

**A PROJECT BASED LEARNING-II REPORT ON**

**QUICK MEDICAL DIAGNOSIS WITH AUTOMATED SYMPTOMS ANALYZER AND  
BASIC HEALTHCARE MANAGEMENT APP**

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE  
IN THE FULFILLMENT OF THE PBL-II TW

OF

**SECOND YEAR OF COMPUTER ENGINEERING**

**SUBMITTED BY**

<b>STUDENT NAME</b>	<b>Roll No:</b>
Advait Naik	21354
Shubham Panchal	21356
Chinmay Patil	21358
Soaham Pimparkar	21362



**DEPARTMENT OF COMPUTER ENGINEERING**

**PUNE INSTITUTE OF COMPUTER TECHNOLOGY**

DHANKAWADI, PUNE – 411043



**SCTR's PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE**

**[DEPARTMENT OF COMPUTER ENGINEERING**

**2022 -2023**

**CERTIFICATE**

This is to certify that the SPPU Curriculum-based Project Based Learning-II report entitled :

**QUICK MEDICAL DIAGNOSIS WITH AUTOMATED SYMPTOMS ANALYZER AND  
BASIC HEALTHCARE MANAGEMENT APP**

Submitted by

<b>STUDENT NAME</b>	<b>Roll No:</b>
Advait Naik	21354
Shubham Panchal	21356
Chinmay Patil	21358
Soaham Pimparkar	21362

has satisfactorily completed the curriculum-based Project Based Learning-II under the guidance of  
Prof. P. P. Joshi towards the fulfillment of second year Computer Engineering Semester IV,  
Academic Year 2022-23 of Savitribai Phule Pune University.

**Prof. P.P. Joshi**  
PBL-II Guide  
Department of Computer Engineering

**Dr. Geetanjali V. Kale**  
Head,  
Department of Computer Engineering

Place:

Date :

## ACKNOWLEDGEMENT

It gives us great pleasure in presenting the PBL-II report on “**Quick Medical Diagnosis with Automated Symptoms Analyzer and Basic Healthcare Management App**”

Firstly, we would like to express our gratitude to our PBL-II guide **Prof. P.P. Joshi** for her guidance and supervision. She gave us valuable suggestions and ideas when we needed them and encouraged us to work on this project which proved to be beneficial in the overall completion of this PBL- II work. Thank you for your continuous support and presence whenever needed.

We are thankful to our Head of the Computer Engineering Department, **Dr. Geetanjali V.Kale**, for her indispensable support and suggestions throughout the PBL-II project.

We would also genuinely like to express my gratitude to the Department PBL-II Coordinator, **Prof. Rutuja A. Kulkarni** for the timely resolution of the doubts related to the PBL-II project.

We are immensely grateful to all the individuals who were directly or indirectly involved in this project as without their inspiration and valuable suggestions it would not have been possible to develop the project within the prescribed time.

### NAME OF THE STUDENTS

Advait Naik  
Shubham Panchal  
Chinmay Patil  
Soaham Pimparkar

## **ABSTRACT**

In India, medical health services are becoming expensive in urban and rural areas. The availability of medical facilities is sparse in rural areas due to which patients are devoid of proper medication. By building this project, we are bringing doctors and patients closer where patients could find the best doctor available for treating their ailment. As a surplus, prescriptions, medical history and dose reminders are also provided for the patient's wellbeing. Epidemic tracking could also be performed by tracking the number of cases of a particular disease occurring in a locality.

Our project includes two mobile applications, one for the user/patient and another for doctors who have subscribed to our service. The user and doctor need to authenticate themselves with the apps, providing essential information that would identify them uniquely. For doctors, appointment details, check-up fees, speciality etc. are also included. The users could provide their health-oriented details such as age, weight, blood group and history of other ailments if needed. The users will then get a gross automated diagnosis based on the symptoms they have entered.

Following the diagnosis, the users will get contact details of doctors and hospitals in their locality that specialize the diagnosed ailment. Moreover, the automated diagnosis also gets verified by a doctor. Prescriptions are sent digitally to the user's app, which creates automatic dose remainders on their device.

## TABLE OF CONTENTS

LIST OF FIGURES

7

Sr. No.	Title of Chapter	Page No.
01	<b>Introduction</b>	7
1.1	<b>Motivation</b>	7
1.2	<b>Problem Statement and Objectives</b>	7
02	<b>Software Requirements Specification</b>	9-12
2.1	<b>Project Scope</b>	8
2.2	<b>SDLC Model</b>	8
2.3	<b>Functional Requirements</b>	9
2.3.1	System Feature 1 : User Registration and Authentication	9
2.3.2	System Feature 2 : Automated Diagnosis	9
2.3.3	System Feature 3 : User Details and Profile	9
2.3.4	System Feature 4 : Doctor Registration	9
2.4	<b>Nonfunctional Requirements</b>	9-10
2.4.1	Performance Requirements	9-10
2.4.2	Safety/ Security Requirements	10
2.5	<b>System Requirements</b>	10-11
2.5.1	Database Requirements	10
2.5.2	Software Requirements (Platform Choice)	10
2.5.3	Hardware Requirements	11
2.6	<b>System Implementation Plan</b>	11
03	<b>System Design</b>	12-22
3.1	Flowchart / Activity Diagrams	12
3.2	Implementation and Results	13-22
04	<b>Other Specification</b>	22-23
4.1	Advantages	22
4.2	Limitations	22-23
4.3	Applications	23
05	<b>Conclusions &amp; Future Work</b>	24-25
06	<b>References</b>	26

## LIST OF FIGURES

FIGURE	ILLUSTRATION	PAGE NO.
1.1	System Overview	3
1.2	System Behavior	5

# **1. INTRODUCTION**

## **1.1 MOTIVATION**

In India, medical health services are becoming expensive in urban and rural areas. The availability of medical facilities is sparse in rural areas due to which patients are devoid of proper medication. By building this project, we are bringing doctors and patients closer where patients could find the best doctor available for treating their ailment. As a surplus, prescriptions, medical history and dose reminders are also provided for the patient's wellbeing. Epidemic tracking could also be performed by tracking the number of cases of a particular disease occurring in a locality.

## **1.2 OBJECTIVES**

While emphasizing on quick medical diagnosis, the following objectives are at the core of the project:

- Providing no-cost medical diagnosis for non-severe symptoms or ailments
- Manage prescriptions and dose-reminders for users
- Providing information on the healthcare infrastructure in a user's locality
- Validating symptoms and the corresponding automated diagnosis to ensure reliability of the diagnosis process

## **1.3. Problem Statement and Objectives**

### **Objectives**

- To provide a convenient and accessible way for users to check their symptoms and get an automated diagnosis based on the latest medical knowledge and algorithms.
- To help users find nearby doctors who can offer further consultation, treatment, or referral based on their diagnosis and preferences.
- To improve the quality and efficiency of healthcare delivery by reducing unnecessary visits to clinics or hospitals and facilitating timely intervention for serious conditions.

## **2. SOFTWARE REQUIREMENTS SPECIFICATION**

The software requirements specification (SRS) outlines the functional and non-functional requirements of our project. These requirements include system features, performance requirements, safety/security requirements, and database, software, and hardware requirements. The SRS serves as a guide for the development team, ensuring that the app meets the necessary specifications and objectives.

### **2.1 PROJECT SCOPE**

Our app can be used by patients who have symptoms concerned with common ailments. Doctors can register themselves and patients can search them based on ailments and locality. All kinds of users would be able to use the 'patient app' whereas only registered doctors can authenticate and add their details as well as details of their clinic.

### **2.2 SDLC MODEL**

Every software program possesses a development lifecycle that has eight steps to go through: Conception, Requirement analysis, Design, Coding - Debugging, Testing, Release, Maintenance, and Retirement. The chosen Software Development Life Cycle (SDLC) model for the project is the Agile methodology.

Agile development models are more suitable for small- and medium-sized projects as they promote iteration and accumulation. These models emphasize focusing resources on the product itself rather than on operational processes such as documentation. Hence, they encourage small but frequent increments to a product rather than larger but less frequent ones since it is easier to maintain trimmer, incremental contributions. Design, Coding – Debugging, and Testing steps are usually iterated many times in these models. Those steps do not happen one after another but take place in parallel and complement each other

Given that the team is small and the application is still at the beginning of its lifecycle, it appears that Agile development models would be an excellent match for this project. The requirements are anticipated to be constantly updated as the development team progresses. Working in iterations also enables the team to learn, review current performance, and improve on shortcomings, which is highly



beneficial as it consists of university students only. Furthermore, dividing features and tasks into manageable pieces would make it easier for the backlog to be logically organized and problems to be detected and resolved.

## **2.3 FUNCTIONAL REQUIREMENTS**

### 2.3.1 System Feature 1: User Registration and Authentication

Users should be able to register and create an account.

The app should provide secure authentication mechanisms.

### 2.3.2 System Feature 2: Automated Diagnosis

Given the symptoms, the user should get an approximate diagnosis or predictions of which disease he/she might have.

### 2.3.3 System Feature 3: User Details and Profile

The user should be able to view their past diagnosis, profile details and ongoing prescriptions in the app.

### 2.3.4. System Feature 4: Doctor Registration

Doctors are able to register themselves with their name, speciality and location.

## **2.4 NON FUNCTIONAL REQUIREMENTS**

### 2.4.1 Performance Requirements:

- The app should be able to process the user input quickly and accurately, and return a list of possible diagnoses based on the symptoms and other relevant factors, such as age, gender, medical history, etc.
- The app should be able to suggest nearby doctors who specialize in the relevant field or condition, and provide contact details, ratings, reviews, availability, etc. for each doctor.

- The app should be able to integrate with the user's location services, calendar, contacts, and other apps or devices that can facilitate booking an appointment or getting directions to the doctor's office.
- The app should be able to handle multiple requests simultaneously, and maintain a high level of security and privacy for the user's data and communication.

#### 2.4.2 Safety/Security Requirements:

- The app must not store or share any personal or medical information of the user without their explicit consent.
- The app must use encryption and authentication methods to protect the data transmission and storage from unauthorized access or modification.
- The app must clearly state the limitations and uncertainties of its diagnosis and suggestions, and advise the user to consult a qualified medical professional for confirmation and treatment.
- The app must monitor and update its diagnosis and suggestion algorithms regularly to ensure accuracy and reliability.
- The app must provide a feedback mechanism for the user to report any errors or issues with the app's functionality or performance.

## **2.5 SYSTEM REQUIREMENTS**

### 2.5.1 SOFTWARE REQUIREMENTS:

- XCode and iOS SDK
- Android Studio and Android Emulator
- Flutter SDK

### 2.5.2 DATABASE REQUIREMENTS:

- Firebase as a service is used as the backend.
- Cloud Firestore is the choice of database that would be used for the Application.

### 2.5.3 HARDWARE REQUIREMENTS

- There are no specific hardware requirements - an Android and iOS device is only needed to run, test and use the app.

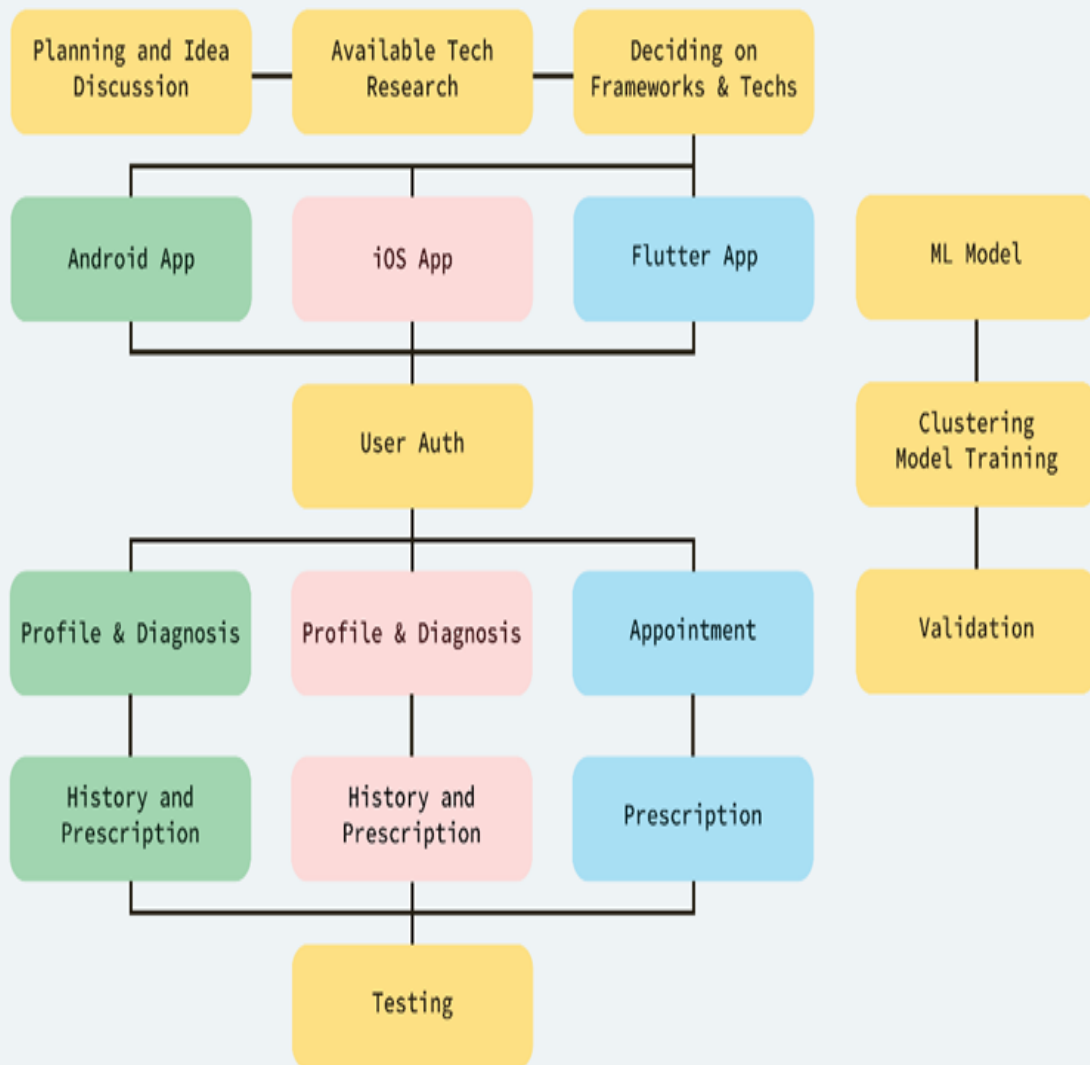
## **2.6 SYSTEM IMPLEMENTATION PLAN**

Our system consists of two projects - the doctor's app and the patient's app. These apps communicate with each other through a central database which is Firestore. The development of the doctor's app and patient's app had started concurrently. In each of these parts, the authentication screens were built first, following other screens and their respective backend logic.

Next, integration of both apps was performed with Firebase so that both apps can send information to the database to bring the required functionality.

### 3. SYSTEM DESIGN

#### 3.1. Flow chart and activity diagrams



## **3.2. Implementation**

### **3.2.1. Design of the Patient's App**

The patient's app is an Android app that is used to provide automated diagnosis along with options to view past diagnosis, medications and profile. It uses Firebase for email-password authentication and also as a primary database. The tools we have used for building the apps are,

1. XCode/iOS development SDK: It is a package that provides all fundamental tools to build and package iOS applications. XCode is an IDE for iOS development maintained by Apple. The apps are built using the Swift programming language - which supports multiple programming paradigms.
2. Swift UI - It is a declarative UI framework for developing user-interfaces in iOS applications with Swift
3. Android app development kit: It is a software development kit used for building Android applications and building and running them on various devices. The apps are built with Kotlin - a programming language which also supports multiple programming paradigms.
4. Jetpack Compose: Similar to SwiftUI, it is a declarative UI framework written in Kotlin and is used to build UIs in Android applications
5. Firebase - It is a platform that provides various cloud-based services like authentication, database, storage and functions.

In this section, we describe the design and implementation of each component of the patient's app in detail:

#### **3.2.1.1. User Authentication:**

The user authentication component is responsible for taking user details and sending them to Firebase for authentication. This component has two parts, the login screen and the register screen. The login screen takes the email address and corresponding password from the user and uses Firebase Authentication to validate the email and password pair. If the password is incorrect or the email address does not exist in the user database, an error response is generated and shown to the user as an alert message.

The register screen takes the following details from the user: email address, password, first name, last name, phone number, gender, blood group, weight, height and date of birth. These details are stored in the Firestore database and a new user is created with the given email and password.

The image displays two screenshots of the MedSwift+ mobile application interface.

**Left Screenshot (20:05):** Shows the login/register screen. The MedSwift+ logo is prominently displayed. Below it, there are two input fields: "Enter your EmailID" (with an envelope icon) and "Password" (with a lock icon and a toggle for visibility). A blue arrow points from the "Enter your EmailID" field to the "Register here" button below it.

**Right Screenshot (19:55):** Shows the "Register" screen. The title "Register" is at the top. Below it, there are several input fields: "Email Address" (with an envelope icon), "Password" (with a lock icon and a toggle for visibility), "First Name", "Last Name", "Phone Number" (with a phone icon), "Gender" (with a dropdown menu showing "Male"), and "Blood Group" (with a dropdown menu showing "A +ve").

### 3.2.1.2. Symptom Selection:

A list of predefined symptoms can be selected by the user according to which the possible diseases are shown. The symptoms are parsed from a JSON file that has the following structure for each symptom:

```
"text": "Any recent nasal congestion, or runny nose present?",
"laytext": "Are you having congested or runny nose?",
"name": "RunnyNoseCongestion",
"type": "categorical",
"default": 2,
"choices": [
  {
    "text": "Data unavailable (i.e. unable to assess).",
    "laytext": "Skip this question.",
    "value": 1,
    "relatedanswertag": null
  },
  {
    "text": "No.",
    "laytext": "No.",
    "value": 2
  },
  {
    "text": "Yes.",
    "laytext": "Yes.",
    "value": 3
  }
]
1,
```

These symptoms are parsed with the Kotlin serialization library into an Kotlin class, that has an identity schema:

```
@kotlinx.serialization.Serializable
data class Symptom(
    val text : String,
    val laytext : String,
    val name : String,
    val type : SymptomChoiceType,
    val min : Double? = null,
    val max : Double? = null,
    val default : Double,
    val category : String,
    val alias : String,
    val wiki : String,
    val wiki2 : String,
    val wiki3 : String,
    val wiki4 : String,
    val subcategory1 : String ,
    val subcategory2 : String ,
    val subcategory3 : String ,
    val subcategory4 : String ,
    val IsPatientProvided : Boolean,
    val step : Double? = null,
    val choices : Array<SymptomChoice>? = null ,
    @Transient var isUserSelected : Boolean = false
)
```

These symptoms are then displayed as a list to the user. Once a symptom is clicked, it is added to another list of selected symptoms which are sent to the API endpoint for getting predictions:

### 3.2.1.3. Automated Diagnosis:

10:06 10:06

VoLTE 45%

runn

**Selected Symptoms**

**All Symptoms**

Any recent nasal congestion, or runny nose present?

Are headaches associated with ipsilateral lacrimation, conjunctival infection, nasal congestion, rhinorrhea, frontal or facial swelling, eyelid edema, miosis or ptosis?

Have the headaches been associated with tearing of the eye, runny nose, or nasal congestion, face or forehead swelling, upper eyelid drop for eyelid swelling felt or seen on the same side as a headache?

Are headaches associated with ipsilateral lacrimation, conjunctival infection, nasal congestion, rhinorrhea, frontal or facial swelling, eyelid edema, miosis or ptosis?

☒ Data unavailable (i.e. unable to assess).

☐ No headaches at all.

☐ No.

☐ Yes.

III O <

The app uses the EndlessMedicalAPI to get automated diagnosis. The API is free for public use and the following steps are initiated while getting automated diagnosis from the API:

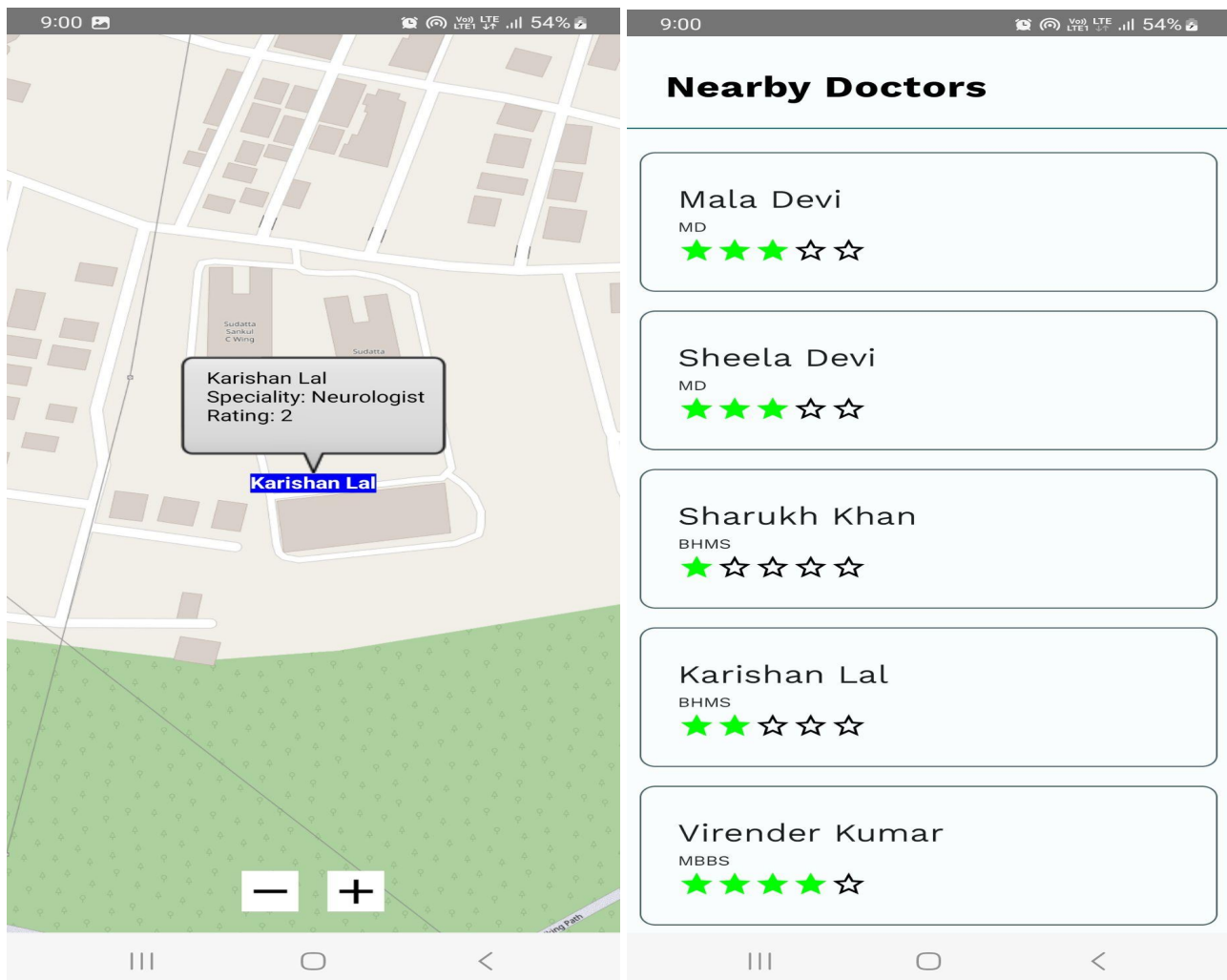
1. Getting the session ID by making a GET request to the host URL [api.endlessmedical.com](https://api.endlessmedical.com)



2. After getting the session ID, the API's terms and conditions need to be accepted by providing a written justification regarding the use of the API.
3. The selected symptoms are then uploaded as (name, value) pairs using the session ID received from step 1.
4. The analyze request returns a JSON mapping of predicted diseases and their confidence.






#### 3.2.1.4. Displaying a list of nearby doctors

The app can fetch a list of nearby doctors from Firebase. To measure the distance, the Euclidean distance formula is used and the doctors are sorted accordingly. The location is displayed with OpenStreetMaps.



### 3.2.2. Design of the Doctor's App

This app in the MedSwift system is meant for Doctors. It helps to streamline the process of online medical consultation and assessment. It offers a user-friendly interface where doctors can create accounts seamlessly and access the reports that patients have sent them through a dedicated dashboard. Patients can submit reports detailing their symptoms, which are accompanied by auto-generated predictions regarding potential health issues. Doctors can then review this information, provide their own assessment, conclusions, and prescriptions, and recommend medical tests if necessary.

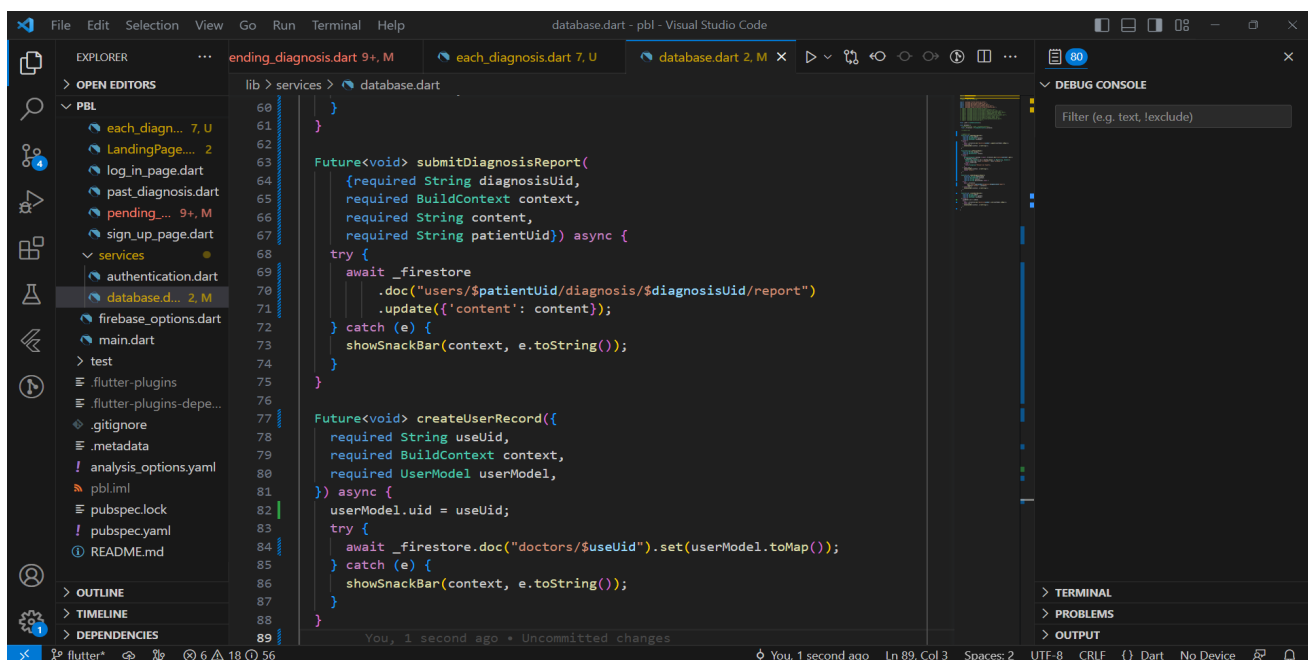
Pending diagnosis	Diagnosis
 Patient Name: Chinmay Patil 20/5/23   11:37 am >	 Patient Name: Soham Pimparkar
 Patient Name: Shubham Panchal 20/5/23   10:37 am >	Symptoms: <ul style="list-style-type: none"><li>⋮ Head ache (frontal)</li><li>⋮ Runny Nose</li><li>⋮ Nausea</li></ul>
 Patient Name: Advait Naik 19/5/23   9:14 am >	Predictions: <ul style="list-style-type: none"><li>⋮ Chronic Sinusitis</li><li>⋮ Influenza</li><li>⋮ Corona Virus Disease 2019</li></ul>
 Patient Name: Chinmay Patil 19/5/23   8:45 am >	

### 3.2.2.1 Features:

- User accounts: The app facilitates easy and secure creation of accounts and maintains an exhaustive profile of the doctors along with their qualifications, ratings, specializations, and history.
- Data security: The app uses Google Firebase for storing data and authentication ensuring the security and privacy of users' information.
- Dashboard: This is the landing page of the app and shows a brief summary of the doctor's activity on Medswift, links to pending and past diagnosis reports, alerts, etc.
- Pending and previous diagnoses: These pages show a list of reports sent by patients which are to be/ have been assessed by the doctor.
- Diagnosis report page: The doctor can read the patient's information, the symptoms and auto-generated predictions and write the patient a report based on his or her judgment.

### 3.2.2.2 Technical Overview:

The app follows an MVC architecture and is built using the Flutter framework, allowing code reuse across multiple platforms. The app supports Android, iOS, and web platforms, extending its reach to a diverse user base. Firebase serves as the backend infrastructure, offering seamless data synchronization and real-time updates. Firestore, a NoSQL database, is used to store and retrieve data related to doctors, patients, medical reports, and diagnosis information. Additionally, Firebase Authentication ensures secure user authentication and authorization.



The screenshot displays the Visual Studio Code interface for a Flutter project. The Explorer panel on the left shows the project structure, including the 'lib' directory with 'services' and 'database.dart'. The main editor area shows the 'database.dart' file with the following Dart code:

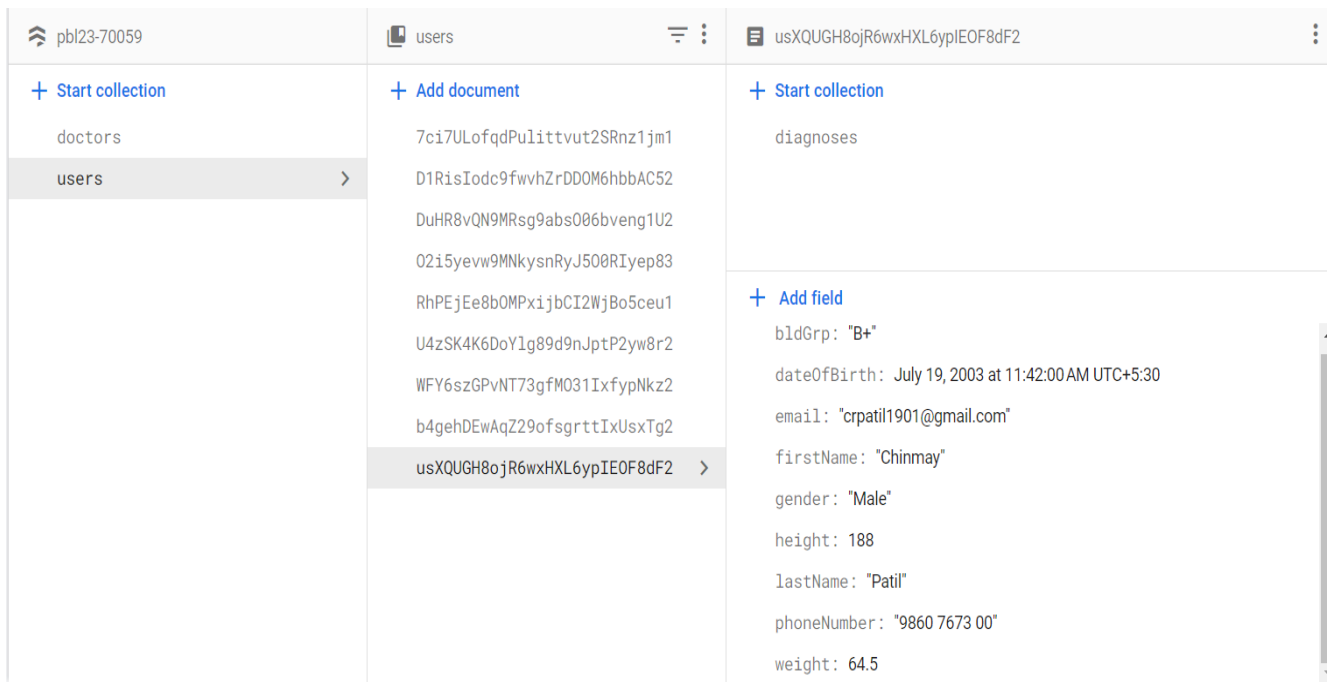
```
Future<void> submitDiagnosisReport(  
  required String diagnosisUid,  
  required BuildContext context,  
  required String content,  
  required String patientUid) async {  
  try {  
    await _firestore  
      .doc("users/$patientUid/diagnosis/$diagnosisUid/report")  
      .update({'content': content});  
  } catch (e) {  
    showSnackBar(context, e.toString());  
  }  
}  
  
Future<void> createUserRecord({  
  required String useUid,  
  required BuildContext context,  
  required UserModel userModel,  
}) async {  
  userModel.uid = useUid;  
  try {  
    await _firestore.doc("doctors/$useUid").set(userModel.toMap());  
  } catch (e) {  
    showSnackBar(context, e.toString());  
  }  
}
```

The bottom status bar indicates the current file is 'database.dart' at line 89, column 3, with 2 spaces, UTF-8 encoding, CRLF line endings, and Dart language.

### 3.2.3. Database

Firebase is a mobile and web application development platform that provides a variety of services to help developers build high-quality apps. One of the services provided by Firebase is Cloud Firestore, which is a NoSQL document database that provides services like store, sync, and query through the application on a global scale

Cloud Firestore stores data in the form of objects also known as Documents. It has a key-value pair and can store all kinds of data like strings, binary data, and even JSON trees. Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps data in sync across clients.



For Android apps, we need to create a Firebase project in the Firebase console and register the app with a unique package name. Then, we need to download a google-services.json file from the console and place it in the app folder. Next, we need to add the Firebase SDK dependencies to the app/build.gradle file and sync the project. Finally, we need to initialize Firebase in our app by calling `FirebaseApp.initializeApp(this)` in the Application class or MainActivity.

For iOS apps, we need to create a Firebase project in the Firebase console and register our app with a unique bundle ID. Then, we need to download a GoogleService-Info.plist file from the console and add it to the Xcode project. Next, we need to add the Firebase SDK dependencies to our Podfile and run pod install. Finally, we need to initialize Firebase in the app by importing Firebase in the AppDelegate.swift file and calling FirebaseApp.configure() in the didFinishLaunchingWithOptions method.

## **4. OTHER SPECIFICATIONS**

### **4.1. Advantages**

1. Offering free of charge medical diagnosis for symptoms or conditions that do not require urgent or specialized care
2. Managing prescriptions and providing reminders for users to adhere to their medication regimen
3. Providing information on the availability and quality of healthcare facilities in a user's vicinity
4. Validating symptoms and the corresponding automated diagnosis to ensure the accuracy and reliability of the diagnosis process

### **4.2. Limitations**

1. The app may not be able to diagnose rare or complex conditions that require physical examination or laboratory tests.
2. The app may not be able to account for individual factors such as medical history, allergies, medications, or lifestyle that may affect the diagnosis or treatment.
3. The app may not be able to guarantee the accuracy, reliability, or availability of the information provided by the app or by the suggested doctors.
4. The app may not be able to protect the privacy and security of the user's personal and health data from unauthorized access or misuse.
5. The app may not be able to replace the professional judgment and advice of a qualified healthcare provider.
6. Lack of Physical Examination: The app's diagnosis is based solely on user-entered symptoms and information, without the ability to conduct a physical examination. Physical examinations, which involve observations, palpitations, and other hands-on assessments by a healthcare professional, can provide critical insights for accurate diagnosis. Without physical examination, the app's diagnostic capabilities may be limited.

7. Limited Contextual Understanding: The app may struggle to fully understand the context surrounding a user's symptoms. Factors such as medical history, pre-existing conditions, medication use, lifestyle, and environmental factors are crucial for accurate diagnosis. Without comprehensive context, the app's suggestions may not fully consider these important variables.

### **4.3. Applications**

Our application is designed to facilitate the diagnosis and treatment of common health conditions. It allows patients who experience symptoms related to these conditions to access qualified medical professionals in their vicinity. Doctors who wish to offer their services through our platform can register themselves and provide relevant information about their qualifications, expertise and clinic location. The application has two modes: a 'patient mode' that can be used by anyone who seeks medical assistance, and a 'doctor mode' that can only be accessed by verified doctors who have completed the registration process.

## 5. CONCLUSIONS AND FUTURE WORK

The future scope for an app that provides automatic diagnosis for symptoms entered by the user and suggests nearby doctors can be quite promising. Here are some potential areas of improvement and expansion for such an app:

**Enhanced Symptom Analysis:** Improve the app's symptom analysis capabilities by incorporating more sophisticated machine learning algorithms. This could involve leveraging deep learning techniques or natural language processing to better understand user input and provide more accurate diagnoses.

**Integration with Medical Databases:** Integrate the app with extensive medical databases and research studies to enhance its diagnostic accuracy. Access to a vast pool of medical knowledge can help the app consider a wide range of conditions and symptoms, leading to more precise diagnoses.

**Personalized Recommendations:** Develop the app to provide personalized recommendations based on the user's medical history, demographics, and lifestyle factors. By considering individual traits and circumstances, the app can offer tailored advice and suggestions for healthcare providers.

**Telemedicine Integration:** Incorporate telemedicine functionality within the app to enable users to have virtual consultations with healthcare professionals. This would allow users to connect with doctors remotely, saving time and improving accessibility, particularly for individuals in remote areas or with limited mobility.

**Health Monitoring and Tracking:** Integrate features for health monitoring and tracking within the app. Users can input data such as vital signs, medication history, or lifestyle habits, and the app can analyze this information to provide more accurate diagnoses and recommendations.



**Seamless Electronic Health Records (EHR) Integration:** Enable seamless integration with electronic health records (EHR) systems. By securely accessing a user's medical history and records, the app can gain a comprehensive understanding of their health profile, aiding in diagnosis and treatment recommendations.

**Machine Learning-Based Continuous Improvement:** Continuously update and refine the app's algorithms through machine learning techniques. By analyzing user feedback, outcomes of medical treatments, and ongoing medical research, the app can improve its accuracy and adapt to new medical findings.

**Collaborative Diagnosis:** Implement a collaborative approach where the app enables users to seek opinions and second opinions from multiple doctors or medical specialists. This can be facilitated through secure communication channels within the app.

**Health Education and Awareness:** Expand the app's features to include educational content on various health conditions, preventive measures, and lifestyle tips. By promoting health literacy, the app can empower users to make informed decisions about their well-being.

**Internationalization and Localization:** Adapt the app for different languages and localize it to cater to users from various regions. This would enable wider adoption and usability on a global scale.

With the help of Android and Flutter frameworks, we aim to provide an app that could provide quick medical diagnosis to users and give basic information on the local healthcare infrastructure and reminders for prescriptions. We wish to make the project open-source and follow all software-development practices so facilitate project-based learning.

## 6. REFERENCES

- Automated Diagnosis: <https://endlessmedical.com>
- Jetpack Compose: <https://developer.android.com/jetpack/compose>
- iOS Development: <https://developer.apple.com/tutorials/app-dev-training#swiftui-essentials>
- Flutter Development: <https://docs.flutter.dev>
- <https://www.ndtv.com/health/year-ender-2020-the-rise-of-digital-healthcare-services-a-doctors-take-2339609>
- <https://www.ndtv.com/india-news/online-health-start-ups-e-connect-patients-doctors-across-india-1457260>
- <https://www.ndtv.com/video/news/banega-swasth-india/the-rural-india-healthcare-crisis-shortage-of-doctors-630067>
- <https://www.forbesindia.com/article/take-one-big-story-of-the-day/tech-for-health-inside-the-rise-of-practo-and-its-ambitious-roadmap-for-the-future/66485/1>
- Firebase CloudStore: <https://firebase.google.com/docs/firestore>
- Open Street Map: <https://github.com/osmdroid/osmdroid>