
}
else if(flag==2)
{
    progressBar.setVisibility(View.INVISIBLE);

}
else
{
    user.sendEmailVerification();
    progressBar.setVisibility(View.INVISIBLE);
    startActivity(new Intent(DoctorsLogin.this, ForgotPassword.class));
    Toast.makeText(Login.this, "Check your Email to verify your account",
Toast.LENGTH_LONG).show();
```

```

        }
    } else {
        Toast.makeText(Login.this, "Failed to Login ! Please check your credentials",
            Toast.LENGTH_LONG).show();
        progressBar.setVisibility(View.INVISIBLE);
    }
}
});
}

private void showChangePasswordDialog() {
    View view = LayoutInflater.from(Login.this).inflate(R.layout.update_password, null);
    final EditText passwordold = view.findViewById(R.id.passwordEt);
    final EditText passwordnew = view.findViewById(R.id.newpasswordEt);
    Button updatepass = view.findViewById(R.id.updatePasswordBtn);
    final AlertDialog.Builder builder = new AlertDialog.Builder(Login.this);
    builder.setView(view);
    AlertDialog dialog = builder.create();
    dialog.show();
    updatepass.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String oldpass = passwordold.getText().toString().trim();
            String newpass = passwordnew.getText().toString().trim();
            if (TextUtils.isEmpty(oldpass)) {
                Toast.makeText(Login.this, "Enter Your Current Password!",
                    Toast.LENGTH_SHORT).show();
                return;
            }
            if (newpass.length() < 6) {
                Toast.makeText(Login.this, "Password Length must have atleast 6 characters!",
                    Toast.LENGTH_SHORT).show();
                return;
            }
            dialog.dismiss();
            flag = 1;
            updatePassword(oldpass, newpass);
        }
    });
}

private void updatePassword(String oldpass, String newpass) {
    FirebaseUser user = myAuth.getCurrentUser();
    AuthCredential authCredential = EmailAuthProvider.getCredential(user.getEmail(),

```

```

oldpass);
    user.reauthenticate(authCredential)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                user.updatePassword(newpass)
                    .addOnSuccessListener(new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Toast.makeText(Login.this, "Password Updated!",
                                Toast.LENGTH_SHORT).show();
                        }
                    })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(Login.this, "" + e.getMessage(),
                            Toast.LENGTH_SHORT).show();
                    }
                });
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(Login.this, "" + e.getMessage(),
                    Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void checkDoctorSession() {

        Session_Mangement _session_mangement =new Session_Mangement(Login.this);
        String isDoctorLoggedIn[] = _session_mangement.getDoctorSession();
        if(!isDoctorLoggedIn[0].equals("-1")){
            moveToDoctorActivity();
        }

    }

    private void moveToDoctorActivity() {
        Session_Mangement _session_mangement = new Session_Mangement(Login.this);

```

```

        String type[] = _session_mangement.getDoctorSession();
        if(type[1].equals("Doctor")){
            Intent intent = new Intent(Login.this, Doctors.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
        }
        else if (type[1].equals("Admin")){
            Intent intent = new Intent(Login.this, Admin_Dashboard.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
        }
        else if (type[1].equals("user")){
            Intent intent = new Intent(Login.this, Patient.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
        }
    }
    public static String EncodeString(String string) {
        return string.replace(".", ",");
    }
}

```

PatientDashboard.java

```

package com.example.vcare.predictor;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.content.res.AssetFileDescriptor;
import android.content.res.AssetManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Spinner;
import android.widget.TextView;

```

```

import android.widget.Toast;
import org.tensorflow.lite.Interpreter;
import com.example.vcare.R;
import com.example.vcare.patient.Patient_Session_Management;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
public class PatientDashboard extends AppCompatActivity{
    EditText ename, eage;
    String patemail;
    String user_name;
    TextView vali;
    Spinner spin1;
    RadioButton r1,r2,r3,r4,r5;
    RadioGroup radio_gender;
    private DatabaseReference reference_user_details;
    Interpreter interpreter;
    List<String> symList = new ArrayList<String>();
    int i = 0;
    String[] selected={ };
    String[] all_symptoms = { };
    float[] input = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

```



```

String[] predict_Diseases = { "Fungal infection", "Allergy", "GERD", "Chronic
cholestasis", "Drug Reaction",
    "Peptic ulcer disease", "AIDS", "Diabetes ", "Gastroenteritis", "Bronchial
Asthma",
    "Hypertension ", "Migraine", "Cervical spondylosis", "Paralysis (brain
hemorrhage)",
    "Jaundice", "Malaria", "Chicken pox", "Dengue", "Typhoid", "hepatitis A",
"Hepatitis B",
    "Hepatitis C", "Hepatitis D", "Hepatitis E", "Alcoholic hepatitis", "Tuberculosis",
    "Common Cold", "Pneumonia", "Dimorphic hemmorhoids(piles)", "Heart attack",
    "Varicose veins", "Hypothyroidism", "Hyperthyroidism", "Hypoglycemia",
    "Osteoarthritis", "Arthritis", "(vertigo) Paroymsal Positional Vertigo",
    "Acne", "Urinary tract infection", "Psoriasis", "Impetigo"};

String[] disease_category = { "Dermatology & Venereology", "Dermatology &
Venereology", "Gastroenterology, Hepatology & Endoscopy", "General Medicine",
    "Dermatology & Venereology", "Immunology", "Endocrinology & Diabetes",
"Gastroenterology, Hepatology & Endoscopy", "General Medicine",
    "General Medicine", "Ear Nose Throat", "Gastroenterology, Hepatology &
Endoscopy", "Ear Nose Throat",
    "General Medicine", "General Medicine", "General Medicine", "General
Medicine", "General Medicine", "General Medicine", "General Medicine",
    "General Medicine", "General Medicine", "General Medicine", "General
Medicine", "General Medicine",
    "General Medicine", "General Medicine", "Gastroenterology, Hepatology &
Endoscopy", "Gastroenterology, Hepatology & Endoscopy",
    "Gastroenterology, Hepatology & Endoscopy", "Ear Nose Throat", "Ear Nose
Throat", "Ear Nose Throat",
    "Gastroenterology, Hepatology & Endoscopy", "Dermatology & Venereology",
"Gastroenterology, Hepatology & Endoscopy",
    "Hormone", "General Medicine", "Dermatology & Venereology", "Dermatology &
Venereology"};

Map<Integer, List<String>> resultMap = new HashMap<Integer, List<String>>();
List<String> selectedSymList = new ArrayList<>();
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_patient_dashboard);
    ename = findViewById(R.id.edit_name);
    eage = findViewById(R.id.edit_age);
    radioButton = (RadioButton) findViewById(selectedId);
    radioButton = (RadioButton) findViewById(selectedId);
    radio_gender=findViewById(R.id.radio_gender);
    vali=findViewById(R.id.vali);

```

```
user_name = ename.getText().toString();
String user_age = eage.getText().toString();
try {
    AssetManager manager = getAssets();
    InputStream in = manager.open("Training.csv");
    resultMap = parse(in);

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
reference_user_details = FirebaseDatabase.getInstance("https://vcare-healthapp-
default-rtdb.asia-southeast1.firebaseio.com").getReference("User_data");
Patient_Session_Management session = new
Patient_Session_Management(PatientDashboard.this);
patemail = session.getSession();
reference_user_details.child(patemail).addListenerForSingleValueEvent(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (snapshot.exists()) {
            ename.setText(snapshot.child("name").getValue(String.class));
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
try {
    interpreter = new Interpreter(loadModelFile(), null);
}
catch (IOException e) {
    e.printStackTrace();
}

spin1 = (Spinner) findViewById(R.id.spinner1);
spin1.setOnItemSelectedListener(this);
Spinner spin2 = (Spinner) findViewById(R.id.spinner2);
spin2.setOnItemSelectedListener(this);
Spinner spin3 = (Spinner) findViewById(R.id.spinner3);
spin3.setOnItemSelectedListener(this);
Spinner spin4 = (Spinner) findViewById(R.id.spinner4);
```

```

        spin4.setOnItemClickListener(this);
        selected=all_symptoms;
        ArrayAdapter aa1 = new ArrayAdapter(this, android.R.layout.simple_spinner_item,
all_symptoms);
        aa1.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spin1.setAdapter(aa1);
        System.out.println(spin1);
        String[] symp2 =null;
        String txt ="";
        spin1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
                String sym = parent.getItemAtPosition(position).toString();
                selectedSymList.add(sym);
                if(!sym.equalsIgnoreCase("Select a symptom")){
                    symList.add(sym);
                }
                int l =0;
                input[position] = 1;
                List<String> arrayList = new ArrayList<>();
                arrayList = displaySym(sym,arrayList);
                String[] array = new String[arrayList.size()+1];
                array[0] = "Select a symptom";
                for (int i = 0; i < arrayList.size(); i++) {
                    array[i+1] = arrayList.get(i);
                }
                Log.d("main", "Main function 1");
                ArrayAdapter aa2 = new ArrayAdapter(parent.getContext(),
android.R.layout.simple_spinner_item, array);
                aa2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                spin2.setAdapter(aa2);
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {

                Log.i("GTOUTOUT", "Nothing Selected");
            }
        });
        spin2.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {

```

```

        String sym = parent.getItemAtPosition(position).toString();
        selectedSymList.add(sym);
        if(!sym.equalsIgnoreCase("Select a symptom")){
            symList.add(sym);
        }
        for(int i=0;i<all_symptoms.length;i++){
            if(all_symptoms[i].equals(sym)){
                input[i] = 1;
            }
        }
        List<String> spinner2list = new ArrayList<>();
        spinner2list = displaySym(sym, spinner2list);
        String[] array2 = new String[spinner2list.size() + 1];
        array2[0] = "Select a symptom";
        if (sym != null) {
            if (!spinner2list.contains(sym)) {
                for (int i = 0; i < (spinner2list.size()); i++) {
                    array2[i + 1] = spinner2list.get(i);
                }
            }
        }
        Log.d("main", "Main function 2");
        ArrayAdapter aa3 = new ArrayAdapter(parent.getContext(),
        android.R.layout.simple_spinner_item, array2);

        aa3.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spin3.setAdapter(aa3);
        Log.d("main", "Main function 3");
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        Log.i("GTOUTOUT", "Nothing Selected");
    }
});
if (parent.getId() == R.id.spinner3) {
    spin3.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int p,
long l) {
            String sym = adapterView.getItemAtPosition(p).toString();
            selectedSymList.add(sym);
            if(!sym.equalsIgnoreCase("select a symptom")){
                symList.add(sym);
            }

```

```

        for(int i=-0;i<all_symptoms.length;i++){
            if(all_symptoms[i].equals(sym)){
                input[i] = 1;
            }
        }
        List<String> spinner3list = Arrays.asList(array2);
        List<String> spinner3list = new ArrayList<>();
        spinner3list = displaySym(sym, spinner3list);
        String[] array3 = new String[spinner3list.size() + 1];
        array3[0] = "Select a symptom";
        if (sym != null) {
            if (!spinner3list.contains(sym)) {
                for (int i = 0; i < (spinner3list.size()); i++) {
                    array3[i + 1] = spinner3list.get(i);
                }
            }
        }
        ArrayAdapter aa4 = new ArrayAdapter(adapterView.getContext(),
        android.R.layout.simple_spinner_item, array3);

        aa4.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spin4.setAdapter(aa4);
        spin4.setOnItemClickListener(this);
        Log.d("main", "Main function 4");
    }
    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {

    }

});
spin4.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long
1) {
        String sym = adapterView.getItemAtPosition(i).toString();
        if(!sym.equalsIgnoreCase("select a symptom")){
            symList.add(sym);
        }
        for(int k=-0;k<all_symptoms.length;k++){
            if(all_symptoms[k].equals(sym)){
                input[k] = 1;
            }
        }
    }
}

```

```

        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {
        }
    });
}

private List<String> displaySym(String sym, List<String> arrayList) {
    arrayList = new ArrayList<>();
    for(int i=0; i<resultMap.size();i++){
        int i=0;
        while(i <resultMap.size() ){
            List<String> values = resultMap.get(i++);
            if(values.contains(sym)){
                for(int k=0;k<values.size();k++){

                    if(!values.get(k).equals(sym)){
                        if(!arrayList.contains(values.get(k)) &&
!selectedSymList.contains(values.get(k))) {
                            arrayList.add(values.get(k));
                        }
                    }
                }
            }
        }
    }
    return arrayList;
}

private Map<Integer, List<String>> parse(InputStream in) throws IOException {
    Map<Integer, List<String>> results = new HashMap<>();
    int k = 0;
    String value = "";
    String[] symArray = null;
    BufferedReader reader = new BufferedReader(new InputStreamReader(in));
    String nextLine = null;
    int m =0;
    while ((nextLine = reader.readLine()) != null) {
        String[] tokens = nextLine.split(" ");
        if(m != 0) {
            List<String> valuesList = new ArrayList<>();
            for (int i = 0; i < tokens.length-2; i++) {
                if (tokens[i].equals("1")) {
                    valuesList.add(symArray[i]);
                }
            }
            results.put(k++, valuesList);
        }
    }
}

```

```

        value = "";
    }else{
        symArray= nextLine.split("\n");
    }
    m++;
}
in.close();
return results;
}
private MappedByteBuffer loadModelFile() throws IOException {
    AssetFileDescriptor assetFileDescriptor = this.getAssets().openFd("vcare.tflite");
    FileInputStream fileInputStream = new
FileInputStream(assetFileDescriptor.getFileDescriptor());
    FileChannel fileChannel = fileInputStream.getChannel();
    long startOffset = assetFileDescriptor.getStartOffset();
    long length = assetFileDescriptor.getLength();
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, length);
}
public float[][] doInference(float[][] val) {
    Log.d("doInference", "doInference method called");
    float[][] output = new float[1][41];
    interpreter.run(val, output);
    return output;
}
private static int findMax(float[][] matrix) {
    Log.d("findMax", "max method called");
    float maxNum = matrix[0][0];
    int k = 0;
    for (int j = 0; j < matrix[0].length; j++) {
        if (maxNum < matrix[0][j]) {
            maxNum = matrix[0][j];
            k = j;
        }
    }
    Log.d("findMax", "k value is" + k);
    return k;
}
public void predict(View view) {
    String lname = ename.getText().toString().trim();
    String age = eage.getText().toString().trim();
    if (lname.isEmpty()) {
        ename.setError("Name is a required field !");
        ename.requestFocus();
        return;
    }

```

```

    }
    if (age.isEmpty()) {
        eage.setError("Age is a required field !");
        eage.requestFocus();
        return;
    }
    if (radio_gender.getCheckedRadioButtonId() == -1)
    {
        vali.setText("Please enter your gender");
        return;
    }
    if (radio_gender.getCheckedRadioButtonId() != -1)

    {
        vali.setText("");
    }
    if (spin1.getSelectedItem().toString().trim().equals("select a symptom")) {
        Toast.makeText(PatientDashboard.this, "please select atleast 2 symptoms",
        Toast.LENGTH_SHORT).show();
    }
    Log.d("predict", "Predict method called");
    float[][] finalinput = {input};
    Log.d("predict", "finalinput" + Arrays.deepToString(finalinput));

    float[][] out = doInference(finalinput);

    //out=[[41 values]]
    int element = findMax(out);

    //create a text view to display
    Log.d("predict", "k value returned,element" + predict_Diseases[element]);

    Toast.makeText(getApplicationContext(), predict_Diseases[element],
    Toast.LENGTH_LONG).show();
    for (i = 0; i <= 131; i++) {
        input[i] = 0;
    }
    Intent intent1 = new Intent(this, Loading.class);
    String dis = predict_Diseases[element];
    intent1.putExtra("Disease_name", dis);
    intent1.putExtra("pname", lname);
    intent1.putExtra("age", age);
    intent1.putExtra("disease_category", disease_category[element]);
    intent1.putExtra("Symptoms", String.valueOf(symList));

```



```
        startActivity(intent1);
    }
}
```

Doctor_Waiting_Approval.java

```
package com.example.vcare.doctor;
import android.content.Context;
import android.content.DialogInterface;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.example.vcare.R;
import com.example.vcare.appointments.Appointment_notif;
import com.example.vcare.appointments.Booking_Appointments;
import com.example.vcare.patient.Patient_Details;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.ArrayList;
import java.util.Date;
import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
```

```
import javax.mail.internet.MimeMessage;
public class Doctor_appointments_waiting_approval extends Fragment {
    private RecyclerView recyclerView;
    private FirebaseUser user;
    private DatabaseReference reference, reference_user, reference_doctor, reference_booking,
reference_patient, reference_details, reference_doctor_appt;
    private ArrayList<Appointment_details> current_appoint;
    private ArrayList<DataSnapshot> data_appoint;
    private String email;
    private Date d1, d2;
    private Doctor_Appointment_Show_Adapter adapter;
    private EditText search;
    private Context mContext;
    FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();
    public Doctor_appointments_waiting_approval() {
    }

    public static Doctor_appointments_waiting_approval getInstance() {
        Doctor_appointments_waiting_approval currentFragment = new
Doctor_appointments_waiting_approval();
        return currentFragment;
    }
    @Override
    public void onAttach(@NonNull Context context) {
        super.onAttach(context);
        mContext = context;
    }
    private void senddocmail(String rvmai,String msg) {
        try {
            final String frommail = "ashishvwprj@gmail.com";
            final String fpass = "ggezrowmplvieddv";
            String subject = "Appointment Status";
            String message=msg;
            String stringHost = "smtp.gmail.com";
            Properties properties = System.getProperties();
            properties.put("mail.smtp.host", stringHost);
            properties.put("mail.smtp.port", "465");
            properties.put("mail.smtp.ssl.enable", "true");
            properties.put("mail.smtp.auth", "true");
            Session session = Session.getInstance(properties,
                new javax.mail.Authenticator() {
                    @Override
                    protected javax.mail.PasswordAuthentication getPasswordAuthentication() {
                        return new PasswordAuthentication(frommail,fpass);
                    }
                }
            );
            MimeMessage messageObj = new MimeMessage(session);
            messageObj.setFrom(new InternetAddress(frommail));
            messageObj.setSubject(subject);
            messageObj.setText(message);
            InternetAddress[] toAddresses = {new InternetAddress(rvmai)};
            messageObj.setRecipients(Message.RecipientType.TO, toAddresses);
            Transport.send(messageObj);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

        }
    });
    MimeMessage mimeTypeMessage = new MimeMessage(session);
    mimeTypeMessage.setFrom(new InternetAddress(frommail));
    mimeTypeMessage.addRecipient(Message.RecipientType.TO, new InternetAddress(rvmail));
    mimeTypeMessage.setSubject(subject);
    mimeTypeMessage.setText(message);
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                Transport.send(mimeTypeMessage);
            } catch (MessagingException e) {
                e.printStackTrace();
            }
        }
    });
    thread.start();

} catch (AddressException e) {
    e.printStackTrace();
}
catch (MessagingException e) {
    e.printStackTrace();
}
}
@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View view = inflater.inflate(R.layout.row_current, container, false);
    user=firebaseAuth.getCurrentUser();
    search = (EditText) view.findViewById(R.id.editTextSearch_current);
    search.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        }
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
        }
        @Override
        public void afterTextChanged(Editable s) {
            filter(s.toString());

```

```

    }
});
recyclerView = (RecyclerView) view.findViewById(R.id.recycler_view);
recyclerView.setHasFixedSize(true);
recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
current_appoint = new ArrayList<>();
data_appoint = new ArrayList<>();
reference = FirebaseDatabase.getInstance("https://vcare-healthapp-default-rtdb.asia-southeast1.firebaseio.com").getReference("Appointment");
reference_booking = FirebaseDatabase.getInstance("https://vcare-healthapp-default-rtdb.asia-southeast1.firebaseio.com").getReference("Doctors_Chosen_Slots");
reference_patient = FirebaseDatabase.getInstance("https://vcare-healthapp-default-rtdb.asia-southeast1.firebaseio.com").getReference("Patient_Chosen_Slots");
reference_details = FirebaseDatabase.getInstance("https://vcare-healthapp-default-rtdb.asia-southeast1.firebaseio.com").getReference("Patient_Details");
reference_doctor_appt = FirebaseDatabase.getInstance("https://vcare-healthapp-default-rtdb.asia-southeast1.firebaseio.com").getReference("Doctors_Appointments");
reference.child("waiting_approval").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (snapshot.exists()) {
            current_appoint = new ArrayList<>();
            data_appoint = new ArrayList<>();
            for (DataSnapshot snapshot1 : snapshot.getChildren()) {
                for (DataSnapshot snapshot2 : snapshot1.getChildren()) {
                    for (DataSnapshot snapshot3 : snapshot2.getChildren()) {
                        Appointment_details appoint_data =
snapshot3.getValue(Appointment_details.class);
                        if(appoint_data.getEmail().equals(user.getEmail().replace(".", ",")))
                        {
                            current_appoint.add(appoint_data);
                            data_appoint.add(snapshot3);
                        }
                    }
                }
            }
        }
    }
});
adapter = new Doctor_Appointment_Show_Adapter(current_appoint);
recyclerView.setAdapter(adapter);

ItemTouchHelper.SimpleCallback touchHelperCallback = new
ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
    @Override

```

```

        public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull
RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
            return false;
        }
        @Override
        public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int
direction) {
            new AlertDialog.Builder(viewHolder.itemView.getContext())
                .setMessage("Do you want to mark this Appointment as Done? or
Cancel this Appointment")
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        int position = viewHolder.getAdapterPosition();
                        DataSnapshot data = data_appoint.get(position);
                        Appointment_details appoint_class = current_appoint.get(position);
                        //appointment approved by doctor

reference.child("appointment_approved").child(appoint_class.getPemail()).child(appoint_class
.getDate()).child(appoint_class.getTime()).setValue(appoint_class);

reference.child("appointment_approved").child(appoint_class.getPemail()).child(appoint_class
.getDate()).child(appoint_class.getTime()).child("flag").setValue(1);

reference.child("appointment_approved").child(appoint_class.getPemail()).child(appoint_class
.getDate()).child(appoint_class.getTime()).child("payment_status").setValue(1);
                        data.getRef().removeValue();
                        Patient_Details details = new
Patient_Details(appoint_class.getPemail().replace(".", ","), appoint_class.getName());
                        //patient email added
                        String encoded_email = appoint_class.getEmail().replace(".", ",");

reference_details.child(encoded_email).child(appoint_class.getPemail().replace(".", ",")).setVal
ue(details);
reference_patient.child(encoded_email).child(appoint_class.getDate()).child(appoint_class.get
Time()).child("flag").setValue(1);
reference_patient.child(encoded_email).child(appoint_class.getDate()).child(appoint_class.get
Time()).child("question").addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        String ques = " ";
                        if (snapshot.exists()) {
                            ques = snapshot.getValue(String.class);
                        }
                    }
                })

```

```

Appointment_notif appointment_notif = new
Appointment_notif("1", appoint_class.getDate(), appoint_class.getTime(), ques,
appoint_class.getPhone(), appoint_class.getName(),appoint_class.getTransaction());
reference_doctor_appt.child(encoded_email).child(appoint_class.getDate()).child(appoint_class.getTime()).setValue(appointment_notif);

String check = appoint_class.getTime().split("-", 5)[0];
Booking_Appointments booking = new
Booking_Appointments(1, appoint_class.getPhone());

reference_booking.child(encoded_email).child(appoint_class.getDate()).addListenerForSingle
ValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot
snapshot) {
        if (snapshot.exists()) {
            String slot_val = "";
            for (DataSnapshot dataSnapshot :
snapshot.getChildren()) {
                String item = dataSnapshot.getKey();
                int s = Integer.parseInt(item.split("-", 5)[0]);
                int e = Integer.parseInt(item.split("-", 5)[1]);
                int t =
Integer.parseInt(appoint_class.getTime().split("-", 5)[0]);
                if (s <= t && t < e) {
                    slot_val = item;
                    break;
                }
            }
            reference_booking.child(encoded_email).child(appoint_class.getDate()).child(slot_val).child(c
heck).child("email").setValue(appoint_class.getPemail());
            senddocmail(appoint_class.getPemail().replace(",","."),"\nHello
"+appoint_class.getName()+","+
                "\n Thanks for booking an appointment on V-
Care."+
                "\nyour appointment is Approved by doctor for
time slot "+appoint_class.getTime()+" on "+appoint_class.getDate());
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});

```

```

    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
    });
}
})
.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        int position = viewHolder.getAdapterPosition();
        DataSnapshot data = data_appoint.get(position);
        Appointment_details appoint_class = current_appoint.get(position);

reference.child("appointment_approved").child(appoint_class.getPemail()).child(appoint_class
.getDate()).child(appoint_class.getTime()).setValue(appoint_class);

reference.child("appointment_approved").child(appoint_class.getPemail()).child(appoint_class
.getDate()).child(appoint_class.getTime()).child("flag").setValue(2);

reference.child("appointment_approved").child(appoint_class.getPemail()).child(appoint_class
.getDate()).child(appoint_class.getTime()).child("payment_status").setValue(2);
        data.getRef().removeValue();
        String encoded_email = appoint_class.getEmail().replace(".", ",");
reference_patient.child(appoint_class.getPhone()).child(encoded_email).child(appoint_class.ge
tDate()).child(appoint_class.getTime()).child("appointment").setValue(2);
        String check = appoint_class.getTime().split("-", 5)[0];
        Booking_Appointments booking = new Booking_Appointments(0,
"null");
reference_booking.child(encoded_email).child(appoint_class.getDate()).addListenerForSingle
ValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (snapshot.exists()) {
            String slot_val = "";
            for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                String item = dataSnapshot.getKey();
                int s = Integer.parseInt(item.split("-", 5)[0]);
                int e = Integer.parseInt(item.split("-", 5)[1]);
                int t = Integer.parseInt(appoint_class.getTime().split("-",
5)[0]);

                if (s <= t && t < e) {

```

```

        slot_val = item;
        break;
    }
}
if (!(slot_val.equals("")))) {
reference_booking.child(encoded_email).child(appoint_class.getDate()).child(slot_val).child(c
heck).setValue(booking);

    String finalSlot_val = slot_val;
reference_booking.child(encoded_email).child(appoint_class.getDate()).child(slot_val).child("
Count").addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot
snapshot) {

        int count = snapshot.getValue(Integer.class);
        count = count - 1;
reference_booking.child(encoded_email).child(appoint_class.getDate()).child(finalSlot_val).ch
ild("Count").setValue(count);
senddocmail(appoint_class.getPemail().replace(",","."),"\nHello
"+appoint_class.getName()+","+
        "\n Thanks for booking an appointment on V-
Care."+
        "\nyour appointment is Declined by  doctor for
time slot "+appoint_class.getTime()+" on "+appoint_class.getDate());

    }
    @Override
    public void onCancelled(@NonNull DatabaseError
error) {

    }
});
}
}
}
@Override
public void onCancelled(@NonNull DatabaseError error) {

}
});
}
.create()
.show();
}
};

```



```

        ItemTouchHelper itemTouchHelper = new
        ItemTouchHelper(touchHelperCallback);
        itemTouchHelper.attachToRecyclerView(recyclerView);
    } else {
        if (mcontext != null) {
            Toast.makeText(mcontext, "There are no Appointments!",
            Toast.LENGTH_SHORT).show();
        }
    }
}
@Override
public void onCancelled(@NonNull DatabaseError error) {
}
});
return view;
}
private void filter(String text) {
    ArrayList<Appointment_details> filterdNames = new ArrayList<>();
    for (Appointment_details data : current_appoint) {
        //if the existing elements contains the search input
        if (data.getDate().toLowerCase().contains(text.toLowerCase())) {
            //adding the element to filtered list
            filterdNames.add(data);
        }
    }
    adapter.filterList(filterdNames);
}
}

```

Predictiondetails.java

```

package com.example.vcare.doctor;
import android.os.Bundle;
import android.text.format.DateFormat;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import com.example.vcare.R;
import com.example.vcare.predictor.DiseasePrediction;
import com.google.firebase.database.DataSnapshot;

```

```

import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.ArrayList;
import java.util.Calendar;
public class predictiondetails extends AppCompatActivity {
    private TextView patientname, symptomsdis, diseasename, specialitytype, patientage,
    docremark,patientgender;
    private ImageView imageicon;
    private Doctor_Images doctor_images;
    private ArrayList<Appointment_details> previous_payment;
    private Button declinebtn,approvebtn,notvrf,vrf;
    private DatabaseReference reference_dispred,
    reference_doctor,reference_status,reference_book,reference_doctor_name;
    private int start, end;
    private String encoded_email,demail,dname;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.prediction_details);
        patientname = (TextView) findViewById(R.id.patientname);
        symptomsdis = (TextView) findViewById(R.id.symptoms);
        diseasename = (TextView) findViewById(R.id.diseasename);
        specialitytype = (TextView) findViewById(R.id.specialitytype);
        patientage = (TextView) findViewById(R.id.patientage);
        patientgender =(TextView) findViewById(R.id.patientgender);
        docremark =(TextView) findViewById(R.id.docremark);
        imageicon = (ImageView)findViewById(R.id.imageicon);
        doctor_about = (TextView) findViewById(R.id.about_doctor);
        emailid = (TextView) findViewById(R.id.email_text_val);
        doctor_image = (ImageView) findViewById(R.id.imageView_doc_display);*/
        declinebtn=(Button)findViewById(R.id.decline);
        approvebtn=(Button)findViewById(R.id.approve);
        notvrf=findViewById(R.id.notVrf);
        vrf=findViewById(R.id.vrf);
        Session_Mangement _session_mangement = new Session_Mangement(this);
        demail = _session_mangement.getDoctorSession()[0].replace(".", "");
        Calendar startDate = Calendar.getInstance();
        startDate.add(Calendar.MONTH, 0);
        String date_val = DateFormat.format("dd MMM yyyy", startDate).toString();
        String email = getIntent().getSerializableExtra("Email ID").toString();
        String approvalId= getIntent().getSerializableExtra("ApprovalId").toString();
        String name = getIntent().getSerializableExtra("Patient Name").toString();

```

```

String symptoms = getIntent().getSerializableExtra("Symptoms").toString();
String speciality = getIntent().getSerializableExtra("speciality").toString();
String predictedDisease = getIntent().getSerializableExtra("PredictedDisease").toString();
String gender = getIntent().getSerializableExtra("gender").toString();
String age = getIntent().getSerializableExtra("age").toString();
patientname.setText(name);
symptomsdis.setText(symptoms);
diseasename.setText(predictedDisease);
specialitytype.setText(speciality);
patientgender.setText(gender);
patientage.setText(age);
encoded_email = email.replace(".", "");
reference_dispred = FirebaseDatabase.getInstance("https://vcare-healthapp-default-
rtbd.firebaseio.com").getReference("disease_prediction");
reference_doctor = FirebaseDatabase.getInstance("https://vcare-healthapp-default-
rtbd.firebaseio.com").getReference("User_data");
reference_doctor_name = FirebaseDatabase.getInstance("https://vcare-healthapp-default-
rtbd.firebaseio.com").getReference("Doctors_Data");
reference_doctor.child(email.replace(".", "")).addValueEventListener(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if(snapshot.exists()){
            String gender = snapshot.child("gender").getValue(String.class);
            patientgender.setText(gender);
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }
});
reference_doctor_name.child(email.replace(".", "")).addValueEventListener(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if(snapshot.exists()){
            dname = snapshot.child("name").getValue(String.class);
            //patientgender.setText(dname);
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }
});

```

```

        getapprovalStatus(reference_dispred,approvalId,email.replace(".",","));
approvebtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String remark = docremark.getText().toString();
        if (remark.isEmpty()) {
            docremark.setError("Please add your remarks");
            docremark.requestFocus();
            return;
        }

        reference_dispred.child(email.replace(".",",")).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if(snapshot.exists()){
                    for(DataSnapshot dnsnapshot : snapshot.getChildren()){
                        DiseasePrediction dp = dnsnapshot.getValue(DiseasePrediction.class);
                        if(dp.getPid().equals(approvalId)){
reference_dispred.child("approved").child(dp.getPatientEmail().replace(".",",")).child(approval
Id).setValue(dp);
reference_dispred.child(dp.getPatientEmail().replace(".",",")).child(approvalId).child("approva
lstatus").setValue(1);
reference_dispred.child(dp.getPatientEmail().replace(".",",")).child(approvalId).child("docemai
l").setValue(demail);
reference_dispred.child(dp.getPatientEmail().replace(".",",")).child(approvalId).child("docnam
e").setValue(dname);
reference_dispred.child(dp.getPatientEmail().replace(".",",")).child(approvalId).child("remarks
").setValue(docremark.getText().toString().trim());
                            imageicon.setVisibility(view.VISIBLE);
                            approvebtn.setVisibility(view.GONE);
                            declinebtn.setVisibility(view.GONE);
                            getapprovalStatus(reference_dispred, approvalId, email.replace(".",
","));
reference_dispred.child("pending").child(dp.getPatientEmail().replace(".",",")).child(approvalI
d).getRef().removeValue();
                        }
                    }
                }
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {

        }
    }
}

```

```

        });
    }
});

declinebtn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        String remark = docremark.getText().toString();
        if (remark.isEmpty()) {
            docremark.setError("Please add your remarks");
            docremark.requestFocus();
            return;
        }
        reference_dispred.child(email.replace(".", ",")).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if(snapshot.exists()){
                    for(DataSnapshot dnsnaspshot : snapshot.getChildren()){
                        DiseasePrediction dp = dnsnaspshot.getValue(DiseasePrediction.class);
                        if(dp.getPid().equals(approvalId)){
reference_dispred.child("approved").child(dp.getPatientEmail().replace(".", ",")).child(approval
Id).setValue(dp);
reference_dispred.child(dp.getPatientEmail().replace(".", ",")).child(approvalId).child("approva
lstatus").setValue(2);
reference_dispred.child(dp.getPatientEmail().replace(".", ",")).child(approvalId).child("remarks
").setValue(docremark.getText().toString().trim());
reference_dispred.child(dp.getPatientEmail().replace(".", ",")).child(approvalId).child("docemai
l").setValue(demail);
reference_dispred.child(dp.getPatientEmail().replace(".", ",")).child(approvalId).child("docnam
e").setValue(dname);

                            approvebtn.setVisibility(view.GONE);
                            declinebtn.setVisibility(view.GONE);
                            getapprovalStatus(reference_dispred, approvalId, email.replace(".",
", "));
reference_dispred.child(dp.getPatientEmail().replace(".", ",")).child(approvalId).getRef().remov
eValue();

                            Toast.makeText(predictiondetails.this, "There predition approval is
cancelled!", Toast.LENGTH_SHORT).show();
                        }
                    }
                }
            }
        });
    }
}

```

```
        @Override
        public void onCancelled(@NonNull DatabaseError error) {

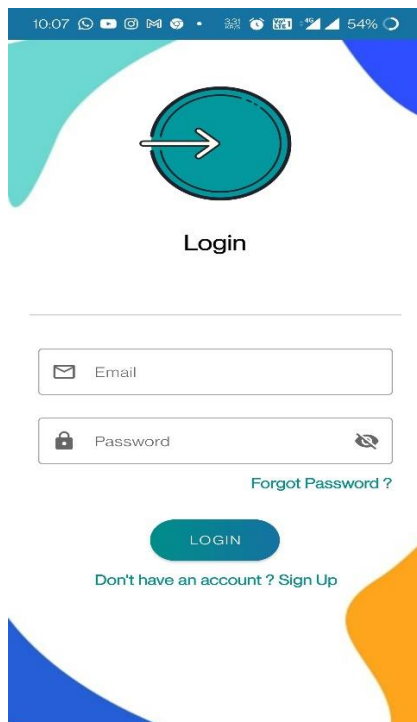
        }
    });
}
```

9.1 Screen Shots

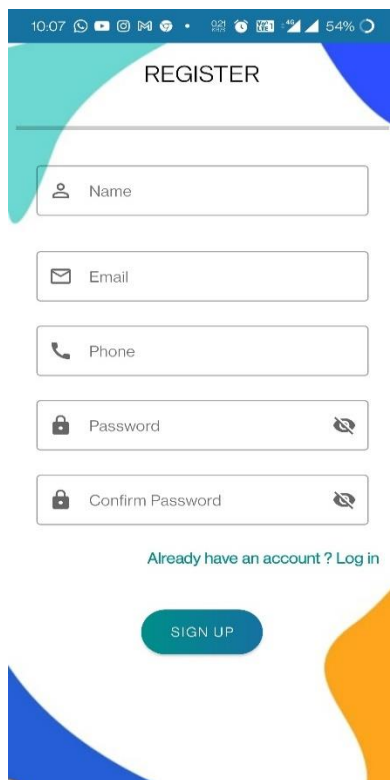
SplashScreen



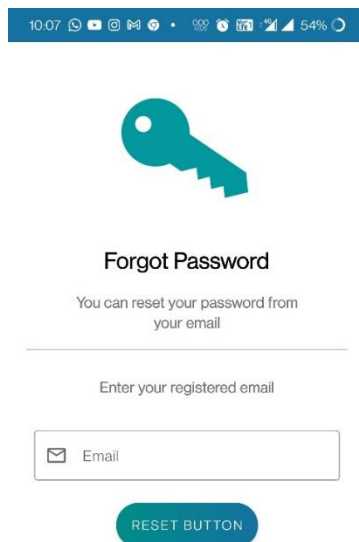
LoginPage

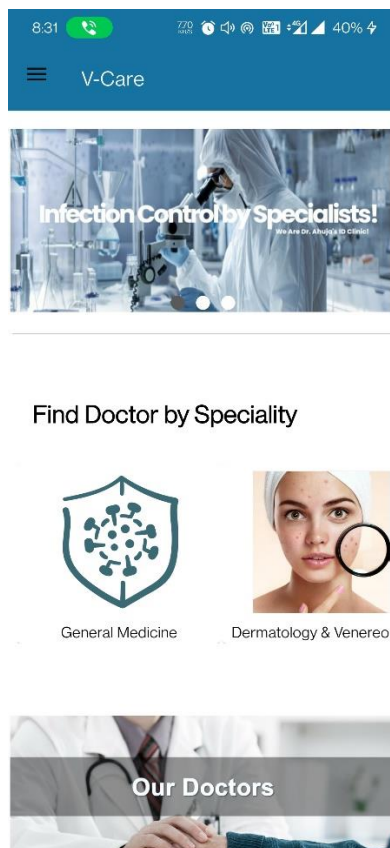
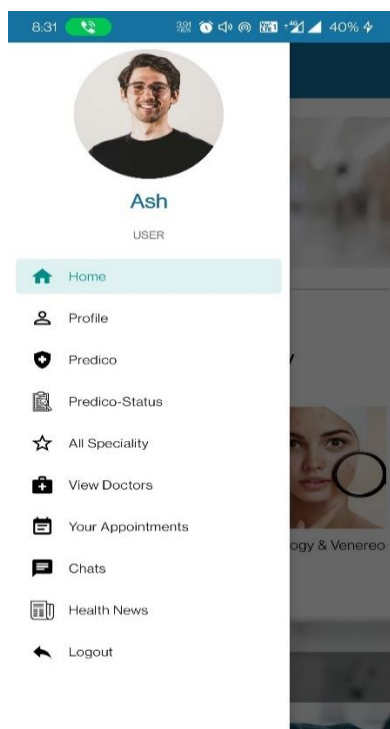


Register page



Forgot Password Page




Patienthome page**Patient NavigationDrawer page**


Patient profile page

8:32

3:44

41%



Edit Profile 

Name
Ash

Gender
Male

Email
ashishvwprj@gmail.com

Phone
9447883940

Bio
I'm a user


Doctor List page

8:34


44%

Available Doctors


Search Doctors here




ashish
ashishwilson@gmail.com
General Medicine



ashish
ashishwhere@gmail.com
Neurology



Sooraj
wilsonavarickamakel@gmail.com
Gastroenterology, Hepatology & Endoscopy



sumith
wilsonvantony@gmail.com
Dermatology & Venereology

Predico patientside Dashboard

The screenshot shows the Predico patientside Dashboard app interface. At the top is a status bar with the time 8:32, a green call icon, and various system icons including signal strength, Wi-Fi, and battery level at 41%. Below the status bar is a header titled "Symptoms". The form includes fields for "Enter Name:" with the value "Ash", "Enter Age:" which is empty, and "Select Gender:" with radio buttons for "Male" and "Female". There are four "Select a symptom" dropdown menus. Below these is a "SELECT DURATION" section with radio buttons for "Days", "Weeks", and "Months". At the bottom is a large green button with a white checkmark and the text "CHECK".

Predico result page

The screenshot shows the Predico result page app interface. At the top is a status bar with the time 8:32, a green call icon, and various system icons including signal strength, Wi-Fi, and battery level at 41%. Below the status bar is a header titled "Allergy" with the text "Approval Pending" below it. There is a green button with the text "NEED A DOCTOR". Below this is a large text block containing information about allergies. The text is as follows:

Allergies, also known as allergic diseases, are a number of conditions caused by hypersensitivity of the immune system to typically harmless substances in the environment. These diseases include hay fever, food allergies, atopic dermatitis, allergic asthma, and anaphylaxis. Symptoms may include red eyes, an itchy rash, sneezing, a runny nose, shortness of breath, or swelling. Food intolerances and food poisoning are separate conditions.


Many allergens such as dust or pollen are airborne particles. In these cases, symptoms arise in areas in contact with air, such as eyes, nose, and lungs. For instance, allergic rhinitis, also known as hay fever, causes irritation of the nose, sneezing, itching, and redness of the eyes. Inhaled allergens can also lead to increased production of mucus in the lungs, shortness of breath, coughing, and wheezing.

Risk factors for allergy can be placed in two general categories, namely host and environmental factors. Host factors include heredity, sex, race, and age, with heredity being by far the most significant. However, there have been recent increases in the incidence of allergic disorders that cannot be explained by genetic factors alone. Four major environmental candidates are alterations in exposure to General Medicines during early childhood, environmental

Appointment Booking patientside page

9:58

56%



ashish

MBBS

Select Date

Jul

11

Mon

Jul

12

Tue

Jul

13

Wed

Jul

14

Thu

Jul

15


Fri

Select Time

11:00 - 11:30


Patient's Name

Enter your name

 Ash

Required3/20









Appointment For

 Any Questions for Doctor?


Required0/200


Appointment payment page

9:58

56%







V-Care

Reference No. #1001


₹ 265


English 


 +919988776655


| ashishvwilson@gmail.com


PREFERRED PAYMENT METHODS


 Netbanking - Axis Bank




 UPI - Google Pay




 Netbanking - IDBI




CARDS, UPI & MORE

 Card


Visa, MasterCard, RuPay, and Maestro

 UPI


Pay with installed app, or use others




Google Pay



PhonePe



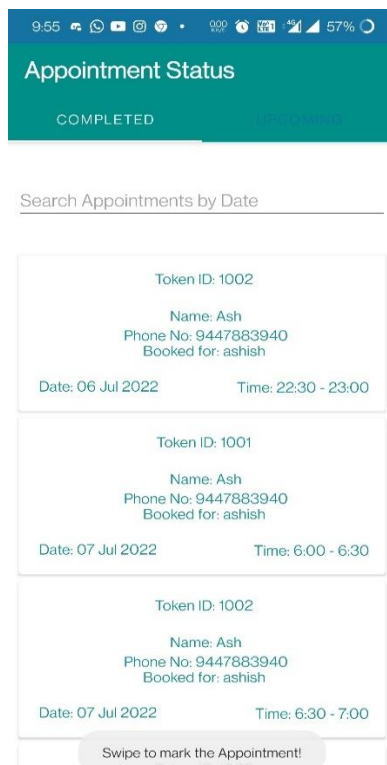
PayTM



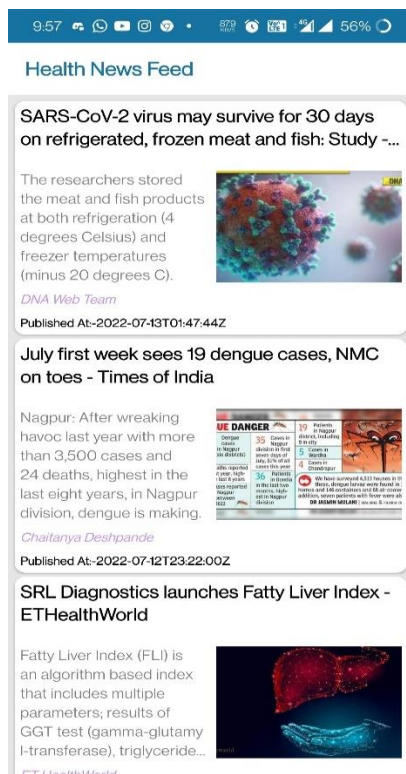
Others

PAY ₹ 265

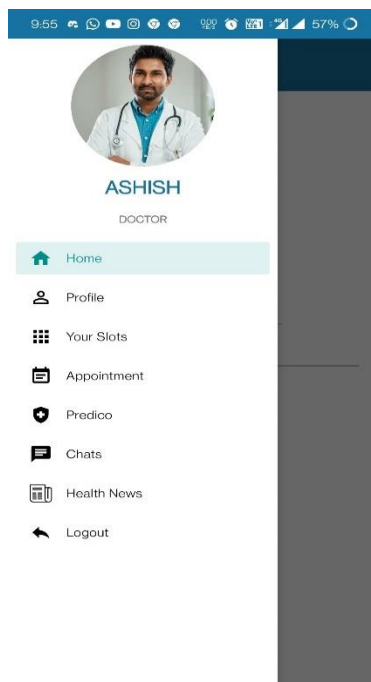
Appointment status Doctor side



Health News



Doctor NavigationDashboard



Add New Slots

A screenshot of a mobile application interface for adding new slots. The title 'Choose your Availability' is at the top. Below it is a calendar icon with red 'X' marks. There are three dropdown menus for selecting a date, time, and another time. The first dropdown shows '13 Jul 2022', the second shows '11:00', and the third shows '13:00'. A blue 'SELECT' button is below the dropdowns. Underneath, the text 'Your Chosen Slots' is followed by two rows of information: 'Date: 06 Jul 2022 Time: 22 - 23 Slots Booked: 2' and 'Date: 07 Jul 2022 Time: 6 - 9 Slots Booked: 2'.

Prediction details doctorside

9:57 9:57 WhatsApp YouTube Instagram Facebook Messenger Email 100% 100% 5G 56% 56%

Prediction details

Ashish 23 Male

skin_rash,
nodal_skin_eruptions,
itching

APPROVE

DECLINE

predictedisease

Fungal infection

Diseasecategory

Dermatology & Venereology

Doctor Remarks

Remarks

Prediction details

9:57 9:57 WhatsApp YouTube Instagram Facebook Messenger Email 110 110 5G 56% 56%

Prediction details

Ashish 23 Male

skin_rash,
nodal_skin_eruptions,
itching

predictedisease

Fungal infection

Diseasecategory

Dermatology & Venereology

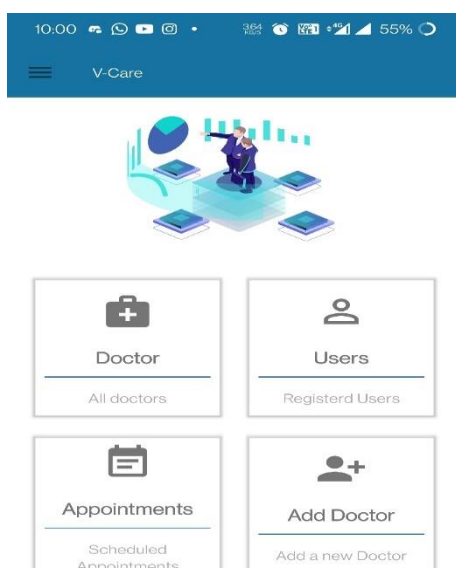
Doctor Remarks

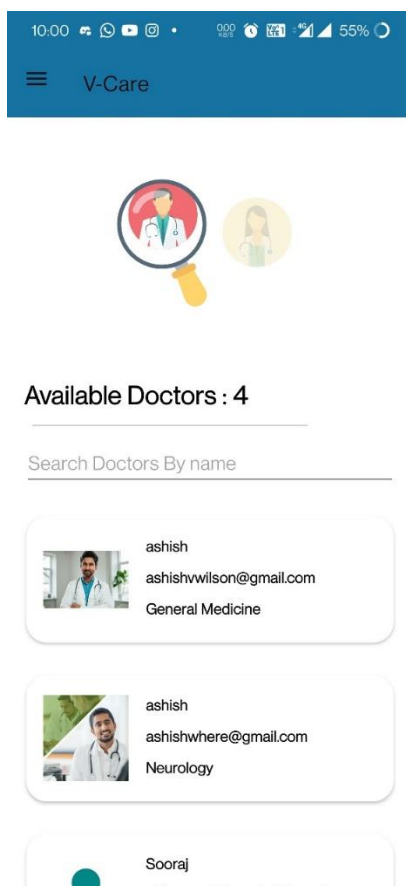
consult a doctor

Chats




Admin Panel



Doctor list view admin side

9.3 Plagiarism Report

		Similarity Report ID: oid:10159:19940880	
PAPER NAME			
ProjectReport.pdf			
WORD COUNT		CHARACTER COUNT	
9499 Words		53882 Characters	
PAGE COUNT		FILE SIZE	
55 Pages		1.3MB	
SUBMISSION DATE		REPORT DATE	
Jul 19, 2022 3:02 PM GMT+5:30		Jul 19, 2022 3:03 PM GMT+5:30	


● 13% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 13% Internet database
- 1% Publications database
- Crossref database

● Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less than 8 words)
- Manually excluded sources

 **turnitin**

Similarity Report ID: oid:10159:19940880

● **13% Overall Similarity**

Top sources found in the following databases:

- 13% Internet database
- 1% Publications database
- Crossref database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	seminarprojects.com	4%
	Internet	
2	pdfcoffee.com	2%
	Internet	
3	dl.icdst.org	<1%
	Internet	
4	couchbase.com	<1%
	Internet	
5	docs.microsoft.com	<1%
	Internet	
6	docshare.tips	<1%
	Internet	
7	coursehero.com	<1%
	Internet	
8	firebase.google.com	<1%
	Internet	
9	geeksforgeeks.org	<1%
	Internet	