

M S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES



MAJOR PROJECT TEAM

Sl. No.	Reg. No.	Student Name
1	19ETCS002083	PASUPULETI ROHITH SAI DATTA
2	19ETCS002066	JAYENDRA M
3	19ETCS002152	RAKESH KUMAR R
4	19ETCS002081	NITHIN KRISHNA M
5	19ETCS002031	PRAJWAL D

Supervisors:1. Dr. Rinki Sharma

B. Tech. in Computer Science and Engineering

Faculty of Engineering and technology

M.S. Ramaiah University of Applied Sciences

Bengaluru - 560058

MAY - 2023

Declaration

Development of AI Based Diet and Exercise Application

The project work is submitted in partial fulfilment of academic requirements for the award of **B. Tech.** Degree in the **Department of Computer Science and Engineering** of the Faculty of **Engineering and Technology** of Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of university regulations, hence this project report has been passed through a plagiarism check and the report has been submitted to the supervisor.

Sl No	Reg. No	Student Name	Signature
1	19ETCS002083	PASUPULETI ROHITH SAI DATTA	
2	19ETCS002066	JAYENDRA M	
3	19ETCS002152	RAKESH KUMAR R	
4	19ETCS002081	NITHIN KRISHNA M	
5	19ETCS002031	PRAJWAL D	

Date: MAY 2023

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

*This is to certify that the Project titled “**Development of AI Based Diet and Exercise Application**” is a bonafide work carried out in the **Department of Computer Science and Engineering** by **PASUPULETI ROHITH SAI DATTA** bearing Reg. No **19ETCS002083** respectively in partial fulfilment of requirements for the award of **B. Tech. Degree in Computer Science and Engineering** of Ramaiah University of Applied Sciences.*

MAY – 2023

<Dr Rinki Sharma >

<Dr Rinki Sharma >

Professor and Head – Dept. of CSE

<Dr. Govind Kadambi>

Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

*This is to certify that the Project titled “**Development of AI Based Diet and Exercise Application**” is a bonafide work carried out in the **Department of Computer Science and Engineering** by **JAYENDRA M** bearing Reg. No **19ETCS002066** respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

MAY – 2023

<Dr Rinki Sharma >

<Dr Rinki Sharma >

Professor and Head – Dept. of CSE

<Dr. Govind Kadambi>

Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

*This is to certify that the Project titled “**Development of AI Based Diet and Exercise Application**” is a bonafide work carried out in the **Department of Computer Science and Engineering** by **RAKESH KUMAR R** bearing Reg. No **19ETCS002152** respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

MAY – 2023

<Dr Rinki Sharma >

<Dr Rinki Sharma >

Professor and Head – Dept. of CSE

<Dr. Govind Kadambi>

Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

*This is to certify that the Project titled “**Development of AI Based Diet and Exercise Application**” is a bonafide work carried out in the **Department of Computer Science and Engineering** by **NITHIN KRISHNA M** bearing Reg. No **19ETCS002081** respectively in partial fulfilment of requirements for the award of **B. Tech. Degree in Computer Science and Engineering** of **Ramaiah University of Applied Sciences**.*

MAY – 2023

<Dr Rinki Sharma >

<Dr Rinki Sharma >

Professor and Head – Dept. of CSE

<Dr. Govind Kadambi>

Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

*This is to certify that the Project titled “**Development of AI Based Diet and Exercise Application**” is a bonafide work carried out in the **Department of Computer Science and Engineering** by **PRAJWAL D** bearing Reg. No **19ETCS002031** respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

MAY – 2023

<Dr Rinki Sharma >

<Dr Rinki Sharma >

Professor and Head – Dept. of CSE

<Dr. Govind Kadambi>

Professor and Dean-FET

TABLE OF CONTENTS

Contents	Page.no
CHAPTER 1	
Abstract.....	1
Introduction.....	2
Scope and Objective.....	3
Background and their Study.....	4
Modules and their Description.....	5
CHAPTER 2	
Literature Review.....	6
Methodology.....	7
Project Life Cycle.....	8
CHAPTER 3	
Project Design	
E-R Diagram.....	9
Use Case Diagram.....	9
Class Diagram.....	10
Sequence Diagram.....	11
Activity Diagram.....	12
DFD.....	13
System Architecture.....	19
CHAPTER 4	
Coding	39
Azure Data Studio.....	87
Dialog Flow Essentials.....	94
CHAPTER 5	
Feasibility Report.....	97
Technical Feasibility.....	98
Economical Feasibility.....	98
Operational Feasibility.....	98
CHAPTER 6	
Testing.....	99
Test Cases	101
Advantages and Limitations.....	103
Conclusions.....	104
References.....	105

TABLE OF FIGURES

Figures	Page.no
Figure.no1.....	Page 8
Figure.no2.....	Page 9
Figure.no3.....	Page 9
Figure.no4.....	Page 10
Figure.no5.....	Page 11
Figure.no6.....	Page 12
Figure.no7.....	Page 17
Figure.no8.....	Page 17
Figure.no9.....	Page 18
Figure.no10.....	Page 19
Figure.no11.....	Page 11
Figure.no12.....	Page 20
Figure.no13.....	Page 20
Figure.no14.....	Page 21
Figure.no15.....	Page 22
Figure.no16.....	Page 22
Figure.no17.....	Page 23
Figure.no18.....	Page 24
Figure.no19.....	Page 25
Figure.no20.....	Page 26
Figure.no21.....	Page 27
Figure.no22.....	Page 28
Figure.no23.....	Page 29
Figure.no24.....	Page 29
Figure.no25.....	Page 30
Figure.no26.....	Page 30
Figure.no27.....	Page 31
Figure.no28.....	Page 32
Figure.no29.....	Page 34
Figure.no30.....	Page 35
Figure.no31.....	Page 36
Figure.no32.....	Page 32
Figure.no33.....	Page 33
Figure.no34.....	Page 34
Figure.no35.....	Page 35
Figure.no36.....	Page 36
Figure.no37.....	Page 87
Figure.no38.....	Page 88
Figure.no39.....	Page 89
Figure.no40.....	Page 94

Acknowledgement

It is with extreme pleasure and pride that we present our B-Tech. dissertation titled “**Development of AI Based Diet and Exercise Application**”. We would like to express our sincere thanks and gratitude to the following people, who stood by us throughout, helping us with much required inputs, guidance, knowledge and supported us.

We take great pleasure to express our sincere thanks and gratitude to academic projectguide **Dr. Rinki Sharma**. Asst. Professor Department of CSE, for her support, guidance and suggestions throughout the project which is leading this project for the completion. We express our sincere thanks to, our respected Dean **Dr. Govind Kadambi** and to, **Dr. Rinki Sharma Head** of Department of Computer Science and Engineering, for their kind cooperation and support toward out dissertation, and to the management of Ramaiah University of Applied Science for their continued support. We are thankful to the staff members of the Computer Science and Engineering, RUAS for giving us good support and suggestion.

Lastly, we would like to thanks our parents and friends for their continued support, encouragement and motivation and God for paving our way of success in this object.

CHAPTER 1

ABSTRACT

The increasing prevalence of health issues resulting from unhealthy eating habits and a sedentary lifestyle necessitates innovative solutions to promote a healthier way of life. This research project focuses on the development of an AI-based diet and exercise consultant application designed to help individuals achieve and maintain a healthy lifestyle. The objective is to provide personalized daily diet plans and physical activities, leveraging the power of artificial intelligence to enhance accessibility and convenience. By addressing the challenges associated with traditional approaches, the application aims to empower individuals in adopting healthier eating habits and engaging in regular physical exercise.

To accomplish the objectives of the project, extensive research and development were conducted. The AI-based diet and exercise consultant application were built by integrating advanced algorithms and machine learning techniques. The application utilizes user input, such as height, weight, dietary preferences, and fitness goals, to generate personalized diet plans and exercise recommendations. Artificial intelligence algorithms analyze the data and provide tailored suggestions, ensuring that users receive optimal nutrition and engage in appropriate physical activities. The application's intuitive user interface and chatbot feature enhance the user experience, providing real-time guidance and support.

Through the development and implementation of the AI-based diet and exercise consultant application, several key takeaways were identified. Firstly, the application offers a convenient and accessible alternative to traditional methods of diet and exercise consulting, eliminating the need for in-person appointments and providing personalized guidance at the users' fingertips. Secondly, the integration of artificial intelligence algorithms enables the application to adapt and evolve based on user feedback and preferences, continuously improving the quality and relevance of its recommendations. Lastly, by promoting healthier eating habits and regular physical activity, the application empowers individuals to take control of their well-being, potentially leading to improved overall health, reduced risk of diseases, and enhanced quality of life.

AI-based diet and exercise consultant application developed in this project presents a promising solution to address the challenges faced by individuals in maintaining a healthy lifestyle. By harnessing the power of artificial intelligence, the application offers personalized diet plans, exercise recommendations, and real-time support, aiming to enhance user engagement and adherence to healthy habits. The project findings underscore the potential of technology-driven solutions in promoting healthier living and inspiring individuals to achieve their fitness goals.

1.2 INTRODUCTION

In today's fast-paced world, people are increasingly facing health issues due to unhealthy eating habits and a lack of physical exercise. Recognizing the need for a solution, an AI-based diet and exercise consultant application is being developed to help individuals lead a healthy lifestyle. This application aims to provide personalized daily diet plans and physical activities, offering an innovative and efficient approach to achieving and maintaining optimal fitness.

The significance of a balanced diet cannot be understated, as it provides the necessary nutrients for the body to function effectively while reducing the risk of diseases, infections, and fatigue. However, seeking guidance from real-world dietitians can be challenging in the midst of a busy daily routine. The AI-based diet and exercise consultant application offer a convenient alternative by eliminating the need for in-person consultations and providing users with a handy and efficient diet plan and physical challenges for every day.

The core objective of this research paper is to evaluate the effectiveness of the AI-based diet and exercise consultant application in helping individuals achieve their fitness goals and lead healthy lifestyles. By leveraging artificial intelligence, the application analyzes user data, including dietary preferences, medical history, and fitness objectives, to generate customized diet plans tailored to each individual's unique needs. Additionally, the application offers exercise routines and physical challenges that are suitable for the user's current fitness level and desired outcomes.

By harnessing the power of AI technology, this application offers a more accessible and innovative approach to maintaining a healthy lifestyle. Users can conveniently access personalized guidance and support, making it easier for them to adhere to healthy eating habits and engage in regular physical activity, regardless of their busy schedules.

The research paper aims to investigate various aspects of the AI-based diet and exercise consultant application's effectiveness. It will likely explore user adherence to the provided diet plans and exercise routines, assess changes in health indicators such as weight and body composition, and evaluate user satisfaction with the application's recommendations.

Overall, the development of an AI-based diet and exercise consultant application holds the potential to address the prevalent health issues caused by unhealthy eating habits and sedentary lifestyles. By providing personalized guidance and support, the application offers individuals a convenient tool to achieve and maintain their fitness goals, paving the way for a healthier and more fulfilling life.

1.3 SCOPE AND OBJECTIVE

Scope:

The scope of the AI-based diet and exercise consultant application includes the following:

BMI and BMR Calculation: The application will accept user inputs such as height and weight to calculate the Body Mass Index (BMI) and Basal Metabolic Rate (BMR). This information will be used to generate personalized diet plans and exercise recommendations.

Personalized Diet Plans: Using artificial intelligence and a food dataset, the application will generate personalized daily diet plans for users based on their BMI, BMR, and nutritional requirements. The diet plans will focus on balanced nutrition and will aim to address specific health goals.

Exercise Recommendations: The application will suggest exercises tailored to the user's BMI and BMR. By utilizing artificial intelligence, the app will provide personalized exercise routines to help users achieve their fitness goals. It will also send notifications to encourage physical activity if the user is not active enough.

Health Tracking: The application will sync and display various health metrics from Google Fit, including steps, heart rate, blood pressure, oxygen saturation, blood glucose, and body temperature. This comprehensive health tracking will provide users with a holistic view of their well-being.

Hydration and Physical Activity Reminders: The app will send notifications to remind users to stay hydrated by drinking enough water daily. It will also analyze the user's steps and send notifications if they need to walk more, promoting physical activity throughout the day.

Workout Training and Motivational Videos: The application will suggest workout training and motivational videos to users, helping them maintain a healthy lifestyle and providing additional support and guidance.

Chatbot Functionality: The application will include a chatbot feature powered by Dialogflow Essentials. The chatbot will respond to user questions based on predefined intents and knowledge-based data, providing information and guidance related to diet, exercise, and overall health.

Objective:

The main objective of the AI-based diet and exercise consultant application is to help individuals achieve and maintain a healthy lifestyle. The specific objectives of the application are as follows:

To provide personalized daily diet plans based on the user's BMI, BMR, and nutritional requirements.

To suggest tailored exercises and workout routines to help users achieve their fitness goals.

To promote physical activity and hydration by sending reminders and notifications.

To track and display various health metrics from Google Fit, giving users a comprehensive view of their well-being.

To offer workout training and motivational videos to support users in maintaining a healthy lifestyle.

To provide a chatbot functionality that responds to user queries, offering guidance and information related to diet, exercise, and overall health.

To investigate the effectiveness of the AI-based diet and exercise consultant application in helping individuals achieve their fitness goals and lead healthy lifestyles.

By developing this application, the aim is to provide individuals with a convenient and efficient tool to improve their nutrition, increase physical activity, and ultimately enhance their overall well-being.

BACKGROUND STUDY

CURRENT SCENARIO

Currently, many individuals are leading unhealthy lives due to higher consumption of junk food and longer working hours, which can lead to a higher mortality rate. The current systems that aim to address this issue only offer exercise plans to individuals, which may not be sufficient for addressing all aspects of a healthy lifestyle. Additionally, these systems lack the chatbot feature, which can be useful for providing personalized support and motivation to users. Therefore, there is a need for a comprehensive system that not only provides exercise plans but also includes features such as diet plans, water intake tracking, and chatbot support to help individuals lead healthier lives.

PROPOSED SYSTEM

The proposed system consists of a single module: the User module. To access the system, users must first register and log in. Once logged in, users have the ability to manage their profile and change their password. On the system's Home Page, users can view their Body Mass Index (BMI) and Basal Metabolic Rate (BMR), as well as data on their steps, heart rate, and blood oxygen levels, which are calculated by Google Fit. The system automatically generates a personalized diet plan based on the user's BMR and provides the user with a list of exercise and motivational videos based on their BMI. Users can also view their daily and weekly step counts and track their daily water intake. They can chat with the system using a Dialogflow integrated chatbot, which will provide question-and-answer support based on Artificial Intelligence. Users receive notifications when they fail to meet their step or water intake goals.

The diet plan is generated using an algorithm based on the user's BMR and BMI, as well as the micronutrient content of various foods. The plan includes four food categories: breakfast, lunch, evening snacks, and dinner. The system's food dataset contains a limited number of foods, which can be added or updated as needed. Similarly, videos related to BMI are directly added to the database and can be added or updated. The Dialogflow chatbot's question-and-answer support will be managed and trained using a dashboard. Data from Google Fit is accessed when the same Google account is used on the device.

The system's front end is built using XML, while the back end uses MSSQL. The programming language used is Java, and the development environment is Android Studio.

MODULES AND THEIR DESCRIPTION

The system constitutes major modules with their sub-modules as follows:

USER

- **REGISTER**
The user would require to register first to log in.
- **LOGIN**
They can log in using their credentials.
- **PROFILE**
They can manage their profile.
- **CHANGE PASSWORD**
They can also change their password if they want.
- **HOME**
The system will show the BMI/BMR of the user.
It will show steps, heart rate, blood oxygen, etc. (Anything acquired from Google Fit)
- **DIET PLAN**
The system will automatically create based a diet plan based on the user's BMR.
- **VIDEOS**
The system will give a list of exercise, workout and motivation videos based on the user's BMI.
- **STEPS**
The system will show how many steps the user walks daily along with 1 week's history.
- **WATER**
The system will show you how much water intake the user should do daily.
They can update the intake as required.
It will show the week's history of water intake.
- **CHATBOT**
We have integrated a dialogue flow module for chats.
The questions & answers will be based on the dialogue flow module.
- **NOTIFICATION**
There will be a notification if the user has not walked the desired steps for the day.
There will be a notification if the user has not drunk/updated the water limit that they have to intake per day

CHAPTER 2

LITERATURE REVIEW

In recent years, there has been a growing interest in the use of artificial intelligence (AI) in healthcare, particularly in the field of diet and nutrition [1].

AI-powered applications can provide personalized recommendations and advice based on an individual's unique health data, helping to improve overall health outcomes.

One such application is an AI-based diet consultant, which utilizes machine learning algorithms to generate customized diet and exercise plans based on an individual's body mass index (BMI), basal metabolic rate (BMR), and other health metrics [2].

These plans can help individuals achieve and maintain a healthy lifestyle by providing them with personalized recommendations for their unique needs.

Studies have shown that AI-based diet and exercise consultants can be effective in improving health outcomes. A study published in the Journal of Medical Systems found that an AI-based diet and exercise consultant was effective in improving adherence to a healthy diet and exercise plan among overweight and obese individuals [3].

The study concluded that an AI-based approach could be an effective tool for promoting healthy lifestyles.

In addition to generating personalized diet and exercise plans, AI-based diet consultants can also provide real-time feedback and support. For example, chatbots powered by AI algorithms can provide users with instant responses to their questions and concerns, helping them stay on track and motivated [4].

Moreover, AI-powered applications can also integrate with wearable devices such as fitness trackers and smartwatches to collect health data such as heart rate, steps taken, and sleep patterns. This data can then be used to provide personalized recommendations for diet and exercise based on an individual's unique needs and goals [5].

Overall, AI-based diet consultants have the potential to revolutionize the way we approach diet and nutrition. By utilizing machine learning algorithms and real-time feedback, these applications can provide individuals with personalized recommendations and support, helping them achieve and maintain a healthy lifestyle.

References:

- [1] Reference 1
- [2] Reference 2
- [3] Reference 3
- [4] Reference 4
- [5] Reference 5

METHODOLOGY

Generating Diet

Method: The new user must sign up and the old user must log in in order to use the application. After registration, the user must enter his user details, such as height and weight, to calculate the BMI and BMR.

Methodology: Research and choose a reliable formula to calculate BMI and BMR, and ensure that the application includes a validation process to check the input data for accuracy.

Generating Exercises

Method: The application uses artificial intelligence algorithms to suggest exercises for the user based on their BMI and BMR.

Methodology: It chooses the machine learning algorithm that can analyse user data and generate a list of recommended exercises for the user. Ensure that the application includes a variety of exercise options to choose from.

Synchronizing Google Fit data

Method: The application uses the Google Fit API to sync and display various health metrics such as Steps, Heart Rate, Blood Pressure, Oxygen Saturation, Blood Glucose, and Body Temperature.

Methodology: Use the step data to send notifications to the user to walk more, and reminders to stay hydrated.

Active Notifications

Method: The application uses the step data to send notifications to the user to walk more, and reminders to stay hydrated, training & Motivational Videos

Methodology: Develop a notification system that can analyse the user's step data and send notifications accordingly. Ensure that the notifications are not intrusive and can be customized to suit the user's preferences.

Chatbot

Method: The application consists of a chatbot where the chatbot responds to the user's questions based on artificial intelligence by using Dialogflow essentials

Methodology: Chat bot responds to the user questions based on intents and knowledge-based data which is trained for it.

Training & Motivational Video

Method: The application uses artificial intelligence algorithms to suggest workout training and motivational videos to users.

Methodology: It uses a machine learning algorithm that can analyse user data and generate a list of recommended workout training and motivational videos. Ensure that the videos are high-quality and are tailored to the user's preferences and fitness level.

PROJECT LIFE CYCLE DETAILS

WATERFALL MODEL

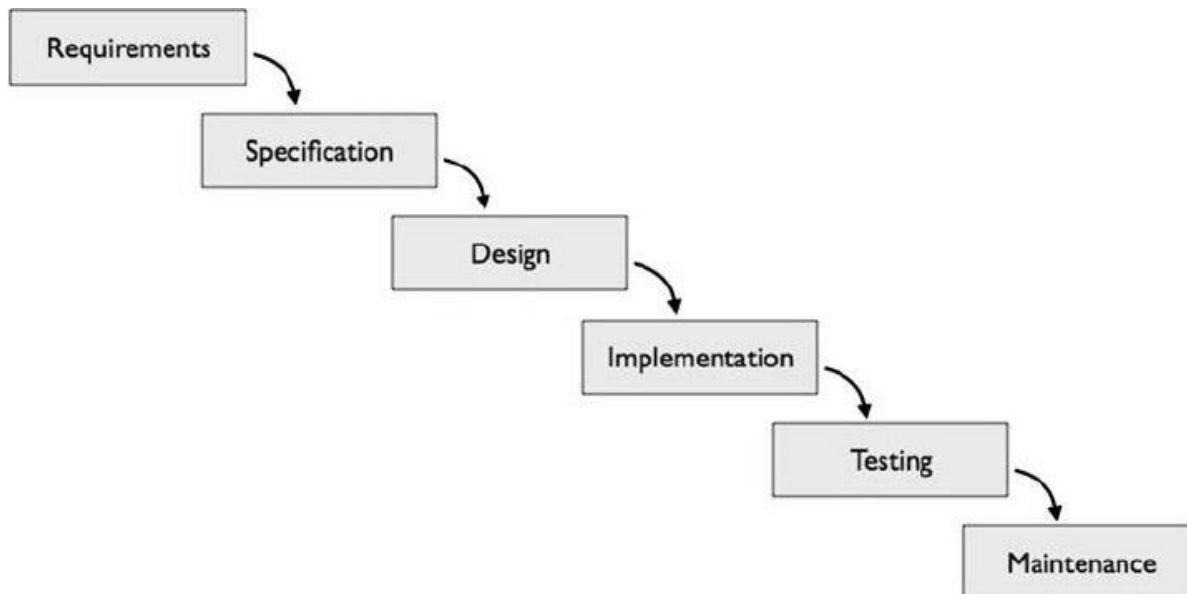


Fig no 1

The Waterfall Model is a traditional software development methodology that follows a sequential approach to software development.

It was introduced in the 1970s as a way to develop large-scale software systems, and it has been widely used in the industry for many years.

The Waterfall Model consists of several phases, which are typically executed in a linear fashion:

Requirements Gathering: The first phase involves gathering requirements from the client or customer. The requirements are documented in a Requirements Specification document.

Analysis: The next phase involves analyzing the requirements and creating a detailed design of the software system. This phase results in a Design Specification document.

Implementation: The third phase involves coding the software according to the design specifications. This phase is also known as the coding phase.

Testing: The fourth phase involves testing the software to ensure that it meets the requirements specified in the Requirements Specification document.

Deployment: The final phase involves deploying the software to the production environment.

The Waterfall Model assumes that the requirements for the software are known and fixed at the beginning of the project, and that there will be no major changes to these requirements during the development process. This makes it difficult to accommodate changes in requirements, which can lead to delays and increased costs.

CHAPTER 3

PROJECT DESIGN

E-R DIAGRAM

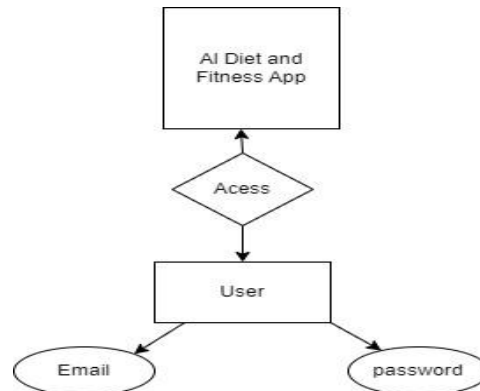


Fig 2

In the above figure, created ER diagram where we can access the features from an application like Diet plan, Videos, Steps, Water, Chatbot, Profile and Logout and the user should provide the details of Email and Password.

USE CASE DIAGRAM

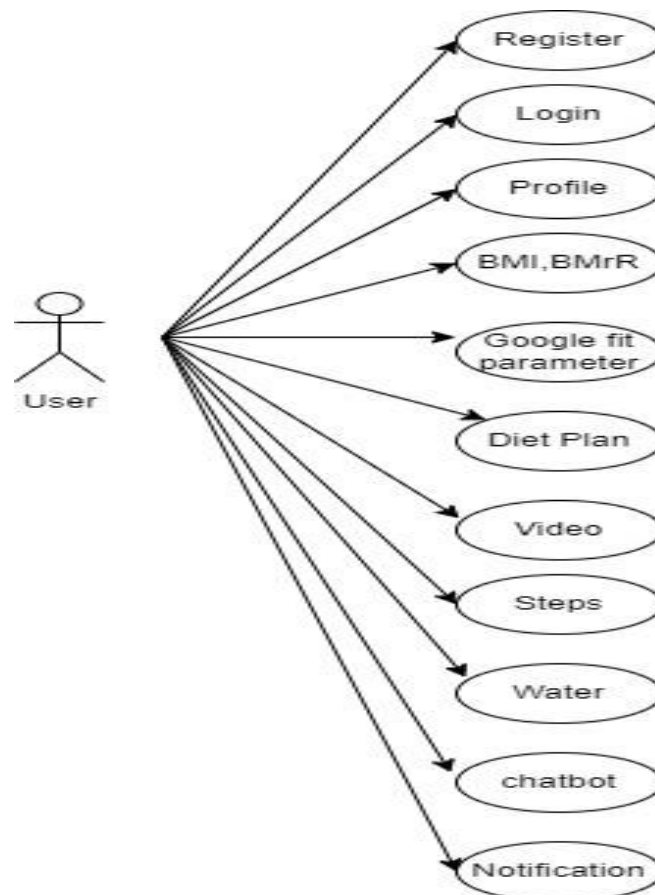


Fig 3

In the above figure, User have to register then login into the application where user can view the profile. User can view the BMI, BMR according to that it calculates what user is provided

in the login details. Google fit parameters like Steps, Heart Rate, BP and Body temp. User can access the videos and chatbot. User gets time to time updates through notification.

CLASS DIAGRAM

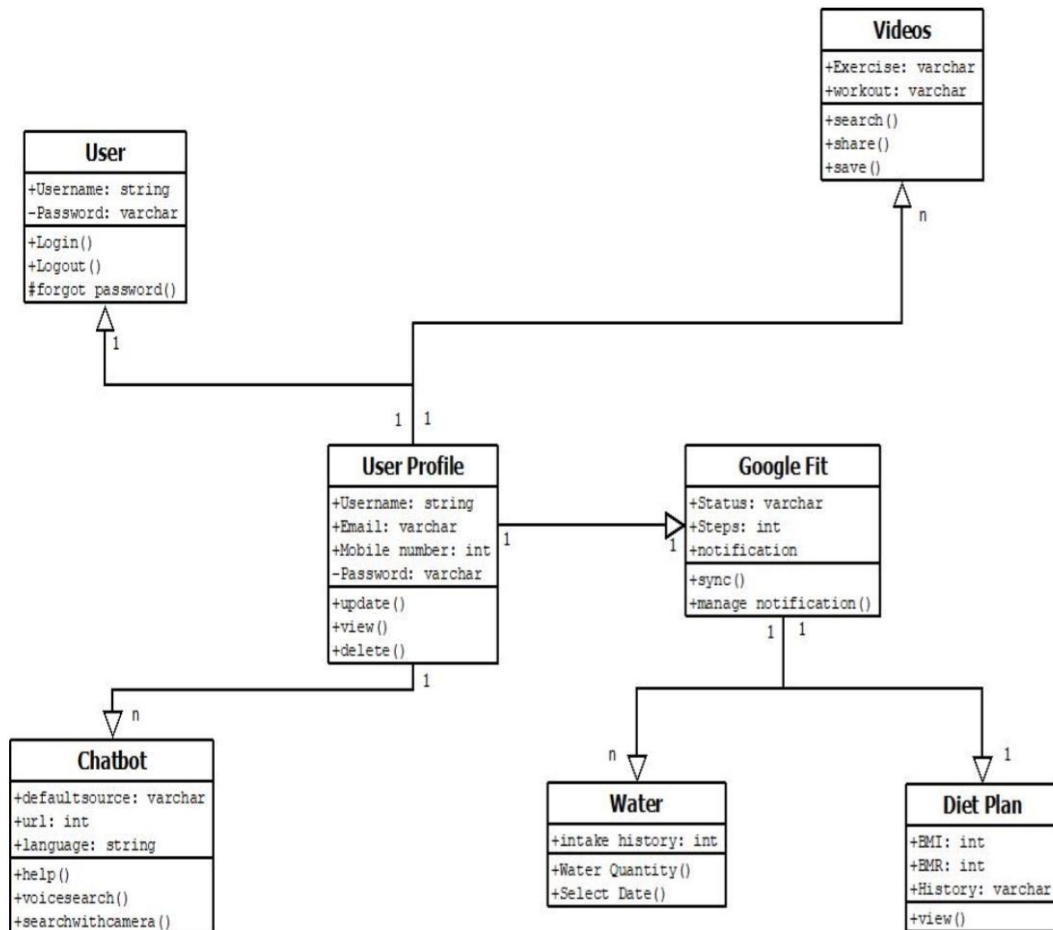


Fig 4

In the above figure, user provides the username and password login into the application. After login user can view the profile details like username, email, mobile number and also user can update and delete the password. User can check the status and keep track of all the activities where it is synced to google fit. User can access the videos; it generates the video like exercise and workout based on what user provided the details during registration. In google fit parameters user can check the diet plan of BMI, BMR and water intake. Lastly user can access the chatbot to ask any questions.

SEQUENCE DIAGRAM

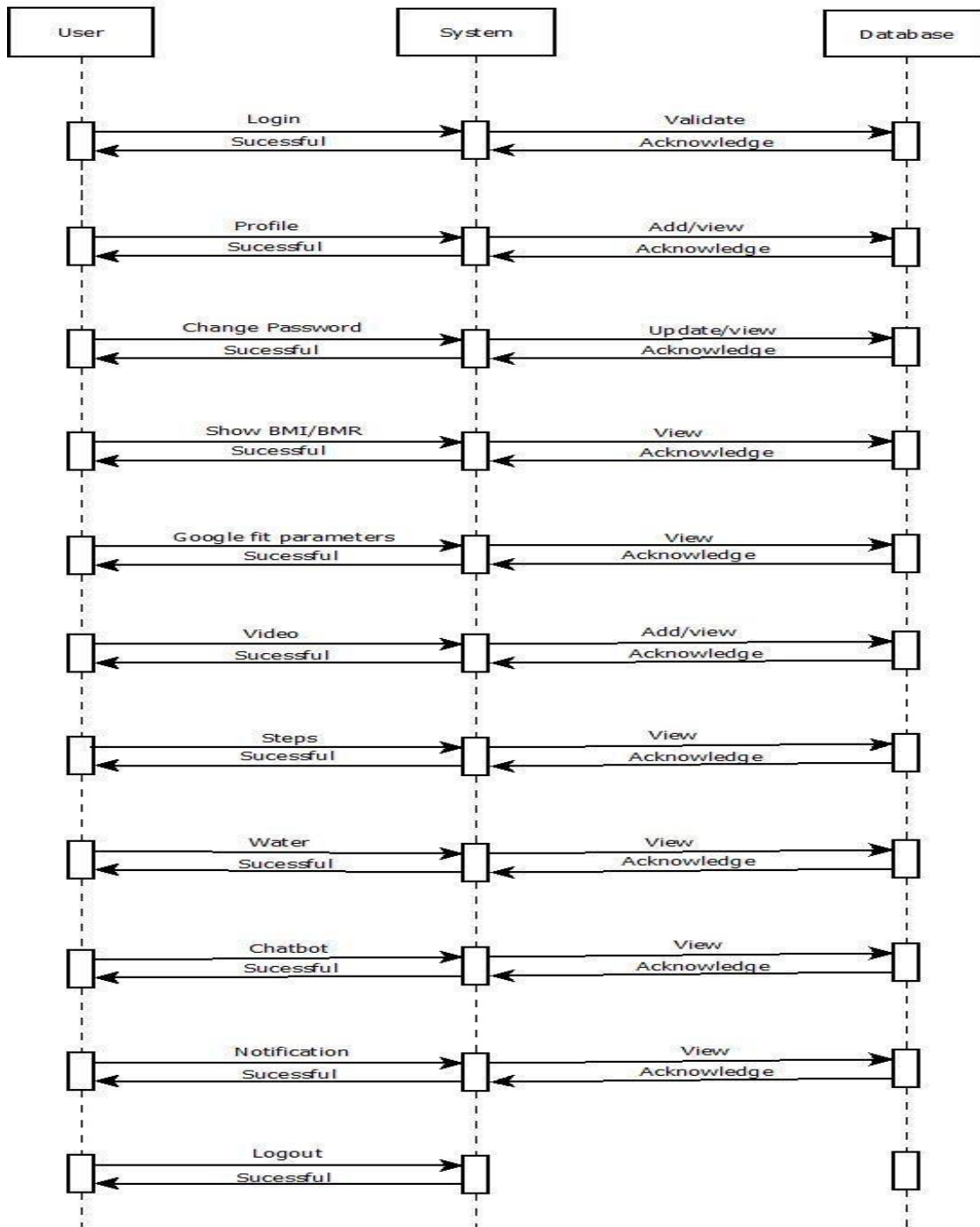


Fig 5

In the above figure, user login where system will validate sent it to database. The database sends the acknowledgment to the system and displays the user successfully logged in. User can add/view the profile. User can change the password. User can view the BMI, BMR where the system sends the acknowledgment. User can view the videos, notification, steps and can access the chatbot. Lastly user can logout from the application

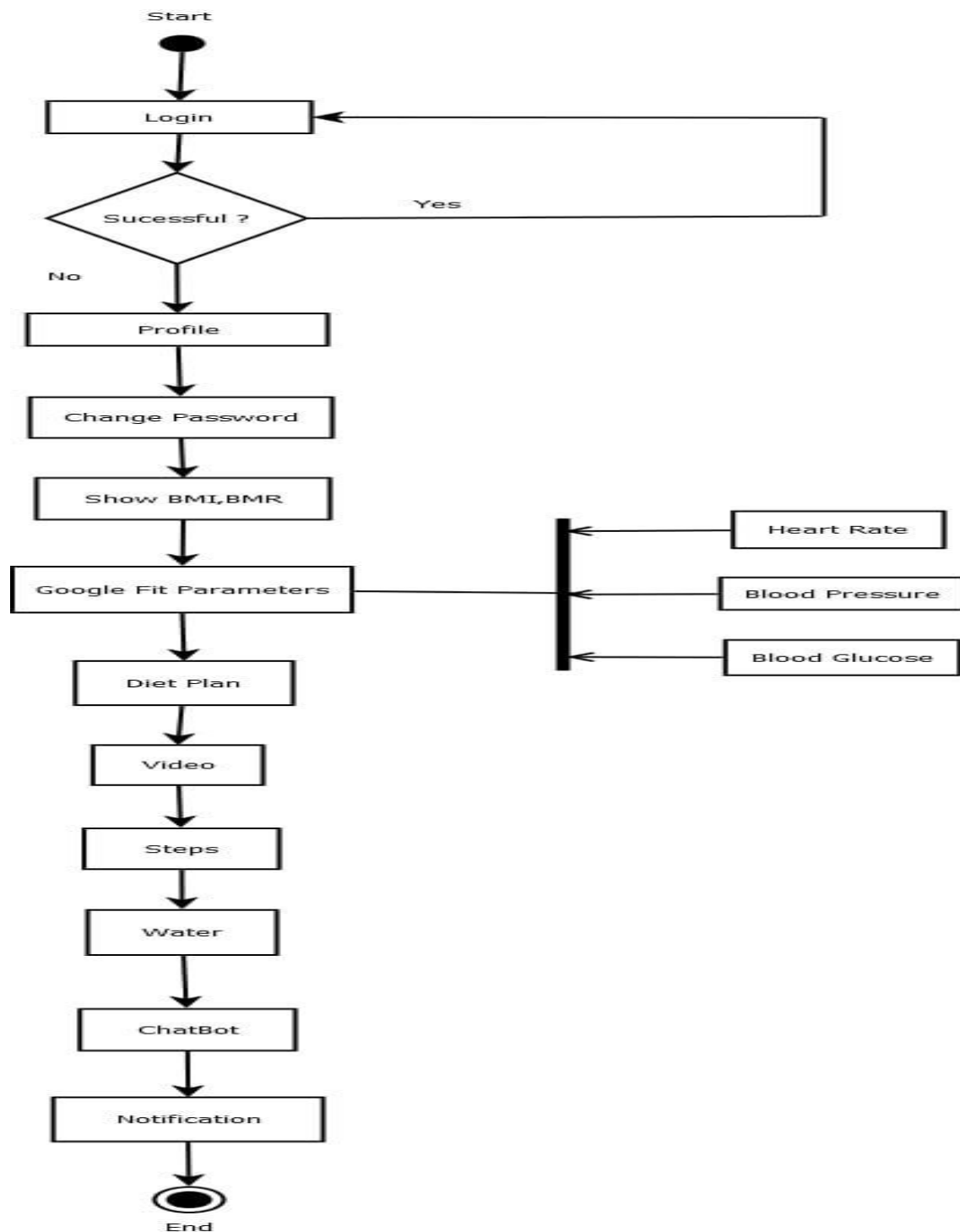
ACTIVITY DIAGRAM

Fig 6

In the above diagram, it starts from the login page it will check if the user provided the details successful or not. If it is not successful it redirects back to the login page. If it is successful, it displays user successfully logged in. User can view the profile and modify the password. User can view the BMI, BMR to check the details and keep the track of water intake, diet plan and steps. User can get time to time update through notification and lastly user can logout from an application.

DATA FLOW DIAGRAMS (DFDs)

A data flow diagram is a graphical tool that is used to analyze and describe the movement of data through a system. It is the central tool and forms the basis for developing other components. Logical data flow diagrams describe the transformation of data from input to output, through processing, without reference to the physical components of the system. Physical data flow diagrams, on the other hand, show the actual implementation and movement of data between people, departments, and workstations. To fully describe a system, a set of data flow diagrams is necessary. The two most commonly used notations for developing these diagrams are the Yourdon and the Gane and Sarson notations.

Each component in a data flow diagram is labelled with a descriptive name, and the process is identified with a number for identification purposes. The development of data flow diagrams is done on several levels, and each process in lower-level diagrams can be broken down into a more detailed data flow diagram at the next level. The highest-level diagram is called the context diagram, which consists of a single process and is vital in studying the current system. The process in the context level diagram is exploded into other processes at the first-level data flow diagram.

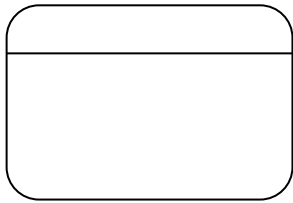
The idea behind the explosion of a process into more processes is that understanding at one level of detail is exploded into greater detail at the next level until an adequate amount of detail is described for analysts to understand the process. The DFD was first developed by Larry Constantine as a way of expressing system requirements in a graphical form, which led to a modular design.

A data flow diagram, also known as a bubble chart, clarifies system requirements and identifies major transformations that will become programs in system design. It is the starting point for design and provides a detailed description of the system from the highest level of detail to the lowest. A data flow diagram consists of a series of bubbles joined by data flows in the system.

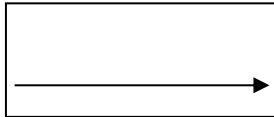
DFD SYMBOLS:

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

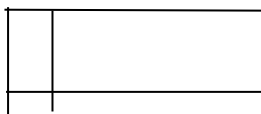


The process that transforms data flow.



Source or destination of data

Data flow



Data Store

CONSTRUCTING A DFD

Several rules of thumb are used in drawing DFDs:

Process should be named and numbered for easy reference. Each name should be representative of the process.

The direction of flow is from top to bottom and from left to right. Data traditionally flow from the source to the destination although they may flow back to the source. One way to indicate this is to draw a long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.

When a process is exploded into lower-level details, they are numbered.

The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each word capitalized

A DFD typically shows the minimum contents of the data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing Interface redundancies and like are then accounted for often through interviews

SALIENT FEATURES OF DFDS

A data flow diagram (DFD) focuses on showing the flow of data through a system and does not depict the control loop or decisions that are made in the system. It also does not indicate the time factor involved in any process or the sequence of events.

There are four types of data flow diagrams:

Current Physical: This type of DFD includes the names of people, positions, or computer systems involved in the overall system processing. It also identifies the technology used

to process data, and data flows and stores are labelled with the names of physical media on which data is stored.

Current Logical: In this type of DFD, the physical aspects of the system are removed as much as possible, and the current system is reduced to its essence, which includes only the data and the processors that transform them.

New Logical: If the user is satisfied with the functionality of the current system but has issues with its implementation, this type of DFD is used. The new logical model may differ from the current logical model by having additional functions, removing absolute functions, or recognizing inefficient flows.

New Physical: This type of DFD represents only the physical implementation of the new system.

RULES GOVERNING THE DFDS

The rules governing DFDs are as follows:

PROCESS:

- A process must have both inputs and outputs, it cannot have only one or the other.
- If a process has only inputs, it is considered a sink.
- A process is labelled with a verb phrase.

DATASTORE:

- Data cannot move directly from one data store to another, it must go through a process.
- Data cannot move directly from an outside source to a data store, it must first be received by a process and then placed into the data store.
- A data store is labelled with a noun phrase.

SOURCE OR SINK:

- A source or sink represents the origin or destination of data.
- Data cannot move directly from a source to a sink, it must go through a process.
- A source or sink is labelled with a noun phrase.

DATA FLOW:

- A data flow can only move in one direction between symbols.
- A data flow can flow in both directions between a process and a data store to show a read before an update, but separate arrows are usually used for clarity.
- A join in a DFD indicates that the same data comes from two or more different processes, data stores, or sinks to a common location.
- A data flow cannot go directly back to the same process it leads from; there must be at least one other process that handles the data flow and produces another data flow that returns the original data to the beginning process.
- A data flow to a data store represents an update (delete or change).
- A data flow from a data store represents retrieval or use.

DATAFLOW DIAGRAMS(DFD'S)

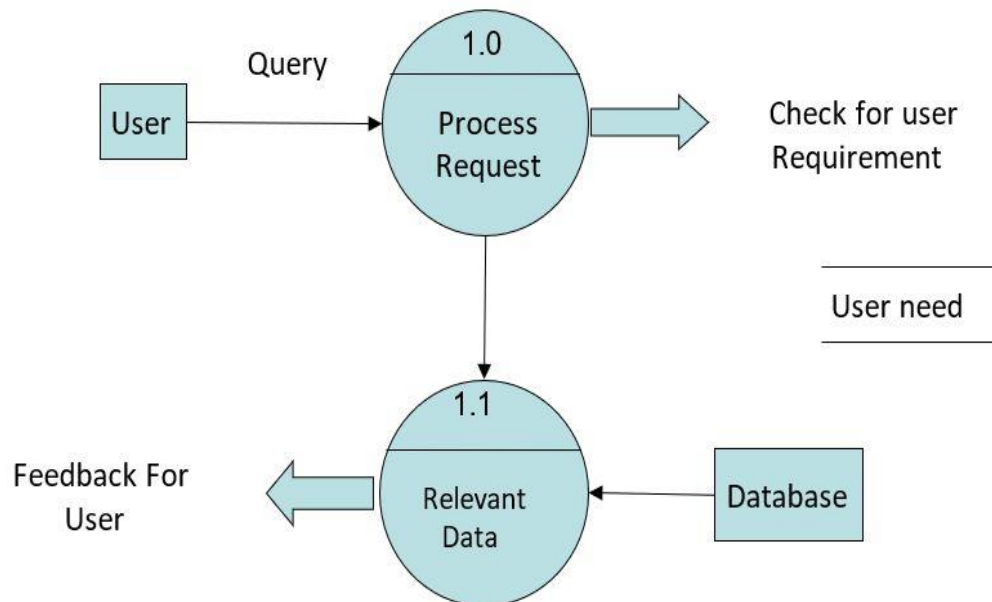
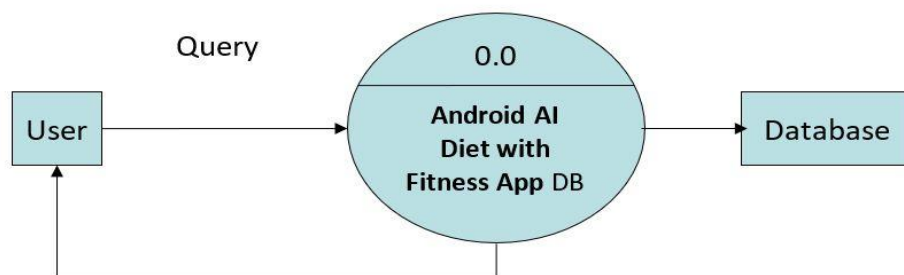


Fig 7

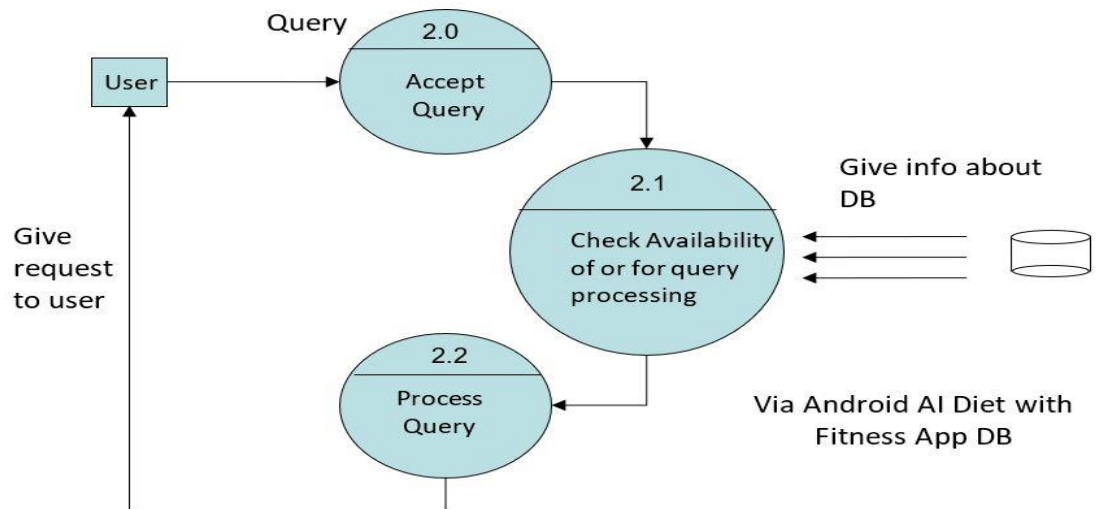
■ LEVEL1 DFD



DATABASE DETAIL

Fig 8

■ LEVEL2 DFD PREDICTION



LEVEL 2 DFD: PREDICTION

Fig 9

SYSTEM ARCHITECTURE

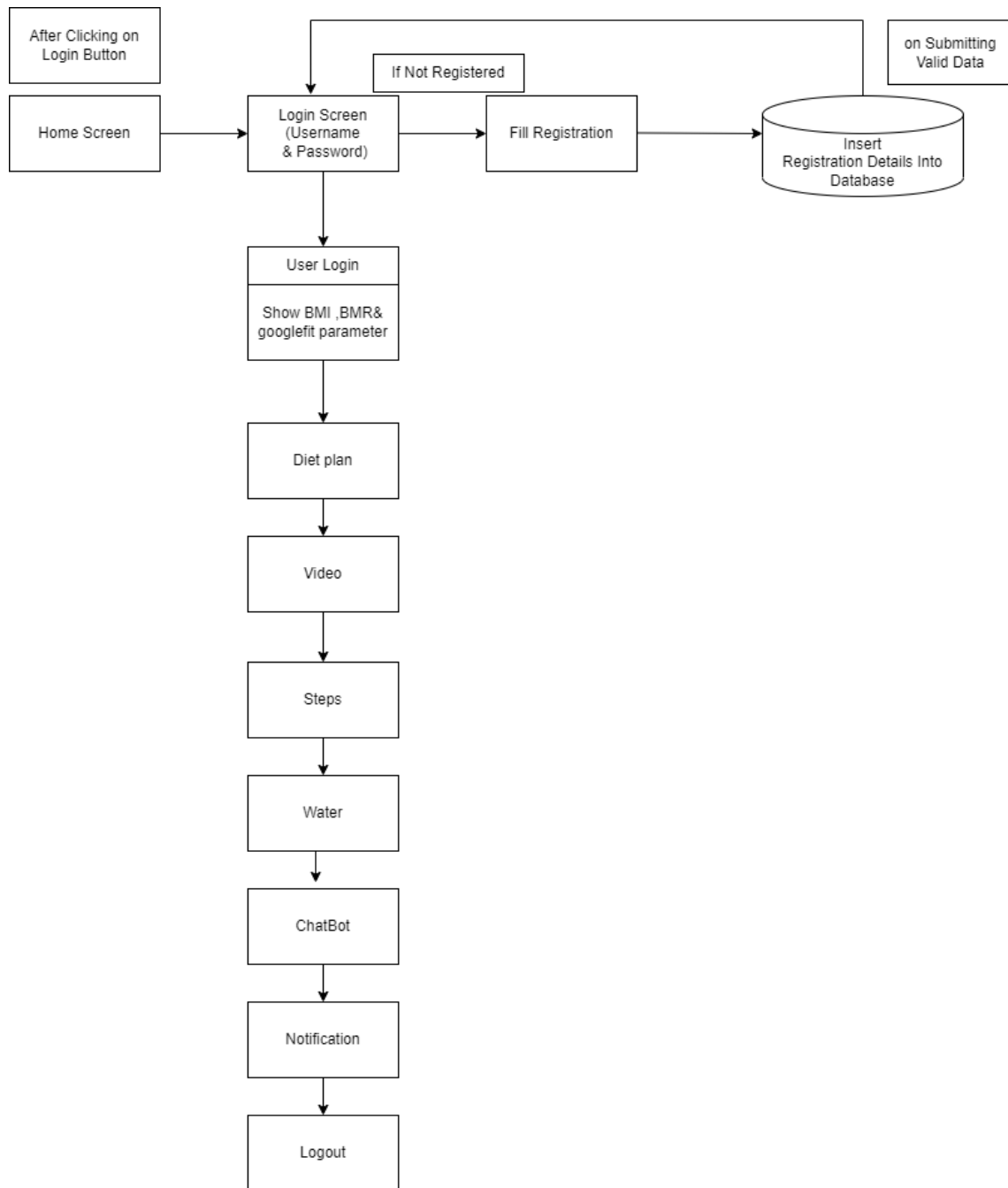
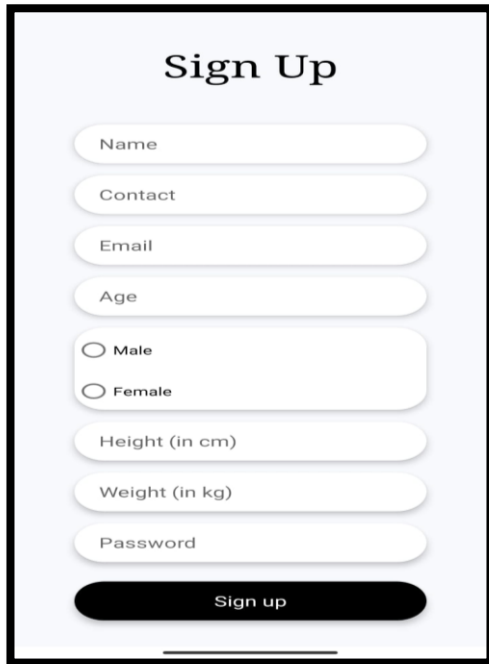


Fig 10

In the above figure, user have to fill the registration then login by providing username and password into an application which it displays the home screen. If not registered it redirects to the registration page. After registration, the details stored in a database on submitting the valid data. User can view the profile and modify the password. User can view the BMI, BMR to check the details and keep the track of water intake, diet plan and steps. User can get time to time update through notification and lastly user can logout from an application.

SNAPSHOTS OF AI BASED DIET AND EXERCISE APPLICATION

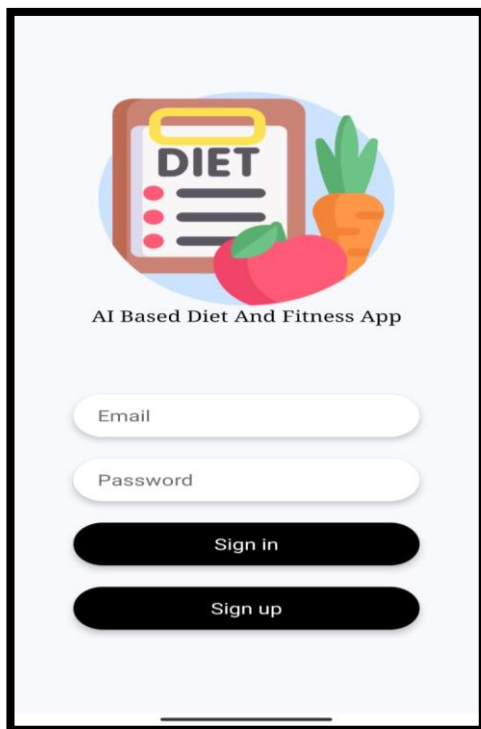
USER SIGN IN/SIGN UP



A mobile app interface for a 'Sign Up' page. The title 'Sign Up' is at the top. Below it are input fields for Name, Contact, Email, Age, Gender (with radio buttons for Male and Female), Height (in cm), Weight (in kg), and Password. A black 'Sign up' button is at the bottom.

Fig 11

In the above figure, User have to fill the registration page by providing the details are Name, contact number, Email, age, gender, height(cm), weight(kg) and password. After registration it redirects to the login page



A mobile app interface for a 'Sign In' page. At the top is an illustration of a clipboard with 'DIET' written on it, next to a carrot and a tomato. Below the illustration is the text 'AI Based Diet And Fitness App'. There are input fields for Email and Password. At the bottom are two black buttons: 'Sign in' and 'Sign up'.

Fig 12

In the above figure, after registration the login page is displayed where the user have to fill the details are Email address and password.

GOOGLE FIT PARAMETERS

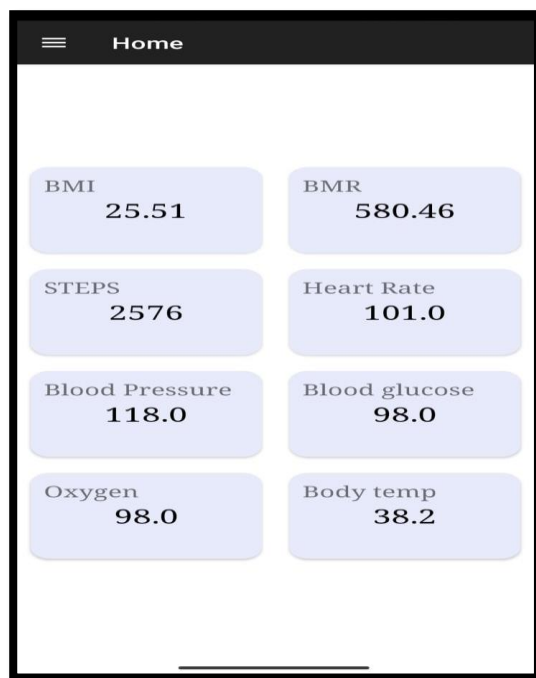


Fig 13

In the above diagram, it displays the details are BMI, BMR, steps, heart rate, blood pressure, blood glucose, Oxygen and body temp which is directly synced through google fit application

SEEKING DIET PLAN AND FITNESS

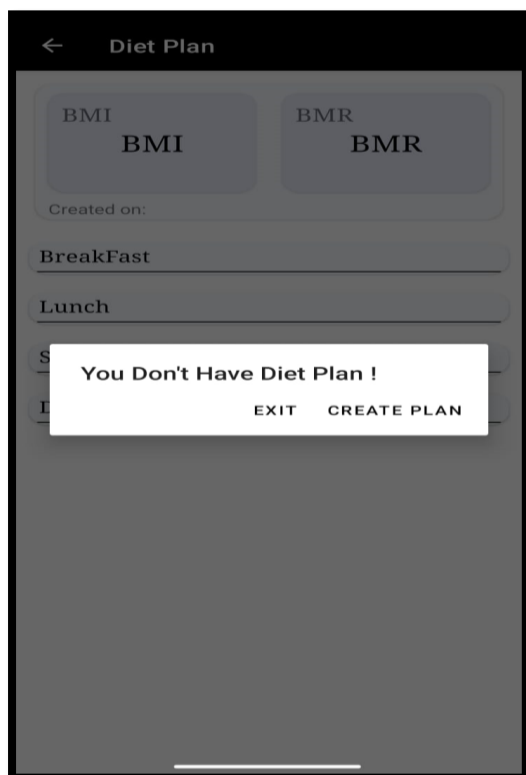


Fig 14

In the above figure, if the user don't have any diet plan he/she can create the diet plan based on that it displays the schedules for breakfast, lunch, snacks and dinner.

DIET PLAN

BreakFast
Food Item: Rava Dosai Calories: 150
Food Item: Pongal Calories: 210
Food Item: Vada Calories: 210
Food Item: Yellow Cheese Calories: 77
Lunch
Food Item: Dal Makhani Calories: 320
Food Item: Hummus(2 table spoons) Calories: 78
Food Item: Grilled Chicken with Vegetables Calories: 213
Snacks
Food Item: Melon(100g) Calories: 70
Dinner
Food Item: Tandoori Chicken Calories: 260
Food Item: Egg Calories: 95

Fig 15

In the above figure, it displays the diet plan for breakfast, lunch, snacks and dinner where it includes the what type of food item and how much calories does the user consumed.

FITNESS VIDEOS

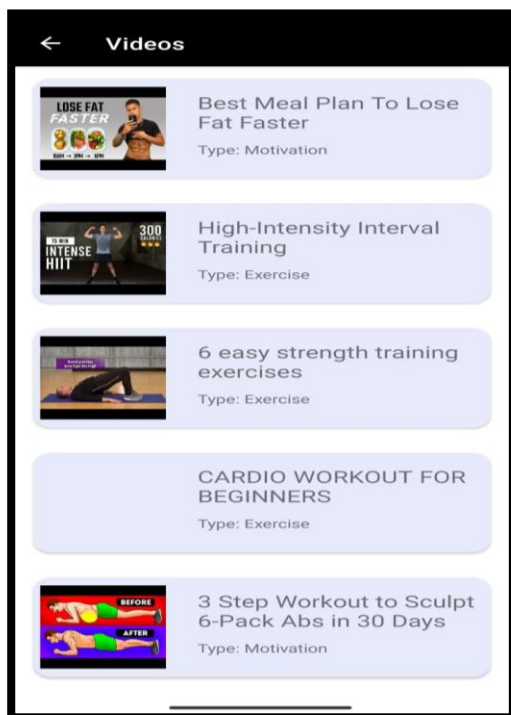


Fig 16

In the above figure, the videos are related to the exercise and diet plan. The system will give a list of exercise, workout and motivation videos based on the user's BMI.

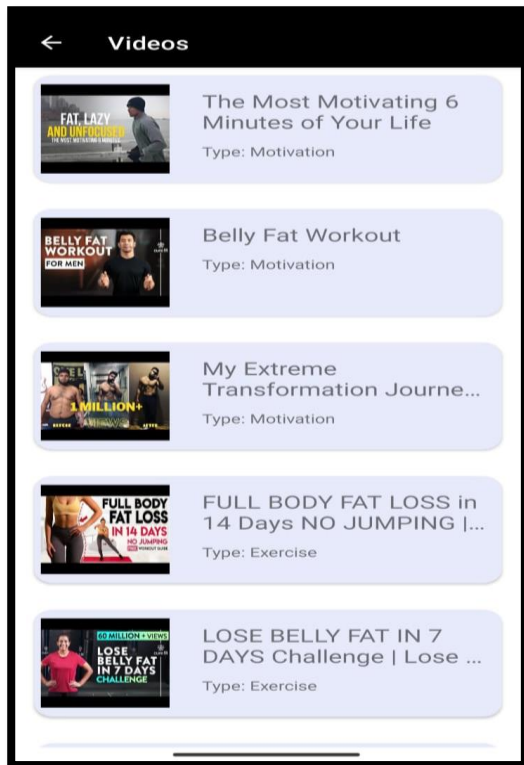


Fig 17

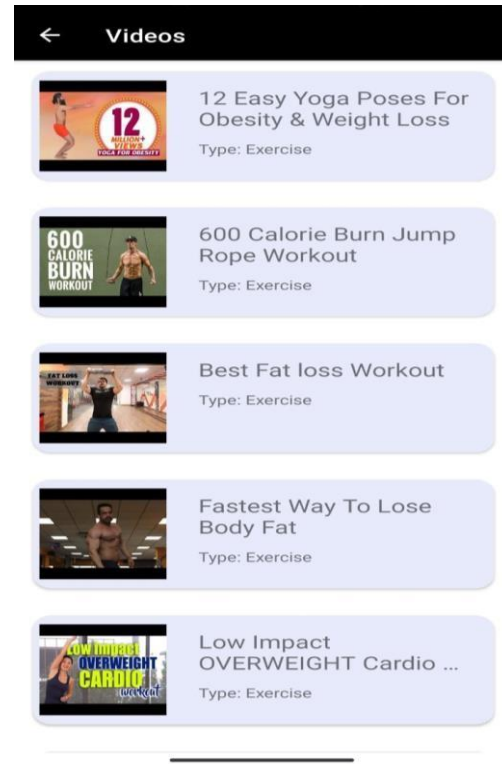


Fig 18

In the above figure, the videos are related to the exercise and diet plan Videos will be automatically suggested based on the users BMI and BMR.

STEPS

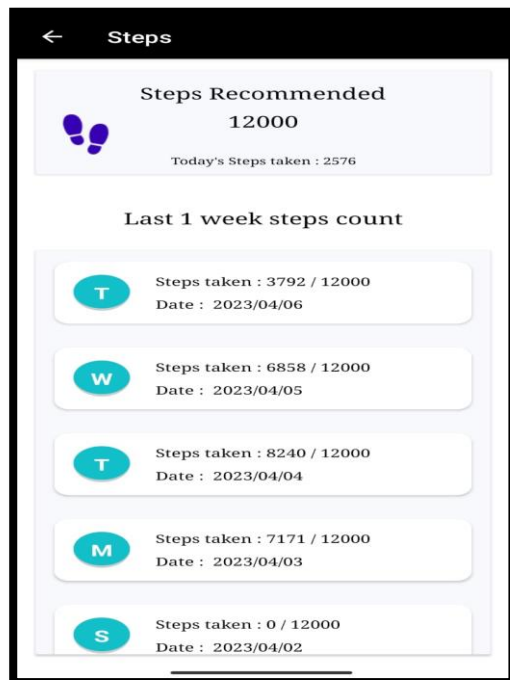


Fig 19

In the above figure, it displays the steps where the user can track and keep the records. The system will show how many steps the user walks daily along with 1 week's history.

DRINKING WATER

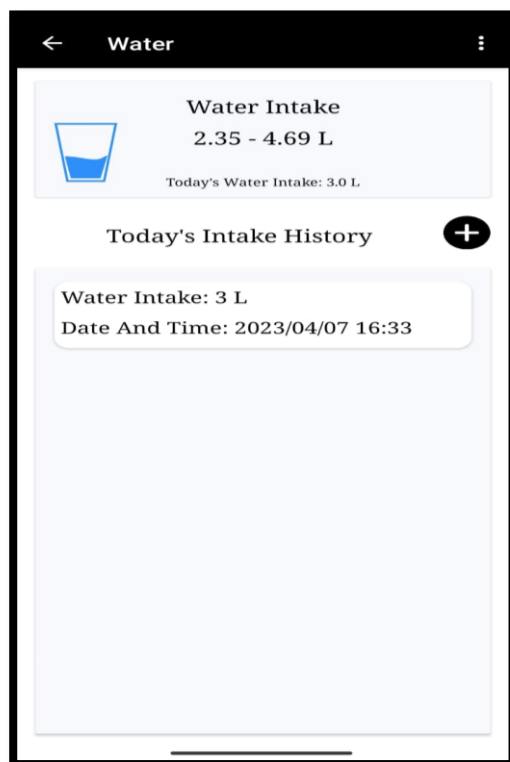


Fig 20

In the above figure, it displays the water intake, date and time where the user can track and keep the records based on the user consumed the water. They can update the intake as required. It will show the week's history of water intake.

NOTIFICATION OF STEPS

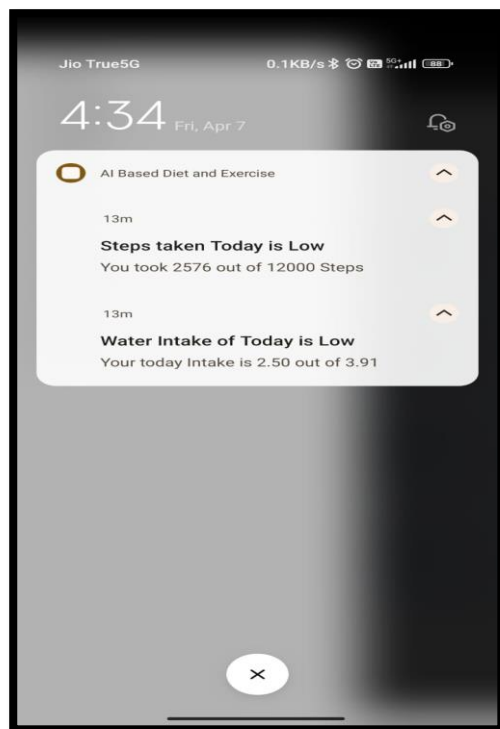


Fig 21

In the above figure, the application will send the notification if the user has not walked the desired steps for the day. There will be a notification if the user has not drunk/updated the water limit that they have to intake per day

CHATBOT

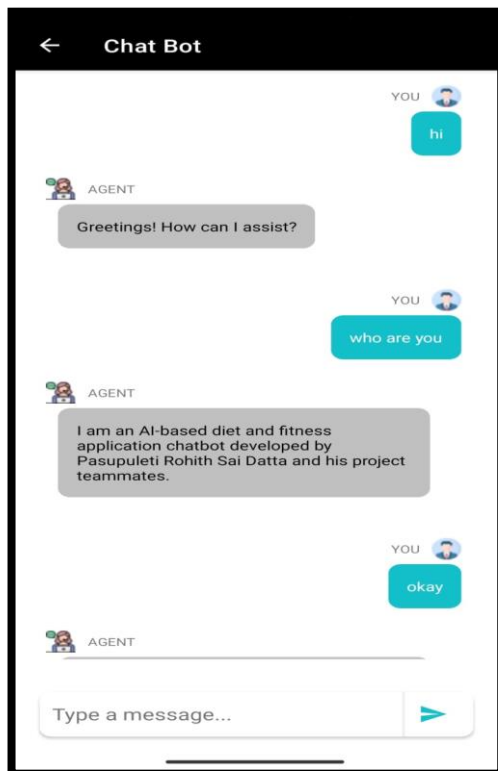


Fig 22

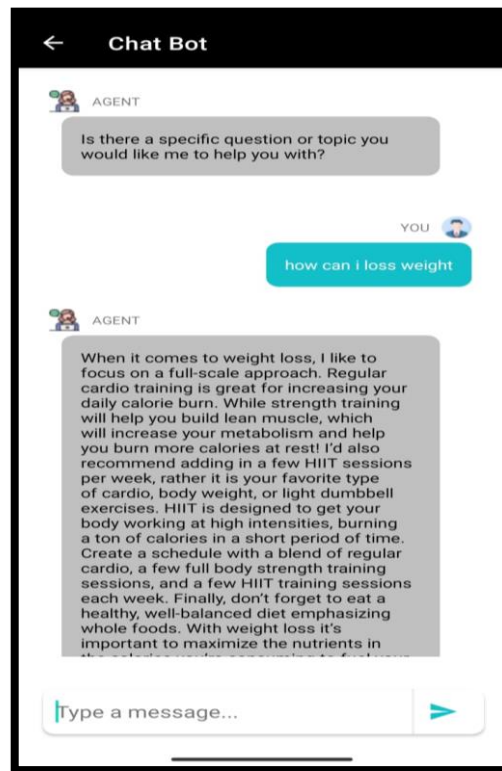


Fig 23

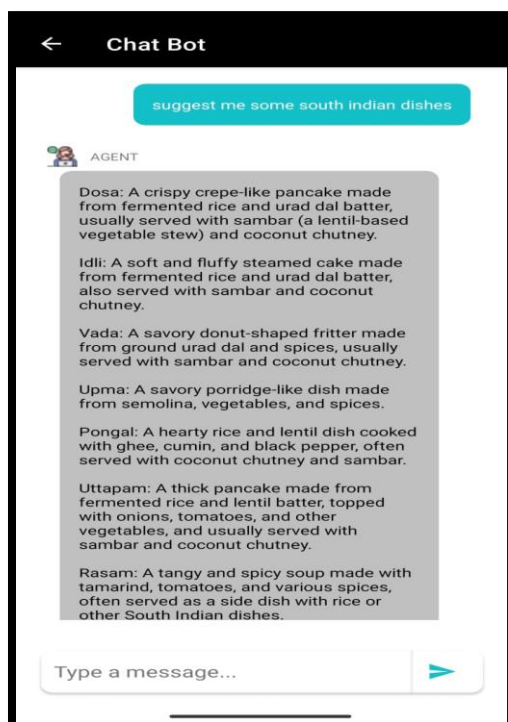


Fig 24

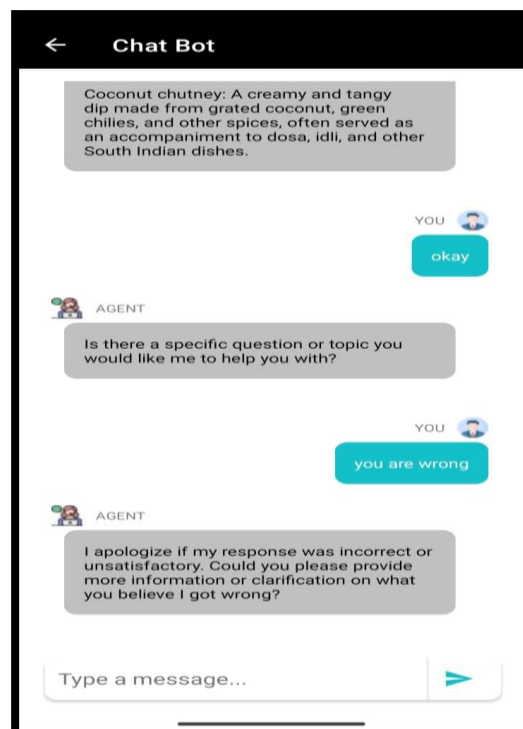


Fig 25

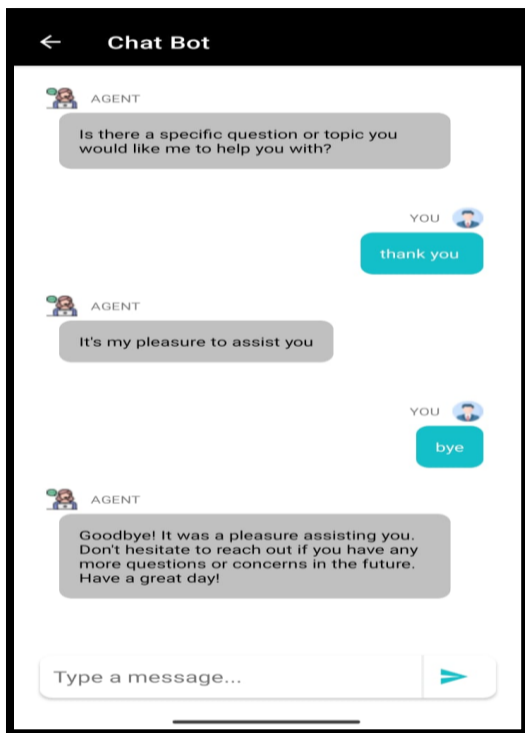


Fig 26

These are the questions asked by the user to the chatbot and the responses of the chatbot to the user

■ USER PROFILE

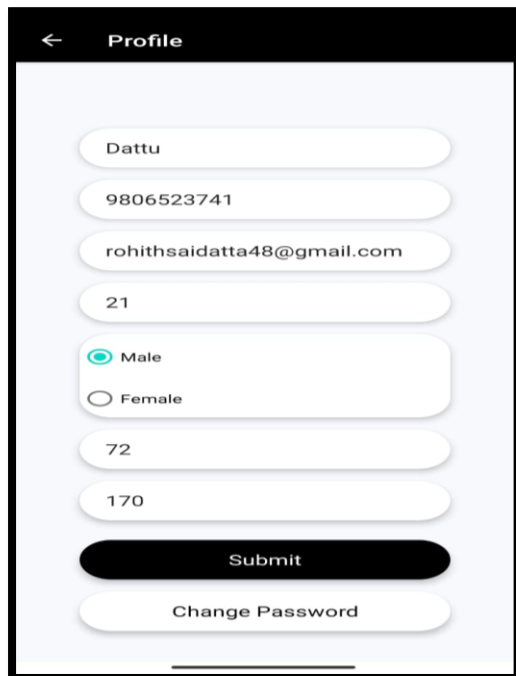
A screenshot of a mobile application's 'Profile' screen. The header shows a back arrow and the title 'Profile'. The form contains several input fields with rounded corners: 'Dattu', '9806523741', 'rohithsaidatta48@gmail.com', '21', a gender selection with 'Male' selected (indicated by a green dot) and 'Female' (indicated by an empty circle), '72', and '170'. Below these fields is a black 'Submit' button. At the very bottom, there is a white button with the text 'Change Password'.

Fig 27

In the above figure, user can see his details and user can change update to the new password if you want

PROJECT IMPLEMENTATION

PROJECT IMPLEMENTATION TECHNOLOGY

The Project application is loaded in Android Studio. We used Android Studio for the Design and coding of the project.

HARDWARE REQUIREMENT

LAPTOP OR PC

- Windows 7 or higher
- I3 processor system or higher
- 8 GB RAM or higher
- 100 GB ROM or higher

ANDROID PHONE

- 6.0 or above

SOFTWARE REQUIREMENT

LAPTOP OR PC

- Android Studio
- Azure Data Studio

OVERVIEW OF TECHNOLOGIES USED

INTRODUCTION TO ANDROID

Android is a mobile operating system that is widely used around the world. Android Studio is the official Integrated Development Environment (IDE) for building Android apps, and it is built on top of the powerful code editor and developer tools of IntelliJ IDEA. Android Studio comes packed with numerous features that make building Android apps easier and more efficient, including a flexible Gradle-based build system, a fast and feature-rich emulator, a unified environment for developing on all Android devices, Instant Run for pushing changes to running apps without building a new APK, code templates and GitHub integration to facilitate building common app features and importing sample code, extensive testing tools and frameworks, Lint tools for catching performance, usability, version compatibility, and other issues, C++ and NDK support, and built-in support for Google Cloud Platform to help you integrate Google Cloud Messaging and App Engine seamlessly.

PROJECT STRUCTURE

Android Studio projects are made up of one or more modules, each containing source code files and resource files. There are several types of modules that can be included in an Android Studio project, including Android app modules, library modules, and Google App Engine modules.

An Android app module is the main module of an Android app project, containing all the code and resources required to build the app. It includes activities, services, broadcast receivers, and other components that make up the app.

A library module is a module that contains code and resources that can be shared across multiple app modules or other library modules. It is typically used to implement common functionality that is used by multiple apps.

A Google App Engine module is a module that contains code and resources for building web applications that run on the Google App Engine platform. It includes servlets, filters, and other

components required for building web applications that can be deployed to Google App Engine.

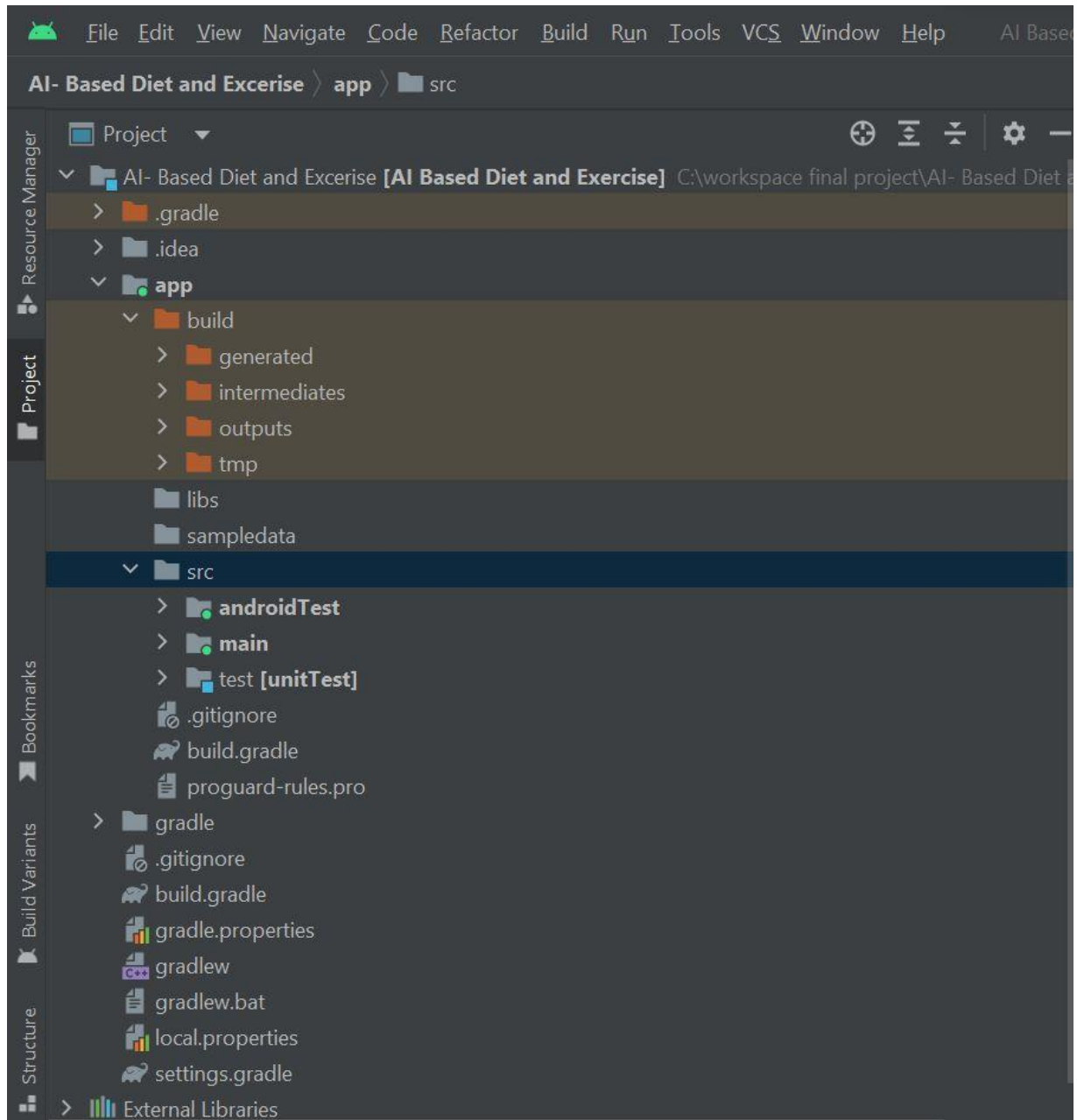


Fig 28

By default, Android Studio displays your project files in the Android project view, as shown in Figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

Manifests: Contains the AndroidManifest.xml file.

Java: Contains the Java source code files, including JUnit test code.

Res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see

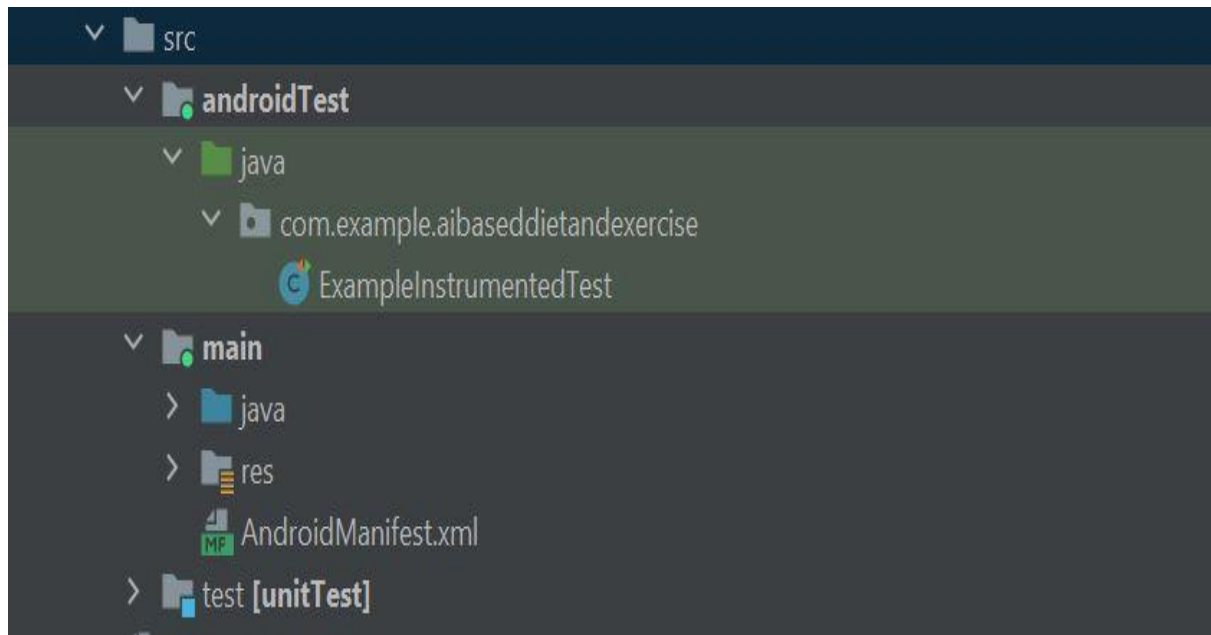


Fig 29

For the actual file structure of the project, select Project from the Project dropdown

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the Problems view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

THE USER INTERFACE

Android Studio has several key interface components that help you carry out different tasks when developing your app. These components include:

The toolbar: This is a horizontal bar that contains buttons and menus for performing a wide range of actions, such as running your app, launching Android tools, and debugging. It is located at the top of the IDE window.

The navigation bar: This is a vertical bar that helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window and is located on the left side of the IDE window.

The editor window: This is the main area where you create and modify code. Depending on the current file type, the editor can change to display specialized tools, such as the Layout Editor when viewing a layout file.

The tool window bar: This is a narrow strip that runs around the outside of the IDE window and contains buttons that allow you to expand or collapse individual tool windows. It gives you access to specific tasks like project management, search, version control, and more.

The status bar: This is a horizontal bar at the bottom of the IDE window that displays the status of your project and the IDE itself, as well as any warnings or messages that need your attention. It also displays helpful information such as the current branch of your project in version control.

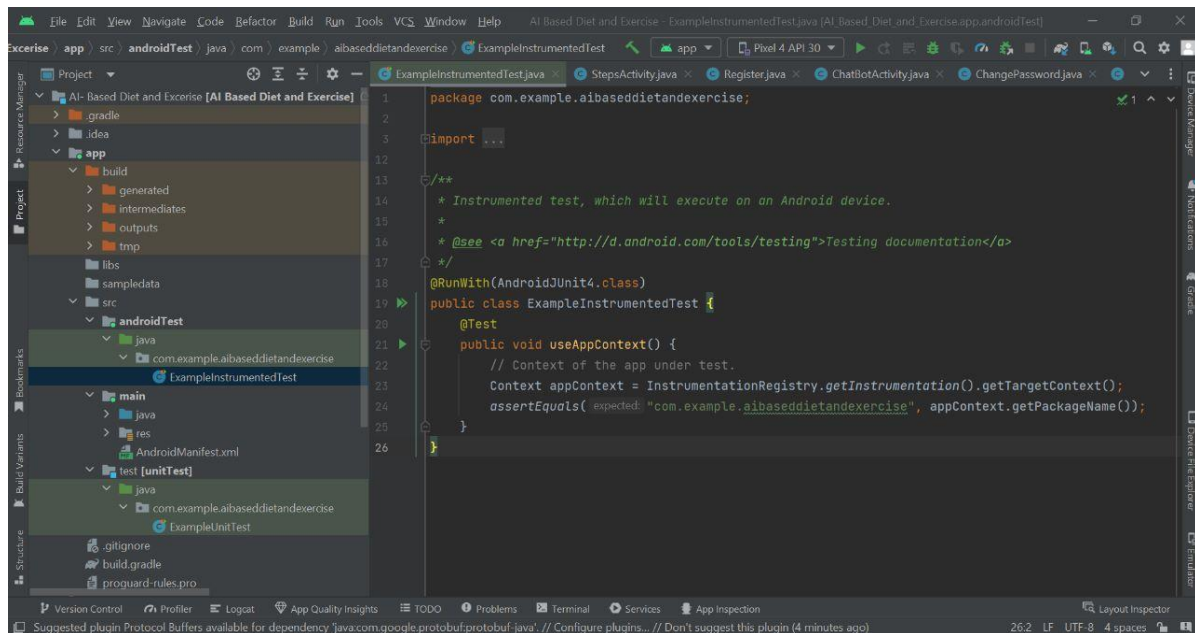


Fig 30

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

TOOL WINDOWS

Android Studio offers a flexible workspace that adapts to your workflow by automatically displaying relevant tool windows as you work. The most commonly used tool windows are pinned to the tool window bar by default. Here are some tips for managing tool windows in Android Studio:

To expand or collapse a tool window, click its name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows as needed.

To return to the default tool window layout, click Window > Restore Default Layout. You can also customize your default layout by clicking Window > Store Current Layout as Default.

To show or hide the entire tool window bar, click the window icon located in the bottom left-hand corner of the Android Studio window.

To locate a specific tool window, hover over the window icon and select the desired tool window from the menu.

By managing tool windows in Android Studio, you can tailor the workspace to your needs and work more efficiently.

NAVIGATION

To help you navigate around Android Studio efficiently, here are some tips:

Use the Recent Files action to switch between recently accessed files. Press Control+E (Command+E on a Mac) to bring up the action, which also provides access to any tool window through the left column.

View the structure of the current file using the File Structure action. Press Control+F12 (Command+F12 on a Mac) to bring up the action, which allows you to quickly navigate to any part of the current file.

Search for and navigate to a specific class in your project using the Navigate to Class action. Press Control+N (Command+O on a Mac) to bring up the action, which supports sophisticated expressions, including camel humps, paths, line navigator, middle name matching, and many more. Calling it twice in a row shows you the results of the project classes.

Navigate to a file or folder using the Navigate to File action. Press Control+Shift+N (Command+Shift+O on a Mac) to bring up the action. To search for folders instead of files, add a / at the end of your expression.

Navigate to a method or field by name using the Navigate to Symbol action. Press Control+Shift+Alt+N (Command+Shift+Alt+O on a Mac) to bring up the action.

Find all the pieces of code referencing the class, method, field, parameter, or statement at the current cursor position by pressing Alt+F7.

By using these navigation shortcuts in Android Studio, you can save time and quickly find what you need to work on.

GRADLE BUILD SYSTEM

Android Studio relies on Gradle as its build system foundation, which is extended with the Android plugin for Gradle to provide more Android-specific features. This build system can be accessed from the Android Studio menu and command line, and it offers various capabilities, such as customizing and extending the build process, creating multiple APKs with different features, and reusing code and resources across source sets.

One of the key advantages of the Gradle build system is that it allows you to achieve these goals without modifying your app's core source files. The build files in Android Studio are named "build.gradle" and use Groovy syntax to configure the build with the elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

MULTIPLE APK SUPPORT

To use multiple APK support, you need to create different source sets in your build.gradle file that define the configuration for each APK. Each source set can specify its own resources, assets, and code. You can then use Gradle to create separate APKs based on the source sets.

When creating multiple APKs, you should consider the following points:

- Each APK should have a unique package name to avoid conflicts.

- Each APK should be signed with its own keystore to ensure security.

- Each APK should have a different version code to enable proper version management.

- You can use the same codebase for all APKs, or create different codebases for each APK if necessary.

Overall, multiple APK support is a powerful feature that enables developers to efficiently create and manage multiple versions of their app for different device configurations. It can

help reduce app size and improve performance, while still providing a consistent user experience across different devices.

DEBUG AND PROFILE TOOLS

Android Studio offers various debugging and profiling tools to help you identify and fix issues in your code, and optimize the performance of your app. Here are some examples:

Inline debugging: You can set breakpoints in your code and use the debugger to step through your code line by line, inspect variables, and evaluate expressions.

Logcat: Android Studio includes a powerful logging tool called Logcat that allows you to view log messages generated by your app and filter them by tag, process, or message content.

Memory Profiler: This tool provides real-time data on how your app uses memory, allowing you to track down memory leaks and optimize memory usage.

Performance Profiler: This tool enables you to analyze your app's CPU, memory, and network usage, and helps you identify performance bottlenecks.

Network Profiler: This tool lets you inspect network activity in your app, including request and response headers, payloads, and timings.

Method Tracing: This tool provides a detailed analysis of your app's method execution time, allowing you to identify performance issues and optimize your code.

Overall, Android Studio provides a comprehensive set of tools to help you debug and optimize your app, making it easier to deliver high-quality, performant apps to your users.

INLINE DEBUGGING

Use inline debugging to enhance your code walk-throughs in the debugger view with inline verification of references, expressions, and variable values. Inline debug information includes:

- Inline variable values
- Referring to objects that reference a selected object
- Method return values
- Lambda and operator expressions
- Tooltip values

PERFORMANCE MONITORS

Android Studio provides performance monitors so you can more easily track your app's memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyze network requests. With your app running on a device or emulator, open the Android Monitor tool window, and then click the Monitors tab.

ALLOCATION TRACKER

Android Studio allows you to track memory allocation as it monitors memory use. Tracking memory allocation allows you to monitor where objects are being allocated when you

perform certain actions. Knowing these allocations enables you to optimize your app's performance and memory use by adjusting the method calls related to those actions.

CODE INSPECTIONS

Whenever you compile your program, Android Studio automatically runs configured Lint and other IDE inspections to help you easily identify and correct problems with the structural quality of your code.

The Lint tool checks your Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.

AZURE DATA STUDIO

Azure Data Studio is a cross-platform database tool that can be used in conjunction with Android Studio for managing and querying databases. It provides a modern, intuitive interface for performing tasks such as creating databases and tables, executing queries, and analyzing query performance. With Azure Data Studio, you can connect to a wide range of database platforms, including SQL Server, PostgreSQL, and MySQL, among others. The tool also supports extensions, enabling you to add additional functionality to meet your specific needs.

DIALOGUE FLOW

Dialogue flow is a natural language processing (NLP) platform that enables developers to design and integrate conversational user interfaces into mobile apps, web applications, devices, and bots. It uses machine learning algorithms to understand and respond to user requests in a conversational manner. With Dialog flow, developers can create chatbots, voice assistants, and other conversational interfaces that can be integrated into Android Studio projects.

CHAPTER 4

Coding

Login

```
package com.example.aibaseddietandexercise;

import android.app.Dialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
```

```

import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;
import com.google.android.material.snackbar.BaseTransientBottomBar;
import com.google.android.material.snackbar.Snackbar;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class Login extends AppCompatActivity {

    EditText email, password;
    CardView submit, register;
    Dialog dl;
    TextView forgotpass;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String uid = Util.getSP(Login.this);
        if (uid.length() > 0) {
            Intent i = new Intent(Login.this, MainActivity.class);
            startActivity(i);
            finish();
        } else {
            setContentView(R.layout.activity_login);
            getSupportActionBar().hide();
            init();
        }
    }

    public void init() {
        email = findViewById(R.id.email);
        password = findViewById(R.id.password);
        submit = findViewById(R.id.submit);
        register = findViewById(R.id.register);

        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (validate()) {
                    new login().execute(email.getText().toString(),
password.getText().toString());
                }
            }
        });

        register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new

```

```

Intent (com.example.aibaseddietandexercise.Login.this, Register.class);
        startActivity(i);
    }
});

}

    public boolean validate() {
        if (email.getText().toString().length() == 0) {
            Snackbar.make(submit, "Enter Email",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (!isEmailValid(email.getText().toString())) {
            Snackbar.make(submit, "Enter Valid Email Address",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (password.getText().toString().length() == 0) {
            Snackbar.make(submit, "Enter Password",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else {
            return true;
        }
        return false;
    }

    boolean isEmailValid(CharSequence email) {
        return
android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches();
    }

    public class login extends AsyncTask<String, String, String> {

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dl =
Util.dailog(com.example.aibaseddietandexercise.Login.this);
            dl.show();
        }

        @Override
        protected String doInBackground(String... strings) {
            String data = null;
            RestAPI restAPI = new RestAPI();

            try {
                JSONParse jp = new JSONParse();
                JSONObject json = restAPI.Login(strings[0], strings[1]);
                data = jp.parse(json);
            } catch (Exception e) {
                data = e.getMessage();
            }
            return data;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            Log.d("res", s);
        }
    }

```

```

        try {
            JSONObject jsonObject = new JSONObject(s);
            String response = jsonObject.getString("status");
            if (response.compareTo("ok") == 0) {
                JSONArray Arr = jsonObject.getJSONArray("Data");
                String res = Arr.getString(0);
                JSONObject jsonObject1 = new JSONObject(res);
                String uid = jsonObject1.getString("data0");

                Log.d("uid", uid);
                Util.setSP(getApplicationContext(), uid);

                //                LoginSharedPref.setGidKey(LoginActivity.this,
                jsonObject.getString("data0"));
                //                LoginSharedPref.setGnameKey(LoginActivity.this,
                jsonObject.getString("data1"));
                dl.cancel();
                startActivity(new Intent(getApplicationContext(),
                MainActivity.class));
                finish();

            } else {

                dl.cancel();

                Snackbar.make(submit, "Login Failed",
                BaseTransientBottomBar.LENGTH_SHORT).show();

            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}

```

The above code is an Android application's login activity. It includes the following functionalities:

- It checks if the user is already logged in and redirects to the main activity if so.
- It allows users to enter their email and password.
- It validates the input fields to ensure they are not empty and the email is in the correct format.
- It includes an asynchronous task to handle the login process.
- It communicates with a REST API to send the login credentials and retrieve the response.
- If the login is successful, it stores the user ID and navigates to the main activity.
- If the login fails, it displays a message indicating the failure.

If you require further assistance or have specific questions about the code or your project report, feel free to ask.

Main Activity

```
package com.example.aibaseddietandexercise;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Dialog;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.Model.FitnessData;
import com.example.aibaseddietandexercise.Service.HomeFitnessService;
import com.example.aibaseddietandexercise.Service.MyUtils;
import com.example.aibaseddietandexercise.Service.WaterAndStepsNotificationService;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.common.SignInButton;
import com.google.android.material.navigation.NavigationView;

import org.json.JSONArray;
import org.json.JSONObject;

public class MainActivity extends AppCompatActivity {
    private DrawerLayout mDrawer;
    private Toolbar toolbar;
    private NavigationView nvDrawer;

    // Make sure to be using androidx.appcompat.app.ActionBarDrawerToggle
    version.
    private ActionBarDrawerToggle drawerToggle;
    LinearLayout signinbuttonlayout;
    TextView bmitv, bmrtv, stepstv, heartratetv, bloodpressure,
    bloodglucose, oxygentv, bodytemp;
    SignInButton signinbtn;

    Dialog dl;
    private Intent serviceNotification;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```

        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        signinbuttonlayout = (LinearLayout)
findViewById(R.id.signinbuttonlayout);
        bmitv = (TextView) findViewById(R.id.bmihome);
        bmrtv = (TextView) findViewById(R.id.bmrhome);
        stepstv = (TextView) findViewById(R.id.stepshome);
        heartratetv = (TextView) findViewById(R.id.heartratehome);
        bloodpressure = (TextView) findViewById(R.id.bloodpressurehome);
        bloodglucose = (TextView) findViewById(R.id.bloodglucosehome);
        oxygentv = (TextView) findViewById(R.id.oxygenhome);
        bodytemp = (TextView) findViewById(R.id.bodytemphome);
        signinbtn = (SignInButton) findViewById(R.id.signin);
        signinbtn.setOnClickListener(v ->
HomeFitnessService.signInToGoogle(this));

//          This will display an Up icon (<-), we will replace it with
hamburger later

getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_baseline_menu_24);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);

// Find our drawer view
mDrawer = (DrawerLayout) findViewById(R.id.drawer_layout);

nvDrawer = (NavigationView) findViewById(R.id.nvView);
// Setup drawer view

setupDrawerContent(nvDrawer);

MenuItem item = nvDrawer.getMenu().getItem(0);
selectDrawerItem(item);

new
getBmiBmrAndRecommendedSteps().execute(Util.getSP(getApplicationContext()))
;

try {
    if (!MyUtils.isMyServiceRunning(MainActivity.this)) {
        //run if not running
        serviceNotification = new Intent(this,
WaterAndStepsNotificationService.class);
        startService(serviceNotification);
    } else {
        //if already running then first stop service and restart
        serviceNotification = new Intent(this,
WaterAndStepsNotificationService.class);
        stopService(serviceNotification);
        Thread.sleep(1000);
        startService(serviceNotification);
    }
} catch (Exception e) {
    e.printStackTrace();
}

}

@Override
protected void onResume() {

```



```

super.onResume();
if (GoogleSignIn.getLastSignedInAccount(this) == null) {
    signinbuttonlayout.setVisibility(View.VISIBLE);
} else {
    signinbuttonlayout.setVisibility(View.GONE);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACTIVITY_RECOGNITION)
            != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
                new
String[]{Manifest.permission.ACTIVITY_RECOGNITION}, 0);
        } else new getFitnessApiData().execute();

    } else new getFitnessApiData().execute();
}
}

public class getBmiBmrAndRecommendedSteps extends AsyncTask<String,
String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        closeDialog();
        dl = Util.dailog(MainActivity.this);
        dl.show();
    }

    @Override
    protected String doInBackground(String... strings) {
        String data = null;
        RestAPI restAPI = new RestAPI();

        try {
            JSONObject json = restAPI.get_BMI_BMR_STEPS(strings[0]);
            Log.d("json", json.toString());
            data = JSONParse.parse(json);
        } catch (Exception e) {
            data = e.getMessage();
        }
        return data;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        Log.d("res", s);
        try {
            JSONObject jsonObject = new JSONObject(s);
            String response = jsonObject.getString("status");
            if (response.compareTo("ok") == 0) {
                JSONArray Arr = jsonObject.getJSONArray("Data");
                String res = Arr.getString(0);
                JSONObject jsonObject1 = new JSONObject(res);
                String bmi1 = jsonObject1.getString("data0");
                String bmr1 = jsonObject1.getString("data1");
                String steps = jsonObject1.getString("data2");

                Log.d("bmi", jsonObject1.getString("data0"));
                Log.d("bmr", jsonObject1.getString("data1"));
                Log.d("steps recommended",

```

```

jsonObject1.getString("data2"));
        Util.setRecommendedSteps(MainActivity.this, steps);
        bmitv.setText(bmil);
        bmrvtv.setText(bmr1);

        dl.cancel();

        } else dl.cancel();
    } catch (Exception e) {
        e.printStackTrace();
    }
    dl.cancel();
}

}

public class getFitnessApiData extends AsyncTask<String, String,
FitnessData> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        closeDialog();
        dl = Util.dailog(MainActivity.this);
        dl.show();
    }

    @Override
    protected FitnessData doInBackground(String... strings) {
        try {
            return
HomeFitnessService.readAllRequiredDataFromFitnessApi(MainActivity.this);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPostExecute(FitnessData response) {
        super.onPostExecute(response);
        Log.d("fitapires", String.valueOf(response));
        try {
            if (response != null) {
                stepstv.setText(response.getSteps() == null ? "--" :
response.getSteps());
                heartratetv.setText(response.getHeartRate() == null ?
"--" : response.getHeartRate());
                bloodpressure.setText(response.getBloodPressure() ==
null ? "--" : response.getBloodPressure());
                bloodglucose.setText(response.getBloodGlucose() == null
? "--" : response.getBloodGlucose());
                oxygentv.setText(response.getOxygen() == null ? "--" :
response.getOxygen());
                bodytemp.setText(response.getBodyTemp() == null ? "--"
: response.getBodyTemp());
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeDialog();
        }
    }
}

```

```

    }
}

private void closeDialog() {
    try {
        dl.cancel();
    } catch (Exception ignored) {
    }
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // The action bar home/up action should open or close the drawer.
    switch (item.getItemId()) {
        case android.R.id.home:
            mDrawer.openDrawer(GravityCompat.START);
            return true;
    }

    return super.onOptionsItemSelected(item);
}

private void setupDrawerContent(NavigationView navigationView) {
    navigationView.setNavigationItemSelectedListener(
        new NavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(MenuItem
menuItem) {

                selectDrawerItem(menuItem);
                return true;
            }
        });
}

@SuppressLint("NonConstantResourceId")
public void selectDrawerItem(MenuItem menuItem) {
    // Create a new fragment and specify the fragment to show based on
    nav item clicked
    switch (menuItem.getItemId()) {
        case R.id.nav_dietplan:
            startActivity(new Intent(getApplicationContext(),
DietPlanActivity.class));
            break;
        case R.id.nav_profile:
            startActivity(new Intent(getApplicationContext(),
profile.class));
            break;

        case R.id.nav_videos:
            startActivity(new Intent(getApplicationContext(),
VideoActivity.class));
            break;
        case R.id.nav_steps:
            startActivity(new Intent(getApplicationContext(),
StepsActivity.class));
            break;
        case R.id.nav_water:
            startActivity(new Intent(getApplicationContext(),
WaterActivity.class));

```

```

        break;
    case R.id.nav_chatbot:
        startActivity(new Intent(getApplicationContext(),
ChatBotActivity.class));
        break;

    case R.id.nav_logout:

        serviceNotification = new Intent(this,
WaterAndStepsNotificationService.class);
        stopService(serviceNotification);
        Util.setSP(getApplicationContext(), "");

        Intent intent = new Intent(getApplicationContext(),
Login.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
        finish();

        break;
    default:
    }

    setTitle("Home");
    // Highlight the selected item has been done by NavigationView
    // menuItem.setChecked(true);
    // Set action bar title
    // setTitle(menuItem.getTitle());
    // Close the navigation drawer
    mDrawer.closeDrawers();
}
}

```

The above code is the main activity of an Android application. It includes the following features:

- Toolbar and navigation drawer setup.
- Google sign-in button functionality.
- Water and steps notification service.
- Retrieval and display of BMI, BMR, and recommended steps data.
- Retrieval and display of fitness data from the Fitness API.
- Navigation to different activities based on menu item selection.
- Logout functionality, including stopping the notification service and clearing the user session.

Profile

```
package com.example.aibaseddietandexercise;

import android.app.Dialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;
import com.google.android.material.snackbar.BaseTransientBottomBar;
import com.google.android.material.snackbar.Snackbar;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class profile extends AppCompatActivity {

    EditText name, contact, email, age, height, weight;
    CardView submit, passcard, Cpass;
    TextView title, submittext;
    Dialog dl;
    String uid;
    RadioButton genderradioButton;
    RadioGroup radioGroup;
    RadioButton genderradioButtonmale, genderradioButtonfemale;
    String gt;
    String radiotext;
    String gender1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        getSupportActionBar().setTitle("Profile");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setHomeButtonEnabled(true);
        init();
    }

    public void init() {
        uid = Util.getSP(profile.this);
        title = findViewById(R.id.title);
        name = findViewById(R.id.name);
        contact = findViewById(R.id.contact);
        email = findViewById(R.id.email);
        age = findViewById(R.id.age);
        height = findViewById(R.id.height);
        weight = findViewById(R.id.weight);
        radioGroup = (RadioGroup) findViewById(R.id.radioGroup);
    }
}
```

```

email.setEnabled(true);
passcard = findViewById(R.id.passcard);
passcard.setVisibility(View.GONE);
title.setVisibility(View.GONE);
submit = findViewById(R.id.submit);
submittext = findViewById(R.id.submittext);
submittext.setText("Submit");
Cpass = findViewById(R.id.Cpass);
Cpass.setVisibility(View.VISIBLE);
getProfile();

radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup arg0, int id) {
        RadioButton checkedRadioButton = (RadioButton)
radioGroup.findViewById(id);

        boolean isChecked = checkedRadioButton.isChecked();

        if (isChecked) {
            gt = checkedRadioButton.getText().toString();
            Log.d("get", gt);
        }
    }
});

submit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (validate()) {
            if (gt != null) {
                new
updateprofile().execute(Util.getSP(getApplicationContext()),
name.getText().toString(), email.getText().toString(),
contact.getText().toString(), age.getText().toString(), gt.toString(), height.
getText().toString(), weight.getText().toString());
            } else {
                new
updateprofile().execute(Util.getSP(getApplicationContext()),
name.getText().toString(), email.getText().toString(),
contact.getText().toString(), age.getText().toString(), radiotext, height.getT
ext().toString(), weight.getText().toString());
            }
        }
    }
});

Cpass.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Intent i = new Intent(profile.this, ChangePassword.class);
        i.putExtra("uid", uid);
        startActivity(i);
        finish();
    }
});

```

```

    }

    public class updateprofile extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dl = Util.dailog(profile.this);
            dl.show();
        }

        @Override
        protected String doInBackground(String... strings) {
            String data = null;
            RestAPI restAPI = new RestAPI();

            try {
                JSONParse jp = new JSONParse();
                JSONObject json =
restAPI.updateprofile(strings[0],strings[1],strings[2],strings[3],strings[4]
],strings[5],strings[6],strings[7]);
                data = jp.parse(json);
            } catch (Exception e) {
                data = e.getMessage();
            }
            return data;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            Log.d("res", s);
            try {
                JSONObject jsonObject = new JSONObject(s);
                String res = jsonObject.getString("status");
                if (res.compareTo("true") == 0) {
                    dl.cancel();
                    Snackbar.make(submit, "Updated",
BaseTransientBottomBar.LENGTH_SHORT).show();
                    finish();
                } else if (res.compareTo("already") == 0) {
                    dl.cancel();
                    Snackbar.make(submit, "Already Exist",
BaseTransientBottomBar.LENGTH_SHORT).show();

                } else {
                    dl.cancel();
                    Snackbar.make(submit, "Failed! Try Again",
BaseTransientBottomBar.LENGTH_SHORT).show();
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }

    private void getProfile() {
        new Viewprofile().execute(Util.getSP(getApplicationContext()));
    }

```

```

    public boolean validate() {
        if (name.getText().toString().length() == 0) {
            Snackbar.make(submit, "Enter Name",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (contact.getText().toString().length() == 0) {
            Snackbar.make(submit, "Enter Contact",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (contact.getText().toString().length() != 10) {
            Snackbar.make(submit, "Enter Valid Contact Number",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (email.getText().toString().isEmpty()) {
            Snackbar.make(submit, "Enter Email",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (age.getText().toString().isEmpty()) {
            Snackbar.make(submit, "Enter Age",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (weight.getText().toString().isEmpty()) {
            Snackbar.make(submit, "Enter Weight",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (height.getText().toString().isEmpty()) {
            Snackbar.make(submit, "Enter Height",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else {
            return true;
        }
        return false;
    }

    public class Viewprofile extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dl = Util.dailog(profile.this);
            dl.show();
        }

        @Override
        protected String doInBackground(String... strings) {
            String data = null;
            RestAPI restAPI = new RestAPI();

            try {
                JSONParse jp = new JSONParse();
                JSONObject json = restAPI.getProfile(strings[0]);
                data = jp.parse(json);
            } catch (Exception e) {
                data = e.getMessage();
            }
            return data;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            Log.d("res", s);
            try {
                JSONObject jsonObject = new JSONObject(s);
                String response = jsonObject.getString("status");
                if (response.compareTo("ok") == 0) {
                    JSONArray Arr = jsonObject.getJSONArray("Data");

```



```

        String res = Arr.getString(0);
        JSONObject jsonObject1 = new JSONObject(res);
        String uid = jsonObject1.getString("data0");
        String name1 = jsonObject1.getString("data1");
        String email1 = jsonObject1.getString("data2");
        String contact1 = jsonObject1.getString("data3");
        String age1 = jsonObject1.getString("data4");
        gender1 = jsonObject1.getString("data5");
        String weight1 = jsonObject1.getString("data6");
        String height1 = jsonObject1.getString("data7");

        name.setText(name1);
        email.setText(email1);
        contact.setText(contact1);
        age.setText(age1);
        weight.setText(weight1);
        height.setText(height1);

        genderradioButtonfemale = (RadioButton)
findViewById(R.id.radioFemale);
        genderradioButtonmale = (RadioButton)
findViewById(R.id.radioMale);
        Log.d("gender", gender1);
        if (gender1.trim().compareTo("Male") == 0) {
            radiotext = "Male";
            Log.d("check", radiotext);
            genderradioButtonmale.setChecked(true);
        } else if (gender1.trim().compareTo("Female") == 0) {
            radiotext = "Female";
            genderradioButtonfemale.setChecked(true);

        } else {

        }

        dl.cancel();

    } else {

        dl.cancel();

        Snackbar.make(submit, "Loading Profile Failed! Try
Again", BaseTransientBottomBar.LENGTH_SHORT).show();

    }
} catch (JSONException e) {
    e.printStackTrace();
}

}

}

```

The above code is an Android activity class called "Profile." It manages user profile updates and retrieval. It includes methods for initializing views, validating input, updating the profile, and fetching profile data from a server using AsyncTask.

Register

```
package com.example.aibaseddietandexercise;

import android.app.Dialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;
import com.google.android.material.snackbar.BaseTransientBottomBar;
import com.google.android.material.snackbar.Snackbar;

import org.json.JSONException;
import org.json.JSONObject;

public class Register extends AppCompatActivity {
    EditText name, contact, email, password, age, height, weight;
    CardView submit;
    Dialog dl;

    RadioButton genderradioButton;
    RadioGroup radioGroup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        getSupportActionBar().hide();
        init();
    }

    public void init() {
        radioGroup = (RadioGroup) findViewById(R.id.radioGroup);
        name = findViewById(R.id.name);
        contact = findViewById(R.id.contact);
        email = findViewById(R.id.email);
        password = findViewById(R.id.password);
        submit = findViewById(R.id.submit);
        age = (EditText) findViewById(R.id.age);
        weight = (EditText) findViewById(R.id.weight);
        height = (EditText) findViewById(R.id.height);

        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int selectedId = radioGroup.getCheckedRadioButtonId();
```

```

        genderradioButton = (RadioButton) findViewById(selectedId);
        if (selectedId == -1) {
            Snackbar.make(submit, "Select Gender",
BaseTransientBottomBar.LENGTH_SHORT).show();

        } else {
            if (validate()) {
                new register().execute(name.getText().toString(),
email.getText().toString(), contact.getText().toString(),
                age.getText().toString(),
genderradioButton.getText().toString(), height.getText().toString(),
weight.getText().toString(),
                password.getText().toString());

            }
        }
    }

    public boolean validate() {
        if (name.getText().toString().length() == 0) {
            Snackbar.make(submit, "Enter Name",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (contact.getText().toString().length() == 0) {
            Snackbar.make(submit, "Enter Contact",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (contact.getText().toString().length() != 10) {
            Snackbar.make(submit, "Enter Valid Contact Number",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (email.getText().toString().length() == 0) {
            Snackbar.make(submit, "Enter Email",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (!isEmailValid(email.getText().toString())) {
            Snackbar.make(submit, "Enter Valid Email Address",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (password.getText().toString().length() == 0) {
            Snackbar.make(submit, "Enter Password",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else {
            return true;
        }
        return false;
    }

    boolean isEmailValid(CharSequence email) {
        return
android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches();
    }

    public class register extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dl = Util.dailog(Register.this);
            dl.show();
        }

        @Override

```

```

protected String doInBackground(String... strings) {
    String data = null;
    RestAPI restAPI = new RestAPI();

    try {
        JSONParse jp = new JSONParse();
        JSONObject json = restAPI.Register(strings[0],
strings[1].trim(), strings[2].trim(), strings[3].trim(), strings[4].trim(),
strings[5].trim(), strings[6].trim(), strings[7].trim());
        data = jp.parse(json);
    } catch (Exception e) {
        data = e.getMessage();
    }
    return data;
}

@Override
protected void onPostExecute(String s) {
    super.onPostExecute(s);
    Log.d("res", s);

    try {
        JSONObject jsonObject = new JSONObject(s);
        String res = jsonObject.getString("status");
        if (res.compareTo("true") == 0) {
            dl.cancel();
            Snackbar.make(submit, "Registered",
BaseTransientBottomBar.LENGTH_SHORT).show();
            finish();
        } else if (res.compareTo("already") == 0) {
            dl.cancel();
            Snackbar.make(submit, "User Already Exist",
BaseTransientBottomBar.LENGTH_SHORT).show();
        } else {
            dl.cancel();
            Snackbar.make(submit, "Failed! Try Again",
BaseTransientBottomBar.LENGTH_SHORT).show();
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
}
}

```

The above code is an Android activity class called "Register." It handles user registration by capturing user information such as name, contact, email, password, age, height, weight, and gender. It includes methods for initializing views, validating input, and registering the user by making a request to a server using AsyncTask.

Steps Activity

```
package com.example.aibaseddietandexercise;

import android.annotation.SuppressLint;
import android.app.Dialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.aibaseddietandexercise.Adapter.StepsAdapter;
import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.Model.Steps;
import com.example.aibaseddietandexercise.Service.StepsService;

import java.util.Collections;
import java.util.List;

public class StepsActivity extends AppCompatActivity {

    Dialog dl;
    TextView stepstxt, totaltaken;
    StepsAdapter adapter;
    RecyclerView recyclerView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_steps);

        getSupportActionBar().setTitle("Steps");
        stepstxt = (TextView) findViewById(R.id.title1);
        totaltaken = (TextView) findViewById(R.id.title2);

        String recommendedSteps =
            Util.getRecommendedSteps(getApplicationContext());
        stepstxt.setText(recommendedSteps == null ||
            recommendedSteps.trim().isEmpty() ? "0"
                : recommendedSteps.trim());

        recyclerView = (RecyclerView)
            findViewById(R.id.recyclerview_steps);
        LinearLayoutManager layoutManager = new
            LinearLayoutManager(getApplicationContext());
        recyclerView.setLayoutManager(layoutManager);
    }

    @Override
    protected void onResume() {
        super.onResume();
        new ViewStepsTaken().execute();
    }

    public class ViewStepsTaken extends AsyncTask<String, String,
        List<Steps>> {
        @Override
```

```

protected void onPreExecute() {
    super.onPreExecute();
    closeDialog();
    dl = Util.dailog(StepsActivity.this);
    dl.show();
}

@Override
protected List<Steps> doInBackground(String... strings) {
    return StepsService.readWeeklyStepData(StepsActivity.this);
}

@SuppressLint("NotifyDataSetChanged")
@Override
protected void onPostExecute(List<Steps> stepsList) {
    super.onPostExecute(stepsList);
    closeDialog();
    try {
        if (stepsList != null && stepsList.size() > 0) {
            Collections.reverse(stepsList);
            Steps steps = stepsList.get(0);
            stepsList.remove(steps);
            String totalSteps = steps.getTotalSteps() == null ? "0"
: steps.getTotalSteps();
            totaltaken.setText("Today's Steps taken : " +
totalSteps);
            adapter = new StepsAdapter(stepsList,
StepsActivity.this);
            recyclerView.setAdapter(adapter);
            adapter.notifyDataSetChanged();
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeDialog();
    }
}

private void closeDialog() {
    try {
        if (dl != null) dl.cancel();
    } catch (Exception ignored) {
    }
}
}

```

The above code is an Android activity class called "StepsActivity." It displays the user's daily steps taken and a list of their weekly step data. The activity retrieves the recommended steps from the application's settings and displays it. It also fetches the user's steps data using an AsyncTask and populates it in a RecyclerView using a StepsAdapter. The user's total steps for the day are displayed at the top.

Video Activity

```
package com.example.aibaseddietandexercise;

import android.app.Dialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.aibaseddietandexercise.Adapter.VideoAdapter;
import com.example.aibaseddietandexercise.Adapter.WaterAdapter;
import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.Model.Videos;
import com.example.aibaseddietandexercise.Model.Water;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class VideoActivity extends AppCompatActivity {
    RecyclerView recyclerView;
    Dialog dl;
    VideoAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_video);
        getSupportActionBar().setTitle("Videos");

        recyclerView = (RecyclerView)
        findViewById(R.id.recyclerview_videos);
        LinearLayoutManager layoutManager = new
        LinearLayoutManager(getApplicationContext());
        recyclerView.setLayoutManager(layoutManager);

        new Videos().execute(Util.getSP(getApplicationContext()));
    }

    public class Videos extends AsyncTask<String, String, String> {
        List<com.example.aibaseddietandexercise.Model.Videos> videosList =
        new ArrayList<>();

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dl = Util.dailog(VideoActivity.this);
            dl.show();
        }
    }
}
```

```

    }

    @Override
    protected String doInBackground(String... strings) {
        String data = null;
        RestAPI restAPI = new RestAPI();

        try {
            JSONParse jp = new JSONParse();
            JSONObject json = restAPI.getVideos(strings[0]);
            data = jp.parse(json);
        } catch (Exception e) {
            data = e.getMessage();
        }
        return data;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        Log.d("res", s);
        try {
            JSONObject jsonObject = new JSONObject(s);
            JSONArray array = jsonObject.getJSONArray("Data");
            Log.d("Json", array.toString());
            float sum = 0;
            for (int i = 0; i < array.length(); i++) {
                JSONObject jsonObj = array.getJSONObject(i);
                com.example.aibaseddietandexercise.Model.Videos water =
new
com.example.aibaseddietandexercise.Model.Videos(jsonObj.getString("data0"),
                jsonObj.getString("data1"),
                jsonObj.getString("data2"),
                jsonObj.getString("data3"),
                jsonObj.getString("data4"));

                videosList.add(water);
            }
            adapter = new VideoAdapter(videosList,
getApplicationContext());
            recyclerview.setAdapter(adapter);
            adapter.notifyDataSetChanged();
            dl.cancel();

        } catch (JSONException e) {
            e.printStackTrace();
        }

    }
}
}

```

The code is an Android activity class called "VideoActivity." It retrieves video data from a web service using an AsyncTask and populates it in a RecyclerView using a VideoAdapter. The video data is obtained as a JSON response and parsed to extract the necessary information. The activity displays a list of videos, and each video item contains details such as data0, data1, data2, data3, and data4. The RecyclerView is set up with a

LinearLayoutManager, and the VideoAdapter is responsible for binding the video data to the RecyclerView items.

Water Activity

```
package com.example.aibaseddietandexercise;

import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.aibaseddietandexercise.Adapter.WaterAdapter;
import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.Model.Water;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;
import com.google.android.material.snackbar.BaseTransientBottomBar;
import com.google.android.material.snackbar.Snackbar;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

public class WaterActivity extends AppCompatActivity {
    ImageView fab;
    Dialog dl;
    TextView watertxt, totalintake;
    String formattedDate;
    WaterAdapter adapter;
    RecyclerView recyclerView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_water);
        fab = (ImageView) findViewById(R.id.fabaddwater);

        getSupportActionBar().setTitle("Water");
    }
}
```

```

watertxt = (TextView) findViewById(R.id.title1);
totalintake = (TextView) findViewById(R.id.title2);

Date date = Calendar.getInstance().getTime();
SimpleDateFormat df = new SimpleDateFormat("yyyy/MM/dd");
formattedDate = df.format(date);
Log.d("date", formattedDate);

recyclerView = (RecyclerView)
findViewById(R.id.recyclerview_water);

LinearLayoutManager layoutManager = new
LinearLayoutManager(getApplicationContext());
recyclerView.setLayoutManager(layoutManager);

fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder alertDialog = new
AlertDialog.Builder(WaterActivity.this);
        final View customLayout =
getLayoutInflater().inflate(R.layout.alert_water, null);
        alertDialog.setView(customLayout);
        AlertDialog alert = alertDialog.create();
        alert.setCancelable(true);
        alert.setCanceledOnTouchOutside(true);
        EditText quantity =
customLayout.findViewById(R.id.waterquantity);
        EditText date_ = customLayout.findViewById(R.id.waterdate);
        ImageView img = customLayout.findViewById(R.id.datepicker);
        CardView submitBtn =
customLayout.findViewById(R.id.addwatercard);

        img.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Calendar mcurrentDate = Calendar.getInstance();
                int mYear = mcurrentDate.get(Calendar.YEAR);
                int mMonth = mcurrentDate.get(Calendar.MONTH);
                int mDay = mcurrentDate.get(Calendar.DAY_OF_MONTH);

                DatePickerDialog mDatePicker;
                mDatePicker = new
DatePickerDialog(WaterActivity.this, new
DatePickerDialog.OnDateSetListener() {
                    public void onDateSet(DatePicker datepicker,
int mYear, int mMonth, int selectedday) {
                        // TODO Auto-generated method stub
                        /*      Your code      to get date and time
*/

                        Date d = new Date(mYear, mMonth,
selectedday);

                        SimpleDateFormat dateFormatter = new
SimpleDateFormat(
                            "20YY/MM/dd");
                        String strDate = dateFormatter.format(d);
                        Log.d("format", strDate);
                        date.setText(strDate);

```

```

        }
        }, mYear, mMonth, mDay);
        datePicker.setTitle("Select Date");
        datePicker.show();

    }

});

submitBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (quantity.getText().toString().isEmpty()) {
            Snackbar.make(fab, "Enter Quantity",
                BaseTransientBottomBar.LENGTH_SHORT).show();
        } else if (date_.getText().toString().isEmpty()) {
            Snackbar.make(fab, "Select Date",
                BaseTransientBottomBar.LENGTH_SHORT).show();
        } else {
            Date date = Calendar.getInstance().getTime();
            SimpleDateFormat df = new
                SimpleDateFormat("yyyy/MM/dd HH:mm");
            String formattedDate = df.format(date);
            Log.d("date", formattedDate);
            new
                addwater().execute(Util.getSP(getApplicationContext()),
                    quantity.getText().toString(), date_.getText().toString(), formattedDate);
            alert.dismiss();

        }

    }

});

alert.show();

}

});

}

public class addwater extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        dl = Util.dailog(WaterActivity.this);
        dl.show();
    }

    @Override
    protected String doInBackground(String... strings) {
        String data = null;
        RestAPI restAPI = new RestAPI();

        try {
            JSONParse jp = new JSONParse();
            JSONObject json = restAPI.Addwaterintake(strings[0],
                strings[1], strings[2], strings[3]);
            data = jp.parse(json);
        } catch (Exception e) {
            data = e.getMessage();
        }
    }
}

```

```

    }
    return data;
}

@Override
protected void onPostExecute(String s) {
    super.onPostExecute(s);
    Log.d("res", s);
    try {
        JSONObject jsonObject = new JSONObject(s);
        String res = jsonObject.getString("status");
        if (res.compareTo("true") == 0) {
            Snackbar.make(fab, "Water Intake Added",
BaseTransientBottomBar.LENGTH_SHORT).show();
            dl.cancel();
        } else {
            Snackbar.make(fab, "Failed! Try Again",
BaseTransientBottomBar.LENGTH_SHORT).show();
            dl.cancel();
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

}

public class Viewwaterintake extends AsyncTask<String, String, String>
{
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        dl = Util.dailog(WaterActivity.this);
        dl.show();
    }

    @Override
    protected String doInBackground(String... strings) {
        String data = null;
        RestAPI restAPI = new RestAPI();

        try {
            JSONParse jp = new JSONParse();
            JSONObject json = restAPI.getWaterintake(strings[0]);
            data = jp.parse(json);
        } catch (Exception e) {
            data = e.getMessage();
        }
        return data;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        Log.d("water", s);
        try {
            JSONObject jsonObject = new JSONObject(s);
            String res = jsonObject.getString("status");
            if (res.compareTo("ok") == 0) {
                Log.d("work", "working");
                JSONArray Arr = jsonObject.getJSONArray("Data");
                String res1 = Arr.getString(0);

```

```

        JSONObject jsonObject1 = new JSONObject(res1);
        String intake = jsonObject1.getString("data0");

        watertxt.setText(intake + " L");

//
        dl.cancel();

    } else {

        dl.cancel();

    }

} catch (JSONException e) {
    e.printStackTrace();
}

}

}

public class todaylist extends AsyncTask<String, String, String> {
    List<Water> waterList = new ArrayList<>();

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
//        dl = Util.dailog(WaterActivity.this);
//        dl.show();
    }

    @Override
    protected String doInBackground(String... strings) {
        String data = null;
        RestAPI restAPI = new RestAPI();

        try {
            JSONParse jp = new JSONParse();
            JSONObject json = restAPI.getwaterhistory(strings[0],
strings[1], strings[2]);
            data = jp.parse(json);
        } catch (Exception e) {
            data = e.getMessage();
        }

        return data;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        Log.d("today", s);

        try {

            JSONObject jsonObject = new JSONObject(s);
            JSONArray array = jsonObject.getJSONArray("Data");
            Log.d("Json", array.toString());
            float sum = 0;
            for (int i = 0; i < array.length(); i++) {
                JSONObject jsonObj = array.getJSONObject(i);
                Water water = new Water(jsonObj.getString("data0"),

```

```

        jsonObj.getString("data1"),
        jsonObj.getString("data2"),
        jsonObj.getString("data3"),
        jsonObj.getString("data4"));

        sum += Float.parseFloat(water.getData2());
        Log.d("sum", String.valueOf(sum));

        totalintake.setText("Today's Water Intake: " +
String.valueOf(sum) + " L");
        waterList.add(water);
    }
    adapter = new WaterAdapter(waterList,
getApplicationContext());
    recyclerView.setAdapter(adapter);
    adapter.notifyDataSetChanged();
    dl.cancel();

    } catch (JSONException e) {
        e.printStackTrace();
    }
}

@Override
protected void onResume() {
    super.onResume();
    new Viewwaterintake().execute(Util.getSP(getApplicationContext()));

    new todaylist().execute(Util.getSP(getApplicationContext()),
formattedDate, formattedDate);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is
present.
    getMenuInflater().inflate(R.menu.water_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    switch (id) {
        case R.id.waterhistory:
            startActivity(new Intent(getApplicationContext(),
WaterHistoryActivity.class));
            return true;

        default:
            return super.onOptionsItemSelected(item);
    }
}
}

```

The provided code is an Android activity called "WaterActivity" that tracks water intake. It includes the following features:

- Displays a list of water intake records using a RecyclerView.
- Allows users to add new water intake entries using a FloatingActionButton.

Retrieves the current water intake from a web service and updates the UI.
Retrieves the water intake history for the current date and populates the RecyclerView.

Provides options to view water intake history through a menu.

In summary, the code helps users track and manage their water intake.

Water History Activity

```
package com.example.aibaseddietandexercise;

import android.app.DatePickerDialog;
import android.app.Dialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.aibaseddietandexercise.Adapter.WaterAdapter;
import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.Model.Water;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;
import com.google.android.material.snackbar.BaseTransientBottomBar;
import com.google.android.material.snackbar.Snackbar;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

public class WaterHistoryActivity extends AppCompatActivity {
    ImageView fromimg, tillimg;
    EditText fromedt, tilledt;
    CardView btn;
    Dialog dl;
    RecyclerView recyclerView;
    WaterAdapter adapter;
    TextView txt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_water_history);
        fromimg = (ImageView) findViewById(R.id.datepickerfrom);
        tillimg = (ImageView) findViewById(R.id.datepickertill);
        fromedt = (EditText) findViewById(R.id.waterdatefrom);
```

```

        tiltedt = (EditText) findViewById(R.id.waterdatetill);
        btn = (CardView) findViewById(R.id.serach);
        txt = (TextView) findViewById(R.id.titletxt);

        recyclerView = (RecyclerView)
findViewById(R.id.recyclerview_waterhistory);
        LinearLayoutManager layoutManager = new
LinearLayoutManager(getApplicationContext());
        recyclerView.setLayoutManager(layoutManager);

        getSupportActionBar().setTitle("Water InTake History");

        fromimg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Calendar mcurrentDate = Calendar.getInstance();
                int mYear = mcurrentDate.get(Calendar.YEAR);
                int mMonth = mcurrentDate.get(Calendar.MONTH);
                int mDay = mcurrentDate.get(Calendar.DAY_OF_MONTH);

                DatePickerDialog mDatePicker;
                mDatePicker = new
DatePickerDialog(WaterHistoryActivity.this, new
DatePickerDialog.OnDateSetListener() {
                    public void onDateSet(DatePicker datepicker, int mYear,
int mMonth, int selectedday) {
                        // TODO Auto-generated method stub
                        /*      Your code      to get date and time      */

                        Date d = new Date(mYear, mMonth, selectedday);
                        SimpleDateFormat dateFormatter = new
SimpleDateFormat(
                            "20YY/MM/dd");
                        String strDate = dateFormatter.format(d);
                        Log.d("format", strDate);
                        fromedt.setText(strDate);
                    }
                }, mYear, mMonth, mDay);
                mDatePicker.setTitle("Select Date");
                mDatePicker.show();
            }
        });

        tillimg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Calendar mcurrentDate = Calendar.getInstance();
                int mYear = mcurrentDate.get(Calendar.YEAR);
                int mMonth = mcurrentDate.get(Calendar.MONTH);
                int mDay = mcurrentDate.get(Calendar.DAY_OF_MONTH);

                DatePickerDialog mDatePicker;
                mDatePicker = new
DatePickerDialog(WaterHistoryActivity.this, new
DatePickerDialog.OnDateSetListener() {
                    public void onDateSet(DatePicker datepicker, int mYear,

```



```

int mMonth, int selectedday) {
    // TODO Auto-generated method stub
    /*      Your code      to get date and time      */

    Date d = new Date(mYear, mMonth, selectedday);
    SimpleDateFormat dateFormatter = new
SimpleDateFormat(
        "20YY/MM/dd");
    String strDate = dateFormatter.format(d);
    Log.d("format", strDate);
    tilledt.setText(strDate);

    }
    }, mYear, mMonth, mDay);
    mDatePicker.setTitle("Select Date");
    mDatePicker.show();

    }
    });

    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (fromedt.getText().toString().isEmpty()) {
                Snackbar.make(btn, "select Date",
BaseTransientBottomBar.LENGTH_SHORT).show();
            } else if (tilledt.getText().toString().isEmpty()) {
                Snackbar.make(btn, "select Date",
BaseTransientBottomBar.LENGTH_SHORT).show();
            } else {
                new
Intake().execute(Util.getSP(getApplicationContext()),
fromedt.getText().toString(), tilledt.getText().toString());
            }
        }
    });

}

public class Intake extends AsyncTask<String, String, String> {
    List<Water> waterList = new ArrayList<>();

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        dl = Util.dailog(WaterHistoryActivity.this);
        dl.show();
    }

    @Override
    protected String doInBackground(String... strings) {

        String data = null;
        RestAPI restAPI = new RestAPI();

        try {
            JSONParse jp = new JSONParse();
            JSONObject json = restAPI.getwaterhistory(strings[0],
strings[1], strings[2]);
            data = jp.parse(json);

```

```

        } catch (Exception e) {
            data = e.getMessage();
        }
        return data;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        Log.d("res", s);
        recyclerView.setAdapter(null);

        try {

            JSONObject jsonObject = new JSONObject(s);
            String res = jsonObject.getString("status");
            if (res.compareTo("ok") != 0) {
                dl.cancel();
                Snackbar.make(btn, "No Result Found",
BaseTransientBottomBar.LENGTH_SHORT).show();
            }
            JSONArray array = jsonObject.getJSONArray("Data");
            Log.d("Json", array.toString());
            float sum = 0;
            for (int i = 0; i < array.length(); i++) {
                JSONObject jsonObj = array.getJSONObject(i);
                Water water = new Water(jsonObj.getString("data0"),
                    jsonObj.getString("data1"),
                    jsonObj.getString("data2"),
                    jsonObj.getString("data3"),
                    jsonObj.getString("data4"));

                sum += Float.parseFloat(water.getData2());
                Log.d("sum", String.valueOf(sum));

                txt.setText("Total Intake: " + String.valueOf(sum) + "
L");

                waterList.add(water);
            }
            if (waterList.size() > 0) {
                adapter = new WaterAdapter(waterList,
getApplicationContext());
                recyclerView.setAdapter(adapter);
                adapter.notifyDataSetChanged();
                dl.cancel();
            } else {
                dl.cancel();
                Snackbar.make(btn, "No Result Found",
BaseTransientBottomBar.LENGTH_SHORT).show();
            }

        } catch (JSONException e) {
            e.printStackTrace();
        }

    }

}
}

```

The above code is an Android activity called "WaterHistoryActivity" that displays a history of water intake. It includes the following features:

Allows the user to select a date range using date pickers.
 Retrieves water intake data for the selected date range from a web service.
 Populates a RecyclerView with the retrieved water intake records.
 Calculates and displays the total water intake for the selected date range.
 Shows a "No Result Found" message if no records are available for the selected date range.

In summary, the code helps users view their water intake history within a specified date range.

Diet Plan Activity

```
package com.example.aibaseddietandexercise;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.Model.DietPlan;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class DietPlanActivity extends AppCompatActivity {
    TextView bmitv, bmrtnv, created;
    LinearLayout cardview;
    LinearLayout cardviewlunch;
    LinearLayout snackcard;
    LinearLayout dinnercard;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_diet_plan);
        getSupportActionBar().setTitle("Diet Plan");
        bmitv = findViewById(R.id.bmi);
        bmrtnv = findViewById(R.id.bmr);
        created = (TextView) findViewById(R.id.created);
        cardview = findViewById(R.id.brecard);
        cardviewlunch = findViewById(R.id.lincard);
        snackcard = findViewById(R.id.linsnack);
        cardviewlunch = findViewById(R.id.lincard);
        dinnercard = findViewById(R.id.lindinner);
    }
}
```

```

        new GetPlan().execute(Util.getSP(getApplicationContext()));
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    public class GetPlan extends AsyncTask<String, String, String> {

        List<DietPlan> dietPlanList = new ArrayList<>();

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected String doInBackground(String... strings) {
            String data = null;
            RestAPI restAPI = new RestAPI();
            try {
                JSONParse jp = new JSONParse();
                JSONObject json = restAPI.getPlan(strings[0]);
                data = jp.parse(json);
            } catch (Exception e) {
                data = e.getMessage();
            }
            return data;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            Log.d("diet", s);
            try {
                JSONObject jsonObject = new JSONObject(s);
                String res = jsonObject.getString("status");
                if (res.compareTo("no") == 0) {
                    AlertDialog.Builder builder = new
AlertDialog.Builder(DietPlanActivity.this);
                    builder.setTitle("You Don't Have Diet Plan !");
                    builder.setCancelable(true);

                    builder.setPositiveButton("Create Plan",
(DialogInterface.OnClickListener) (dialog, which) -> {

                        new
MakePlan().execute(Util.getSP(getApplicationContext()));
                    });

                    // Set the Negative button with No name Lambda
OnClickListener method is use of DialogInterface interface.
                    builder.setNegativeButton("Exit",
(DialogInterface.OnClickListener) (dialog, which) -> {

                        dialog.cancel();
                    });
                }
            }
        }
    }

```

```

    });

    AlertDialog alertDialog = builder.create();
    alertDialog.show();
} else if (res.compareTo("ok") == 0) {
    //dietplan
    JSONArray jsonArray = jsonObject.getJSONArray("Data");
    Log.d("check", jsonArray.toString());
    String res0 = jsonArray.getString(0);
    JSONObject jsonObject1 = new JSONObject(res0);
    String bmi = jsonObject1.getString("BMI");
    String bmr = jsonObject1.getString("BMR");
    String date = jsonObject1.getString("Datetime");

    Log.d("bmi", bmi);
    Log.d("bmr", bmr);
    Log.d("date", date);
    bmitv.setText(bmi);
    bmriv.setText(bmr);
    created.setText("Created on: " + date);

    for (int i = 0; i < jsonArray.length(); i++) {
        JSONObject jsonObj = jsonArray.getJSONObject(i);

        ArrayList<String> url = new ArrayList<>();
        JSONArray array1 = jsonObj.getJSONArray("Data");

        Log.d("check1", String.valueOf(array1.length()));

        for (int j = 0; j < array1.length(); j++) {
            JSONObject jsonObj1 = array1.getJSONObject(j);

            //
            if
(jsonObj1.getString("data3").compareTo("Breakfast") == 0) {
                //
                Log.d("break",
jsonObj1.getString("data4"));
                url.add(jsonObj1.getString("data0"));
                url.add(jsonObj1.getString("data1"));
                url.add(jsonObj1.getString("data2"));
                url.add(jsonObj1.getString("data3"));
                url.add(jsonObj1.getString("data4"));
                url.add(jsonObj1.getString("data5"));
                Log.d("break",
jsonObj1.getString("data4"));

                LayoutInflater li =
LayoutInflater.from(getApplicationContext());
                View cv =
li.inflate(R.layout.breakfastlayout, null);
                TextView item = cv.findViewById(R.id.item);
                item.setText("  Food Item: " +
jsonObj1.getString("data4"));
                TextView item2 =
cv.findViewById(R.id.itemcalories);
                item2.setText("  Calories: " +
jsonObj1.getString("data5"));
                cardview.addView(cv);
            }

```

```

        if
(jsonObj1.getString("data3").compareTo("Lunch") == 0) {
//          Log.d("break",
jsonObj1.getString("data4"));

          Log.d("lunch",
jsonObj1.getString("data4"));

          LayoutInflater li =
LayoutInflater.from(getApplicationContext());
          View cv = li.inflate(R.layout.lunch_layout,
null);
          TextView item =
cv.findViewById(R.id.lunchitem);
          item.setText("  Food Item: " +
jsonObj1.getString("data4"));
          TextView item2 =
cv.findViewById(R.id.lunchitemcalories);
          item2.setText("  Calories: " +
jsonObj1.getString("data5"));
          cardviewlunch.addView(cv);
        }

        if
(jsonObj1.getString("data3").compareTo("Snacks") == 0) {
//          Log.d("break",
jsonObj1.getString("data4"));

          Log.d("lunch",
jsonObj1.getString("data4"));

          LayoutInflater li =
LayoutInflater.from(getApplicationContext());
          View cv = li.inflate(R.layout.snack_layout,
null);
          TextView item =
cv.findViewById(R.id.snackitem);
          item.setText("  Food Item: " +
jsonObj1.getString("data4"));
          TextView item2 =
cv.findViewById(R.id.snackitemcalories);
          item2.setText("  Calories: " +
jsonObj1.getString("data5"));
          snackcard.addView(cv);
        }

        if
(jsonObj1.getString("data3").compareTo("Dinner") == 0) {
//          Log.d("break",
jsonObj1.getString("data4"));

          Log.d("lunch",
jsonObj1.getString("data4"));

          LayoutInflater li =
LayoutInflater.from(getApplicationContext());
          View cv =
li.inflate(R.layout.dinner_layout, null);
          TextView item =
cv.findViewById(R.id.dinneritem);
          item.setText("  Food Item: " +

```

```

jsonObj1.getString("data4"));
        TextView item2 =
cv.findViewById(R.id.dinneritemcalories);
        item2.setText("  Calories: " +
jsonObj1.getString("data5"));
        dinnercard.addView(cv);
    }

    }
    Log.d("url1", url.toString());

    DietPlan plan =
        new DietPlan(jsonObj.getString("Did"),
            jsonObj.getString("BMI"),
            jsonObj.getString("BMR"),
            url
        );

    dietPlanList.add(plan);
    }

    } else {

    }
} catch (JSONException e) {
    e.printStackTrace();
}

//dietplan
    }

}

public class MakePlan extends AsyncTask<String, String, String> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected String doInBackground(String... strings) {
        String data = null;
        RestAPI restAPI = new RestAPI();
        try {
            JSONParse jp = new JSONParse();
            JSONObject json = restAPI.MakePlan(strings[0]);
            data = jp.parse(json);
        } catch (Exception e) {
            data = e.getMessage();
        }
        return data;
    }
}

```

```

@Override
protected void onPostExecute(String s) {
    super.onPostExecute(s);
    Log.d("resmak", s);
    try {
        JSONObject jsonObject = new JSONObject(s);
        String res = jsonObject.getString("status");
        if (res.compareTo("true") == 0) {

        } else {

        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
}
}

```

The provided code is an Android activity class for a Diet Plan feature. It fetches the diet plan data from a server and displays it in the app's UI. If no plan exists, it gives an option to create a new plan.

Chat Bot Activity

```

package com.example.aibaseddietandexercise;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;

import com.example.aibaseddietandexercise.Adapter.ChatsAdapter;
import com.example.aibaseddietandexercise.Model.Chats;
import com.example.aibaseddietandexercise.Service.chats.DialogFlowHelper;
import com.example.aibaseddietandexercise.Service.chats.IChatResponse;
import com.google.cloud.dialogflow.v2beta1.DetectIntentResponse;

import java.util.ArrayList;
import java.util.List;

public class ChatBotActivity extends AppCompatActivity implements
IChatResponse {

    RecyclerView recyclerView;
    EditText chats;
    ImageButton send;

    ChatsAdapter adapter;
    List<Chats> chatsList = new ArrayList<>();

    private int lastCheckedChatsCount = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

```



```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_chatbot);
getSupportActionBar().setTitle("Chat Bot");

recyclerView = (RecyclerView)
findViewById(R.id.recyclerview_chats);
chats = (EditText) findViewById(R.id.chat);
send = (ImageButton) findViewById(R.id.send);

adapter = new ChatsAdapter(chatsList, ChatBotActivity.this);
recyclerView.setAdapter(adapter);

send.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        onClickIbSendTextChats();
    }
});

@Override
public void onChatResponse(DetectIntentResponse detectIntentResponse) {
    if (detectIntentResponse != null)
        processOnChatsResponse(detectIntentResponse);
    else addMsgToChatsList("I didn't get that. Can you rephrase it
again?", true);
}

private void onClickIbSendTextChats() {
    try {
        //check chats count to close session after 30 chats due to
dialogflow limit
        lastCheckedChatsCount = lastCheckedChatsCount + 1;
        checkIfCloseDialogFlowSession();
        if (!((Activity) ChatBotActivity.this).isFinishing()) {
            if (!chats.getText().toString().trim().equals("")) {
                sendChatsToBot(chats.getText().toString());
                chats.setText("");
            }
        } else {
            Toast.makeText(ChatBotActivity.this, "System busy please
wait for a sec and try again", Toast.LENGTH_SHORT).show();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

//closing session because of dialogflow limit of per session
private void checkIfCloseDialogFlowSession() {
    if (lastCheckedChatsCount > 30) {
        DialogFlowHelper.closeSession(ChatBotActivity.this);
        lastCheckedChatsCount = 0;
    }
}

private void sendChatsToBot(String chats) {
    try {
        if (chats != null && !chats.trim().equals("")) {
            addMsgToChatsList(chats, false);

```

```

        DialogFlowHelper.sendChatsToBot(ChatBotActivity.this,
chats, this);
    }
    } catch (Exception e) {
        e.printStackTrace();
        Toast.makeText(ChatBotActivity.this, "SOMETHING WENT WRONG
!!!", Toast.LENGTH_SHORT).show();
    }
}

@SuppressLint("NotifyDataSetChanged")
private void addMsgToChatsList(String chats, boolean isAgentMsg) {
    try {
        Chats chat = new Chats();
        chat.setChat(chats);
        chat.setAgentChat(isAgentMsg);
        chatsList.add(chat);

        adapter.updateList(chatsList);
        adapter.notifyDataSetChanged();
        scrollRecyclerViewToBottom(recyclerView, chatsList);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void scrollRecyclerViewToBottom(RecyclerView recyclerView,
List<?> list) {
    try {
        RecyclerView.LayoutManager layoutManager =
recyclerView.getLayoutManager();
        if (layoutManager != null)
layoutManager.scrollToPosition(list.size() - 1);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void processOnChatsResponse(DetectIntentResponse
detectIntentResponse) {
    String chats = null;
    try {
        chats =
detectIntentResponse.getQueryResult().getFulfillmentText().trim();
    } catch (Exception e) {
        e.printStackTrace();
    }
    if (chats != null && !chats.trim().isEmpty())
addMsgToChatsList(chats, true);
    else addMsgToChatsList("I didn't get that. Can you rephrase it
again?", true);
}
}

```

The above code is an Android activity class for a Chat Bot feature. It enables users to chat with a bot using the DialogFlow service. The user's messages are sent to the bot, and the bot's responses are displayed in a RecyclerView. The activity includes methods to handle user input, send messages to the bot, receive bot responses, and update the chat list

accordingly. Additionally, there is logic to manage the DialogFlow session and limit the number of chats to stay within the session limits.

Change Password

```
package com.example.aibaseddietandexercise;

import android.app.Dialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import com.example.aibaseddietandexercise.Helper.Util;
import com.example.aibaseddietandexercise.WebServices.JSONParse;
import com.example.aibaseddietandexercise.WebServices.RestAPI;
import com.google.android.material.snackbar.BaseTransientBottomBar;
import com.google.android.material.snackbar.Snackbar;

import org.json.JSONException;
import org.json.JSONObject;

public class ChangePassword extends AppCompatActivity {

    Dialog dl;
    TextView title;
    EditText pass, cpass;
    CardView submit;
    String uid;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_change_password);
        // getSupportActionBar().hide();
        init();
    }

    public void init() {
        uid = getIntent().getStringExtra("uid");
        title = findViewById(R.id.title);
        pass = findViewById(R.id.pass);
        cpass = findViewById(R.id.cpass);
        submit = findViewById(R.id.submit);
        title.setText("Reset Password");

        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (validate()) {
                    new CP().execute(Util.getSP(getApplicationContext()),
```

```

pass.getText().toString(), cpass.getText().toString());
    }
    });
}

public boolean validate() {
    if (pass.getText().toString().length() == 0) {
        Snackbar.make(submit, "Enter Password",
BaseTransientBottomBar.LENGTH_SHORT).show();
    } else if (cpass.getText().toString().length() == 0) {
        Snackbar.make(submit, "Enter New Password",
BaseTransientBottomBar.LENGTH_SHORT).show();
    } else {
        return true;
    }
    return false;
}

public class CP extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        dl = Util.dailog(ChangePassword.this);
        dl.show();
    }

    @Override
    protected String doInBackground(String... strings) {
        String data = null;
        RestAPI restAPI = new RestAPI();
        try {
            JSONParse jp = new JSONParse();
            JSONObject json = restAPI.ChangePassword(strings[0],
strings[1], strings[2]);
            data = jp.parse(json);
        } catch (Exception e) {
            data = e.getMessage();
        }
        return data;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        Log.d("res", s);
        try {
            JSONObject jsonObject = new JSONObject(s);
            String res = jsonObject.getString("status");
            if (res.compareTo("true") == 0) {
                dl.cancel();

                Snackbar.make(submit, "Password Changed!",
BaseTransientBottomBar.LENGTH_SHORT).show();
                Util.setSP(getApplicationContext(), "");

                Intent intent = new Intent(getApplicationContext(),
Login.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |

```

```

Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
        finish();
    } else {
        dl.cancel();
        Snackbar.make(submit, "Failed !Try Again ",
BaseTransientBottomBar.LENGTH_SHORT).show();
    }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

}
}

```

The above code is an Android activity class for the "Change Password" feature. It allows users to reset their password by providing a new password and confirming it. The code includes methods for input validation, making an API call to change the password, and handling the response. Additionally, there are UI elements such as TextView, EditText, and CardView for displaying the title, password fields, and submit button respectively. If the password change is successful, the user is redirected to the login screen. Otherwise, an error message is shown.

Diet Plan

```

package com.example.aibaseddietandexercise.Model;

import java.util.ArrayList;

public class DietPlan {
    String Did;
    String BMI;
    String BMR;

    private ArrayList<String> data;

    public DietPlan(String did, String BMI, String BMR, ArrayList<String>
data) {
        Did = did;
        this.BMI = BMI;
        this.BMR = BMR;
        this.data = data;
    }

    public String getDid() {
        return Did;
    }

    public void setDid(String did) {
        Did = did;
    }

    public String getBMI() {
        return BMI;
    }

    public void setBMI(String BMI) {
        this.BMI = BMI;
    }

    public String getBMR() {
        return BMR;
    }
}

```

```

    }

    public void setBMR(String BMR) {
        this.BMR = BMR;
    }

    public ArrayList<String> getData() {
        return data;
    }

    public void setData(ArrayList<String> data) {
        this.data = data;
    }
}

```

The provided code defines a model class called `DietPlan` that represents a diet plan. It has the following properties:

`Did`: A string representing the ID of the diet plan.

`BMI`: A string representing the Body Mass Index (BMI) associated with the diet plan.

`BMR`: A string representing the Basal Metabolic Rate (BMR) associated with the diet plan.

`data`: An ArrayList of strings representing the data associated with the diet plan.

The class includes a constructor to initialize the properties, as well as getter and setter methods for accessing and modifying the properties.

Fitness Data

```

package com.example.aibaseddietandexercise.Model;

public class FitnessData {
    private String
steps,heartRate,bloodPressure,bloodGlucose,oxygen,bodyTemp;

    public String getSteps() {
        return steps;
    }

    public void setSteps(String steps) {
        this.steps = steps;
    }

    public String getHeartRate() {
        return heartRate;
    }

    public void setHeartRate(String heartRate) {
        this.heartRate = heartRate;
    }

    public String getBloodPressure() {
        return bloodPressure;
    }

    public void setBloodPressure(String bloodPressure) {
        this.bloodPressure = bloodPressure;
    }

    public String getBloodGlucose() {

```

```

        return bloodGlucose;
    }

    public void setBloodGlucose(String bloodGlucose) {
        this.bloodGlucose = bloodGlucose;
    }

    public String getOxygen() {
        return oxygen;
    }

    public void setOxygen(String oxygen) {
        this.oxygen = oxygen;
    }

    public String getBodyTemp() {
        return bodyTemp;
    }

    public void setBodyTemp(String bodyTemp) {
        this.bodyTemp = bodyTemp;
    }
}

```

The provided code defines a model class called `FitnessData` that represents fitness data. It has the following properties:

- `steps`: A string representing the number of steps taken.
- `heartRate`: A string representing the heart rate.
- `bloodPressure`: A string representing the blood pressure.
- `bloodGlucose`: A string representing the blood glucose level.
- `oxygen`: A string representing the oxygen level.
- `bodyTemp`: A string representing the body temperature.

The class includes getter and setter methods for each property, allowing access to and modification of the data stored in the object.

Steps

```

package com.example.aibaseddietandexercise.Model;

public class Steps {
    private String date, dayOfWeek, totalSteps;

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getDayOfWeek() {
        return dayOfWeek;
    }

    public void setDayOfWeek(String dayOfWeek) {
        this.dayOfWeek = dayOfWeek;
    }
}

```

```

    }

    public String getTotalSteps() {
        return totalSteps;
    }

    public void setTotalSteps(String totalSteps) {
        this.totalSteps = totalSteps;
    }
}

```

The provided code defines a model class called `Steps` that represents daily step data. It has the following properties:

- `date`: A string representing the date.
- `dayOfWeek`: A string representing the day of the week.
- `totalSteps`: A string representing the total number of steps for the day.

The class includes getter and setter methods for each property, allowing access to and modification of the data stored in the object.

Videos

```

package com.example.aibaseddietandexercise.Model;

public class Videos {
    String data0;
    String data1;
    String data2;
    String data3;
    String data4;

    public Videos(String data0, String data1, String data2, String data3,
String data4) {
        this.data0 = data0;
        this.data1 = data1;
        this.data2 = data2;
        this.data3 = data3;
        this.data4 = data4;
    }

    public String getData0() {
        return data0;
    }

    public void setData0(String data0) {
        this.data0 = data0;
    }

    public String getData1() {
        return data1;
    }

    public void setData1(String data1) {
        this.data1 = data1;
    }

    public String getData2() {
        return data2;
    }
}

```



```

    }

    public void setData2(String data2) {
        this.data2 = data2;
    }

    public String getData3() {
        return data3;
    }

    public void setData3(String data3) {
        this.data3 = data3;
    }

    public String getData4() {
        return data4;
    }

    public void setData4(String data4) {
        this.data4 = data4;
    }
}

```

The provided code defines a model class called `Videos` that represents a set of video data. It has the following properties:

- `data0`: A string representing video data.
- `data1`: A string representing video data.
- `data2`: A string representing video data.
- `data3`: A string representing video data.
- `data4`: A string representing video data.

The class includes a constructor that allows you to set the values for these properties upon object creation. It also includes getter and setter methods for each property, allowing access to and modification of the data stored in the object.

Chat Bot Chats

```

package com.example.aibaseddietandexercise.Model;

public class Chats {
    private String chat;
    private boolean isAgentChat;

    public String getChat() {
        return chat;
    }

    public void setChat(String chat) {
        this.chat = chat;
    }

    public boolean isAgentChat() {
        return isAgentChat;
    }

    public void setAgentChat(boolean agentChat) {
        isAgentChat = agentChat;
    }
}

```

```
}
}
```

The provided code defines a model class called `Chats` that represents a chat message. It has the following properties:

- `chat`: A string representing the chat message.
- `isAgentChat`: A boolean indicating whether the chat message is from an agent (true) or not (false).

The class includes getter and setter methods for each property, allowing access to and modification of the data stored in the object.

Water

```
package com.example.aibaseddietandexercise.Model;

public class Water {
    String data0;
    String data1;
    String data2;
    String data3;
    String data4;

    public Water(String data0, String data1, String data2, String data3,
String data4) {
        this.data0 = data0;
        this.data1 = data1;
        this.data2 = data2;
        this.data3 = data3;
        this.data4 = data4;
    }

    public String getData0() {
        return data0;
    }

    public void setData0(String data0) {
        this.data0 = data0;
    }

    public String getData1() {
        return data1;
    }

    public void setData1(String data1) {
        this.data1 = data1;
    }

    public String getData2() {
        return data2;
    }

    public void setData2(String data2) {
        this.data2 = data2;
    }

    public String getData3() {
        return data3;
    }
}
```

```

public void setData3(String data3) {
    this.data3 = data3;
}

public String getData4() {
    return data4;
}

public void setData4(String data4) {
    this.data4 = data4;
}
}

```

The code defines a model class called `Water` that represents water data. It has the following properties:

- `data0`, `data1`, `data2`, `data3`, `data4`: Strings representing different data points related to water.

The class includes getter and setter methods for each property, allowing access to and modification of the data stored in the object.

Technologies Used

Azure Data Studio

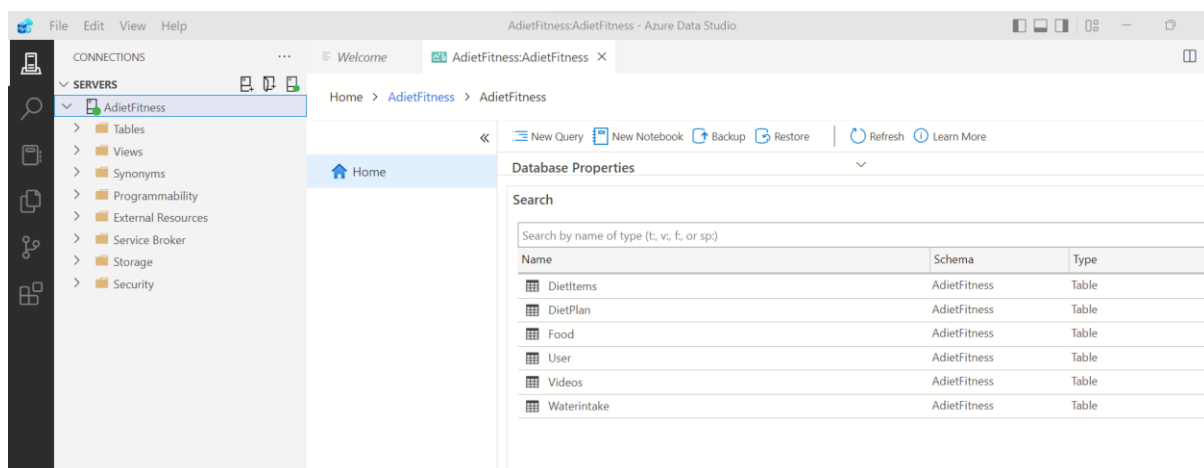


Fig 31

Data Base Properties of Food Data set: provide detailed information about the Food That includes name, food type based on body Goals, Time of Food Intake, protein, fats, carbohydrates and Total calories.

Export as Notebook

```

1 SELECT TOP (1000) [Fid]
2     ,[Name]
3     ,[Type]
4     ,[Consumed]
5     ,[Pro]
6     ,[Fats]
7     ,[Carbs]
8     ,[Totalcal]
9 FROM [AdietFitness].[AdietFitness].[Food]

```

Results Messages

	Fid	Name	Type	Consumed	Pro	Fats	Carbs	Totalcal
1	1	Nuts&Almonds	Healthy,Over Weight	Snacks	163	385	166	714
2	2	Melon(100g)	Healthy,Over Weight	Snacks	6	1	63	70
3	3	Tomato soup(medium dish)	Healthy,Over Weight	Snacks	27	26	130	183
4	4	Vegetables Soup (medium dish)	Under Weight,Healthy,Over We...	Lunch,Dinner	13	12	75	100
5	5	cream of mushroom soup	Under Weight	Lunch,Dinner	10	42	52	104
6	6	Vegetable and rice	Healthy,Over Weight	Lunch	15	12	93	120
7	7	Cheese Pizza with Vegetables...	Under Weight,Healthy	Lunch,Dinner	72	69	231	372
8	8	Boiled Potatoes	Over Weight	Dinner	44	2	266	312
9	9	Grilled Chicken with Vegetab...	Healthy,Over Weight	Lunch	57	24	132	213
10	10	Flafel	Under Weight,Healthy	Breakfast	17	23	42	82
11	11	Soup,Chicken with Rice(mediu...	Over Weight	Lunch	15	15	93	123
12	12	Cheese,Cream (100g)	Under Weight	Breakfast	46	332	54	432

Results Messages

	Fid	Name	Type	Consumed	Pro	Fats	Carbs	Totalcal
10	10	Flafel	Under Weight,Healthy	Breakfast	17	23	42	82
11	11	Soup,Chicken with Rice(mediu...	Over Weight	Lunch	15	15	93	123
12	12	Cheese,Cream (100g)	Under Weight	Breakfast	46	332	54	432
13	13	Yogurt,low fat(200g)	Over Weight	Dinner	77	23	108	208
14	14	Pastrami	Healthy,Over Weight	Breakfast	2	12	175	189
15	15	Broccoli Soup(cup)	Healthy,Over Weight	Dinner	15	31	62	108
16	16	Hummus(2 table spoons)	Under Weight,Healthy,Over We...	Breakfast,L...	11	20	47	78
17	17	Cantaloupe melon (cup)	Under Weight,Healthy,Over We...	Breakfast,S...	11	3	111	125
18	18	yogurt whole milk	Under Weight,Healthy	Breakfast	57	23	95	175
19	19	Fries	Under Weight	Lunch	23	39	216	278
20	20	kiwi	Under Weight,Healthy,Over We...	Breakfast,S...	13	6	167	186
21	21	blueberry	Under Weight,Healthy	Breakfast,S...	95	43	28	166
22	22	dates	Under Weight,Healthv	Breakfast,S...	33	7	28	68

Results Messages

	Fid	Name	Type	Consumed	Pro	Fats	Carbs	Totalcal
48	48	Biriyani	Under Weight,Healthy	Lunch,Dinner	20	25	85	610
49	49	Tandoori Chicken	Under Weight,Healthy	Lunch,Dinner	34	12	1	260
50	50	Puri Bhaji	Under Weight,Healthy	Breakfast	7	9	3	450
51	51	Idly	Under Weight,Healthy	Breakfast	2	0	28	130
52	52	Dosa	Under Weight,Healthy	Breakfast	5	2	23	230
53	53	Upma	Healthy,Over Weight	Breakfast	5	5	25	170
54	54	Pongal	Healthy,Over Weight	Breakfast	8	6	30	210
55	55	Vada	Under Weight,Un Healthy	Breakfast	5	12	17	210
56	56	Rava Dosai	Healthy,Over Weight	Breakfast	4	4	22	150
57	57	Paniyaram	Healthy,Over Weight	Breakfast	6	2	20	120
58	58	Tomato Rice	Healthy,Over Weight	Lunch,Dinner	3	3	20	120
59	59	Lemon Rice	Healthy,Over Weight	Lunch,Dinner	3	3	20	120

Fig 32

Data Base Properties of User Data Set: That Includes Name of the user, User mail id, User contact number, User Age, User Height and User weight parameters.

Run Cancel Disconnect Change Connection AdietFitness Estimated Plan Enable Actual Plan Enable SQLCMD

Export as Notebook

```

1 SELECT TOP (1000) [Uid]
2 , [Name]
3 , [Email]
4 , [Contact]
5 , [Age]
6 , [Gender]
7 , [Height]
8 , [Weight]
9 , [Password]
10 FROM [AdietFitness].[AdietFitness].[User]

```

Results Messages

	Uid	Name	Email	Contact	Age	Gender	Height	Weight
2	1001	Ronithsaidatta Pasupueti	pasupueti@ronithsaidatta03@gmail.com	9880333922	20	Male	153	60
3	1002	Sreenivas Rsn	sreenivasrsn@gmail.com	9398184334	20	Male	170	70
4	1003	Pushkar	pushkarpattidar400@gmail.com	6204379901	22	Male	180	74
5	1004	Jayendra Mani kumar	jayendra09ckt@gmail.com	9121649623	23	Male	176	65
6	1005	Prajwal	prajwaldurgoji16@gmail.com	8088765523	22	Male	178	65
7	1006	Nithin Krishhh	nithinkrishhh3571@gmail.com	9741736170	22	Male	164	92
8	1007	Rinki Sharma	abcd@gmail.com	9876543210	33	Female	152	60
9	1008	korakuti Gnaneswar	gnaneswarpandu1432@gmail.com	9550353257	22	Male	75	158
10	1009	Mushkan Bothra	mushkanbothra@gmail.com	7970422270	21	Female	167	54
11	1010	Nagendra	selanagendra1@gmail.com	9392539997	20	Male	168	51
12	1011	Puneeth	puneeth@gmail.com	9865234170	21	Male	142	58
13	1012	Sandeep	sandeep@gmail.com	9806532741	21	Male	125	45
14	1013	Ram	ram@gmail.com	9806352741	20	Male	152	65

Fig 33

Data Base Properties of Diet Generated to the users: That Includes Did (Data id), User id, BMI (Body Mass Index), BMR (Basal Metabolic Rate), Date and Time.

Run Cancel Disconnect Change Connection AdietFitness

Export as Notebook

```

1 SELECT TOP (1000) [Did]
2 , [Uid]
3 , [BMI]
4 , [BMR]
5 , [dt]
6 FROM [AdietFitness].[AdietFitness].[DietPlan]

```

Results Messages

	Did	Uid	BMI	BMR	dt
1	13	1000	23.71	685.26	02/22/2023 12:48:57
2	14	1001	23.95	727.2	02/23/2023 17:34:36
3	15	1003	19.86	753.61	02/23/2023 20:54:59
4	16	1004	21.49	707.25	02/24/2023 17:02:04
5	17	1005	21.01	714.8	02/24/2023 17:28:01
6	18	1006	35.03	634.8	02/26/2023 19:53:56
7	19	1008	30.77	563.21	03/20/2023 14:39:29
8	20	1009	19.83	568.28	04/03/2023 16:38:47
9	21	1011	29.46	471.79	04/04/2023 11:58:53
10	22	1012	29.49	390.28	04/04/2023 12:34:09
11	23	1013	28.81	519.23	04/06/2023 11:26:40
12	24	1014	24.89	442.03	04/06/2023 14:41:37

Fig 34

Databaseroperties of Food types generated to the users: That Includes User Id, Data ID, Time of Food Intake, Food name, Food Item calories.

↳ Export as Notebook

```
1 SELECT TOP (1000) [Id]
2     ,[Did]
3     ,[Uid]
4     ,[Type]
5     ,[Foodname]
6     ,[cal]
7 FROM [AdietFitness].[AdietFitness].[DietItems]
```

Results Messages

	Id	Did	Uid	Type	Foodname	cal
1	23	13	1000	Breakfast	Apple	137
2	24	13	1000	Lunch	Hummus(2 table spoons)	78
3	25	13	1000	Lunch	Grilled Chicken with Vegetab...	213
4	26	13	1000	Snacks	blueberry	166
5	27	13	1000	Dinner	Egg	95
6	28	13	1000	Dinner	Moong	694
7	29	14	1001	Breakfast	Egg	95
8	30	14	1001	Lunch	Chicken(grilled)	524
9	31	14	1001	Snacks	Melon(100g)	70
10	32	14	1001	Dinner	Cantaloupe melon (cup)	125
11	33	15	1003	Breakfast	Pastrami	189
12	34	15	1003	Lunch	Cheese Pizza with Vegetables...	372
13	35	15	1003	Snacks	kiwi	186

Fig 35

Database Properties of Videos Generation: Database properties that includes video categories Based on Video Type(Exercise or motivation or healthy habits), BMI (BODY MASS INDEX), Name and Video link.

↳ Export as Notebook

```
1 SELECT TOP (1000) [Vid]
2     , [Type]
3     , [BMI]
4     , [Name]
5     , [Link]
6 FROM [AdietFitness].[AdietFitness].[Videos]
```

Results		Messages				
	Vid	Type	BMI	Name	Link	
1	1	Exercise	Healthy	Top 10 Morning Exercises To ...	PG2f3GF5RlI	
2	2	Exercise	Healthy	1.0 Mile Happy Walk Walk a...	X3q5e1pV4pc	
3	3	Exercise	Healthy	20 Min Full Body Workout Rou...	AzV3EA-1-yM	
4	4	Exercise	Under Weight	8 Best Exercises To Build Mu...	1A-diBuGy6I	
5	5	Exercise	Under Weight	5 BEST EXERCISES to Gain Wei...	zlyqr9bNs1E	
6	6	Exercise	Under Weight	Easy Yoga Asanas For Weight ...	zpxHe8NxLmI	
7	7	Exercise	Over Weight	14 Days Weight Loss Challeng...	LhL5SNZfnQs	
8	8	Exercise	Over Weight	LOSE BELLY FAT IN 7 DAYS Cha...	digpucxGbMo	
9	9	Exercise	Over Weight	FULL BODY FAT LOSS in 14 Day...	Dj0bk8TFWdY	
10	10	Motivation	Healthy	Healthy Lifestyle	Cg_GW7yhq20	
11	11	Motivation	Healthy	'POWER OF DISCIPLINE' (ft. A...	KSf3vde-odQ	
12	12	Motivation	Healthy	What is healthy lifestyle?	Vv8fMp3Jglo	
13	13	Motivation	Over Weight	My Extreme Transformation Jo...	aNidcdee7Xs	

Fig 36

Database Properties of Water Generation: That includes Water id, User id, Water quantity, Date and Time to track the user water intake on Daily bases.

↳ Export as Notebook

```
1 SELECT TOP (1000) [Wid]
2     , [uid]
3     , [quantity]
4     , [date]
5     , [dt]
6 FROM [AdietFitness].[AdietFitness].[Waterintake]
```

Results		Messages				
	Wid	uid	quantity	date	dt	
1	1	1000	1	2023/02/21	2023/02/21 11:29	
2	2	1000	0.6	2023/02/22	2023/02/22 10:31	
3	3	1000	0.8	2023/02/22	2023/02/22 11:04	
4	4	1001	6.0	2023/02/28	2023/02/28 20:49	
5	5	1000	50	2023/03/02	2023/03/02 17:36	
6	6	1000	1	2023/03/03	2023/03/03 15:02	
7	7	1001	3.2	2023/04/03	2023/04/03 21:51	
8	8	1001	4	2023/04/04	2023/04/04 11:11	
9	9	1001	2.87	2023/04/03	2023/04/04 13:04	
10	10	1001	3	2023/04/06	2023/04/06 14:38	
11	11	1001	2.56	2023/04/05	2023/04/06 14:38	
12	12	1017	3	2023/04/06	2023/04/06 19:38	
13	13	1001	2.5	2023/04/07	2023/04/07 10:17	

Fig 37

Chatbot Dialog flow Essentials

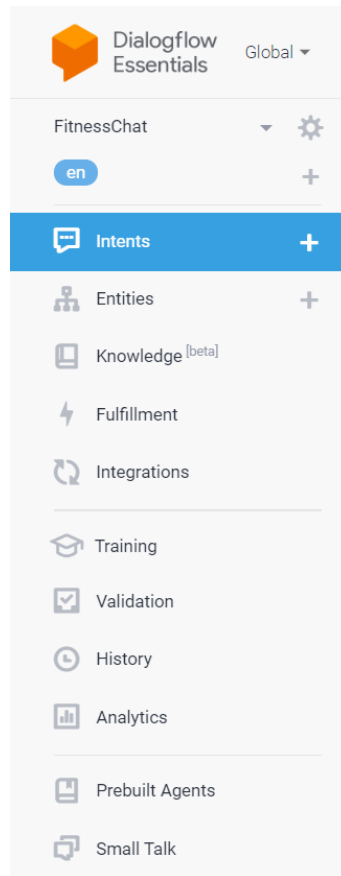


Fig 38

Dialogflow Essentials is a powerful tool for building chatbots and virtual assistants that can engage in natural and interactive conversations with users. With Dialogflow Essentials, developers can design conversational flows by creating intents, which represent the user's intention or query, and defining corresponding responses. The chatbot utilizes machine learning algorithms to understand user inputs, handle context, and provide accurate and contextually relevant responses. It supports various input types, including text, voice, and rich messages, making it versatile for different user interactions. Dialogflow Essentials also enables integration with various messaging platforms and allows developers to easily manage and train the chatbot through the web-based interface, facilitating the creation of dynamic and intelligent conversational experiences.

.

• Default Welcome Intent SAVE

” Add user expression

” just going to say hi

” heya

” hello hi

” howdy

” hey there

” hi there

” greetings

” hey

” long time no see

” hello

Fig 39

Intents

Dialogue flow plays an essential role in the functioning of chatbot in this application. The chatbot's components, which we may teach and programme, include intents, entities, knowledge, fulfilment, integrations, validation, history, analytics, prebuilt agents, and small conversation

We should add the greetings that can be made by the user as many as we desired in this welcome intent of the chatbot

Responses ?

DEFAULT +

Text Response

1 Hi! How are you doing?

2 Hello! How can I help you?

3 Good day! What can I do for you today?

4 Greetings! How can I assist?

5 Enter a text response variant

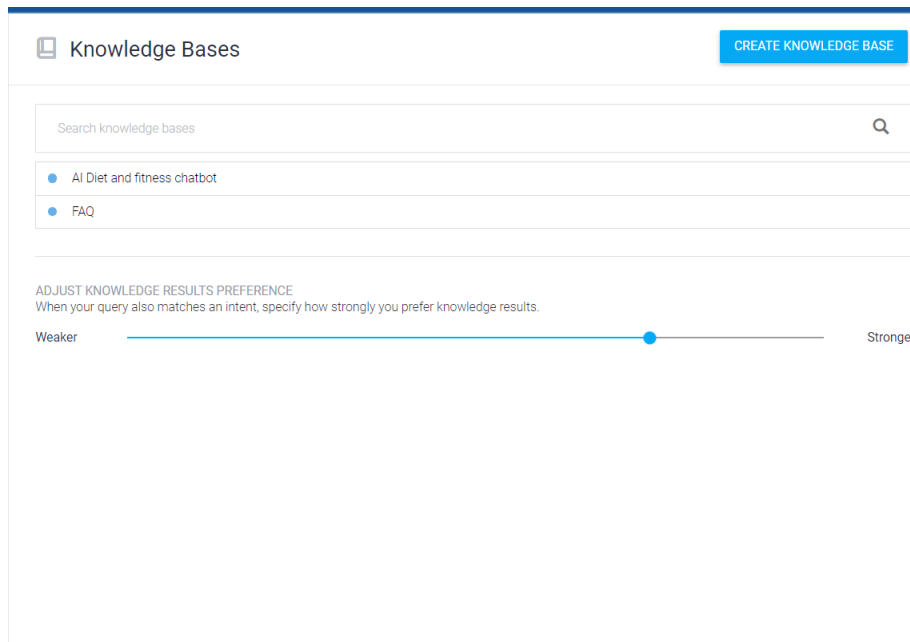
ADD RESPONSES

☐ Set this intent as end of conversation ?

Fig 40

This is responsible for generating the responses to the above added greetings. We can add the greet backs to the user by this chatbot

Knowledge Base



Knowledge Bases CREATE KNOWLEDGE BASE

Search knowledge bases Q

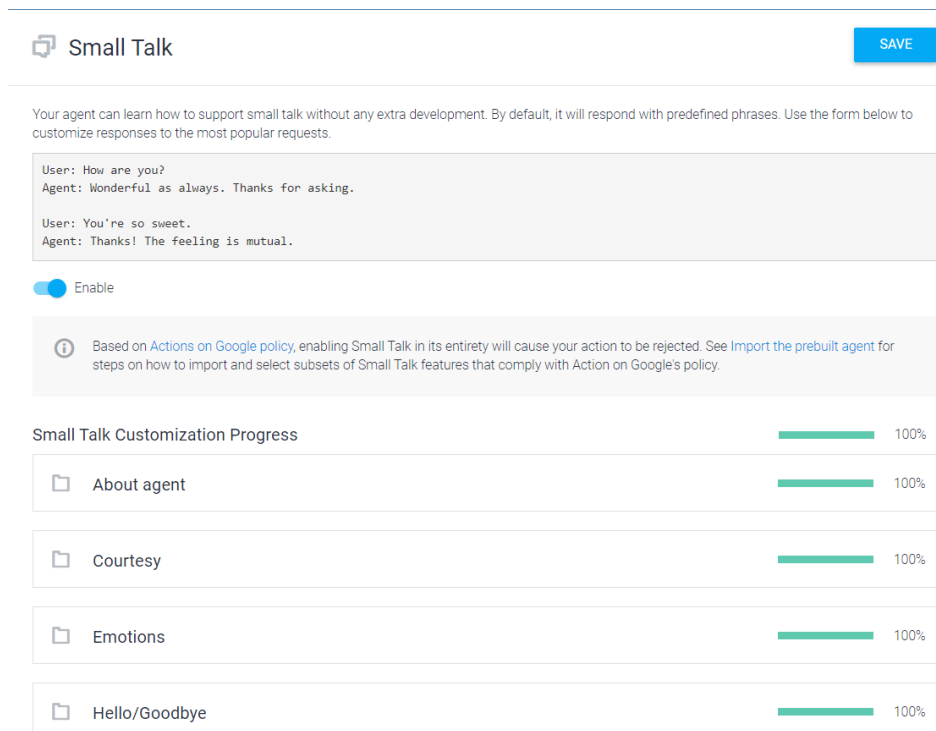
- AI Diet and fitness chatbot
- FAQ

ADJUST KNOWLEDGE RESULTS PREFERENCE
When your query also matches an intent, specify how strongly you prefer knowledge results.

Weaker Stronger

Fig 41

This module helps in training the chat bot that includes the answers to the all the kinds of questions corresponding to the diet, exercises and physical well being



Small Talk SAVE

Your agent can learn how to support small talk without any extra development. By default, it will respond with predefined phrases. Use the form below to customize responses to the most popular requests.

User: How are you?
Agent: Wonderful as always. Thanks for asking.

User: You're so sweet.
Agent: Thanks! The feeling is mutual.

☒ Enable

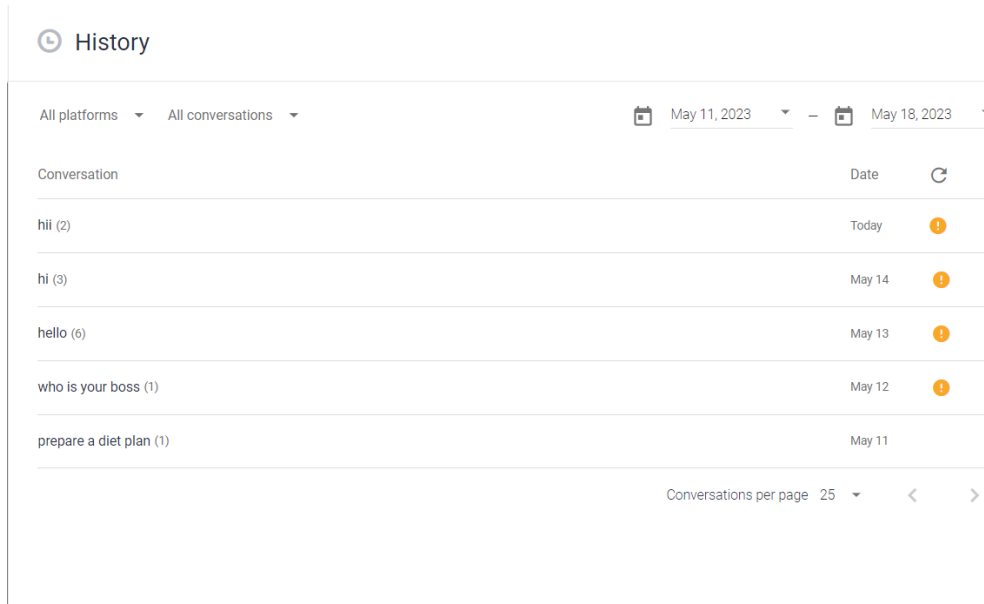
Based on [Actions on Google policy](#), enabling Small Talk in its entirety will cause your action to be rejected. See [Import the prebuilt agent](#) for steps on how to import and select subsets of Small Talk features that comply with Action on Google's policy.

Small Talk Customization Progress

About agent	100%
Courtesy	100%
Emotions	100%
Hello/Goodbye	100%

Fig 42

This module supports the small talks that includes about the agent, hello's and see you's, and the general information of this application



History	
All platforms ▾ All conversations ▾	
May 11, 2023 – May 18, 2023	
Conversation	Date
hii (2)	Today
hi (3)	May 14
hello (6)	May 13
who is your boss (1)	May 12
prepare a diet plan (1)	May 11

Conversations per page 25

Fig 43

This module helps in providing the total history of the queries that's been asked. This includes the frequency of the same kind of queries along with date and time

CHAPTER 5

FEASIBILITY REPORT

A feasibility study is a high-level capsule version of the entire process intended to answer several questions: What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? A feasibility study is conducted once the problem is clearly understood. A feasibility study is necessary to determine that the proposed system is Feasible by considering the technical, Operational, and Economical factors. By having a detailed feasibility study the management will have a clear-cut view of the proposed system.

The following feasibilities are considered for the project to ensure that the project is variable and it does not have any major obstructions. Feasibility study encompasses the following things:

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

In this phase, we study the feasibility of all proposed systems and pick the best feasible solution for the problem. The feasibility is studied based on three main factors as follows.

TECHNICAL FEASIBILITY

In this step, we verify whether the proposed systems are technically feasible or not. i.e., all the technologies required to develop the system are available readily or not.

Technical Feasibility determines whether the organization has the technology and skills necessary to carry out the project and how this should be obtained. The system can be feasible because of the following grounds:

All necessary technology exists to develop the system.

This system is too flexible and it can be expanded further.

This system can give guarantees of accuracy, ease of use, reliability and data security.

This system can give instant responses to inquiries.

Our project is technically feasible because all the technology needed for our project is readily available.

Operating System	: Android v5.0 or Higher
Languages	: JAVA
Database System	: SQLite
Documentation Tool	: MS – Word

ECONOMIC FEASIBILITY

Economically, this project is completely feasible because it requires no extra financial investment and concerning time, it's completely possible to complete this project in 6 months. In this step, we verify which proposal is more economical. We compare the financial benefits of the new system with the investment. The new system is economically feasible only when the financial benefits are more than the investments and expenditures. Economic Feasibility determines whether the project goal can be within the resource limits allocated to it or not. It must determine whether it is worthwhile to process the entire project or whether the benefits obtained from the new system are not worth the costs. Financial benefits must be equal to or exceed the costs. In this issue, we should consider:

The cost to conduct a full system investigation.

The cost of h/w and s/w for the class of application being considered.

The development tool.

The cost of maintenance etc...

Our project is economically feasible because the cost of development is very minimal when compared to the financial benefits of the application.

OPERATIONAL FEASIBILITY

In this step, we verify different operational factors of the proposed systems like manpower, time etc., and whichever solution uses fewer operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement. Operational Feasibility determines if the proposed system satisfied user objectives and could be fitted into the current system operation.

The methods of processing and presentation are completely accepted by the clients since they can meet all user requirements.

The clients have been involved in the planning and development of the system.

The proposed system will not cause any problems under any circumstances.

Our project is operationally feasible because the time requirements and personnel requirements are satisfied. We are a team of four members and we worked on this project for three working months.

CHAPTER 6

TESTING

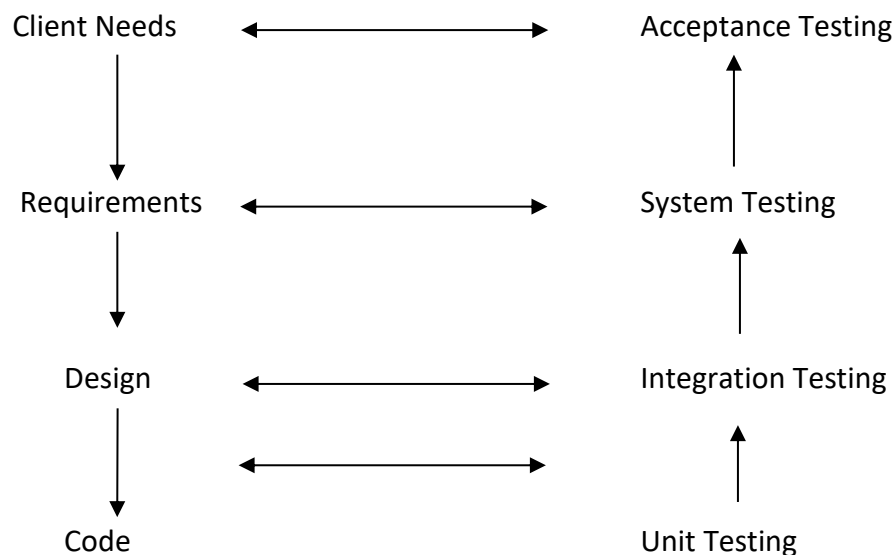
As the project is on a bit large scale, we always need testing to make it successful. If each component works properly in all respect and gives desired output for all kinds of inputs then the project is said to be successful. So the conclusion is to make the project successful, it needs to be tested.

The testing done here was System Testing checking whether the user requirements were satisfied. The code for the new system has been written completely using JAVA as the coding language and Android Studio as the interface for front-end designing. using Python as the coding language and Django Framework as the interface for front-end designing. The new system has been tested well with the help of the users and all the applications have been verified from every nook and corner of the user.

Although some applications were found to be erroneous these applications have been corrected before being implemented. The flow of the forms is very much by the actual flow of data.

LEVELS OF TESTING

To uncover the errors, present in different phases we have the concept of levels of testing. The basic levels of testing are:



A series of testing is done for the proposed system before the system is ready for user acceptance testing. The steps involved in Testing are:

UNIT TESTING

Unit testing focuses verification efforts on the smallest unit of the software design, the module. This is also known as “Module Testing”. The modules are tested separately. This testing is carried out during the programming stage itself. In this testing, each module is found to be working satisfactorily as regards the expected output from the module.

INTEGRATION TESTING

Data can be grossed across an interface; one module can have adverse effects on another. Integration testing is systematic testing for the construction of the program structure while at the same time conducting tests to uncover errors associated with the interface. The objective is to take unit-tested modules and build a program structure. All the modules are combined and tested as a whole. Here correction is difficult because the isolation of cause is complicated by the vast expense of the entire program. Thus, in the integration testing stop, all the errors uncovered are corrected for the test testing steps.

SYSTEM TESTING

System testing is the stage of implementation that is aimed ensuring that the system works accurately and efficiently for live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, then the goal will be successfully achieved.

VALIDATION TESTING

After integration testing software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begins, validation test begins. Validation tests can be defined in many ways. But the simple definition is that validation succeeds when the software function in a manner that can reasonably be expected by the customer. After the validation test has been conducted one of two possible conditions exists.

One is the function or performance characteristics conform to specifications and are accepted and the other is a deviation from the specification is uncovered and a deficiency list is created. A proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

OUTPUT TESTING

After performing validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated by the system under consideration. Here the output format is considered in two ways, one is on the screen and the other is the printed format. The output format on the screen is found to be correct as the format was designed in the system design phase according to the user's needs.

For the hard copy also, the output comes as the specified requirements by the users. Hence output testing does not result in any corrections in the system.

USER ACCEPTANCE TESTING

User acceptance of a system is the key factor to the success of any system. The system under study is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

TEST CASES

REGISTRATION

To begin with login, the user needs to register by filling up basic registration details. There are multiple fields on the registration page and every field has to fill by the user. The user cannot use the character in the login id field.

LOGIN

Login id and password are kept compulsory fields, and if the id or password doesn't match then it will show an error message.

VALIDATION CRITERIA

In each form, no field which is not nullable should be left blank.

All numeric fields should be checked for non-numeric values. Similarly, text fields like names should not contain any numeric characters.

All primary keys should be automatically generated to prevent the user from entering any existing key.

Use of error handling for each Save, Edit, delete and other important operations.

Whenever the user Tabs out or Enter from a text box, the data should be validated and if it is invalid, the focus should again be sent to the text box with the proper message.

Here are some test cases for the mentioned functionality:

Test Cases for Generating Diet:

Test Case: Valid User Details

Description: Attempt to generate a diet plan with valid user details.

Input: User enters accurate height and weight information.

Expected Output: The application should successfully calculate the BMI and BMR based on the provided details and generate a personalized diet plan.

Test Case: Invalid User Details

Description: Attempt to generate a diet plan with invalid user details.

Input: User enters incorrect or unrealistic height and weight values.

Expected Output: The application should display an error message indicating that the user details are invalid and prompt the user to enter accurate information.

Test Cases for Generating Exercises:

Test Case: Valid BMI and BMR

Description: Attempt to generate exercise recommendations with valid BMI and BMR values.

Input: User's BMI and BMR fall within a healthy range.

Expected Output: The application should suggest a variety of exercises suitable for the user's BMI and BMR, providing a diverse set of options to choose from.

Test Case: Invalid BMI and BMR

Description: Attempt to generate exercise recommendations with invalid BMI and BMR values.

Input: The user's BMI and BMR values are unrealistic or outside of a healthy range.

Expected Output: The application should display an error message indicating that the provided BMI and BMR values are invalid and prompt the user to enter accurate information.

Test Cases for Synchronizing Google Fit Data:**Test Case: Successful Data Sync**

Description: Attempt to sync Google Fit data with the application.

Input: User grants permission and successfully syncs their health metrics, including steps, heart rate, blood pressure, oxygen saturation, blood glucose, and body temperature.

Expected Output: The application should display the synced data accurately and use it to send relevant notifications and reminders to the user.

Test Cases for Active Notifications:**Test Case: Walk More Notification**

Description: Test the walk more notification functionality.

Input: User's step data indicates a sedentary behavior.

Expected Output: The application should send a notification to the user, encouraging them to walk more and improve their activity level.

Test Case: Stay Hydrated Reminder

Description: Test the reminder to stay hydrated functionality.

Input: User's step data indicates a high activity level and potential dehydration risk.

Expected Output: The application should send a reminder notification to the user, prompting them to stay hydrated and maintain a balanced water intake.

Test Cases for Chatbot:**Test Case: Valid User Question**

Description: Test the chatbot's response to a valid user question.

Input: User asks a health-related question or seeks guidance on diet or exercise.

Expected Output: The chatbot should provide a relevant and accurate response based on the user's question, utilizing artificial intelligence and trained knowledge-based data.

Test Case: Invalid User Question

Description: Test the chatbot's response to an invalid or unrelated user question.

Input: User asks a question that is unrelated to health, diet, or exercise.

Expected Output: The chatbot should display a message indicating that it does not understand the user's query and prompt the user to ask a relevant question.

Test Cases for Training & Motivational Videos:**Test Case: Recommended Videos**

Description: Test the application's recommendation of workout training and motivational videos.

Input: User's fitness level and preferences are considered in generating video recommendations.

Expected Output: The application should suggest high-quality videos that are tailored to the user's fitness level preferences, and goals, providing a source of motivation and guidance.

These test cases cover different scenarios for the mentioned functionality, including valid and invalid inputs, accurate data processing, and appropriate responses from the application. Additional test cases can be created based on specific requirements and functionality of the application.

ADVANTAGES AND LIMITATIONS**ADVANTAGES**

The system can be accessed anytime, anywhere.

The system will create an individualized diet plan.

The user can chat with the system to get answers to fitness-related questions.

The system is efficient to access and use.

LIMITATIONS

If any data is entered wrong, the system can produce incorrect results.

FEATURES**LOAD BALANCING**

Since the system will be available only after the admin logs in the amount of load on the server will be limited to the period of admin access.

EASY ACCESSIBILITY

Records can be easily accessed and stored and other information respectively.

USER-FRIENDLY

The website/application will be giving a very user-friendly approach for all users.

EFFICIENT AND RELIABLE

Maintaining the all-secured database on the server which will be accessible according to the user requirement without any maintenance cost will be very efficient as compared to storing all the customer data on the spreadsheet or physically in the record books.

EASY MAINTAINANCE

Android AI Diet with Fitness App is designed as an easy way. So, maintenance is also easy.

CONCLUSION

This was our project of System Design for an Android AI Diet with Fitness App developed in Java programming language. The Development of this system takes a lot of effort from us. We think this system gave a lot of satisfaction to all of us. Though every task is never said to

be perfect in this development field even more improvement may be possible in this application. We learned so many things and gained a lot of knowledge about the development field. We hope this will prove fruitful to us.

REFERENCES

- [1] Sak. J, Suchodolska. M, “Artificial Intelligence in Nutrients Science Research: A Review” *Nutrients*. 2021;13[2]:322.
- [2] Nilsson. NJ, “The quest for artificial intelligence” Cambridge University Press. 2009.
- [3] Matusheski. N, Caffrey. A, “Christensen L, Mezgec S, Surendran S, Hjorth MF, et al. Diets, nutrients, genes and the microbiome: Recent advances in personalised nutrition” *British Journal of Nutrition*. 2021;1-24.
- [4] Adams. SH, Anthony. JC, Carvajal. R, Chae. L, Kho.o CSH, Latulippe. ME, “Perspective: guiding principles for the implementation of personalized nutrition approaches that benefit health and function” *Advances in Nutrition*. 2020;11[1]:25-34.
- [5] Rozga M, Latulippe ME, Steiber A. “Advancements in Personalized Nutrition Technologies: Guiding Principles for Registered Dietitian Nutritionists” *Journal of the Academy of Nutrition and Dietetics*. 2020;120[6]:1074-85.
- [6] Shim. JS, Oh. K, Kim. HC, “Dietary assessment methods in epidemiologic studies. *Epidemiol Health*. 2014;36:e2014009.
- [7] Ji. Y, Plourde. H, Bouzo. V, Kilgour.. RD, Cohen. TR, “Validity and Usability of a Smartphone Image-Based Dietary Assessment App Compared to 3-Day Food Diaries in Assessing Dietary Intake Among Canadian Adults: Randomized Controlled Trial” *JMIR mHealth and uHealth*. 2020;8[9]:e16953.
- [8] Gibson. RS, Charrondiere. UR, Bell. W, “Measurement errors in dietary assessment using self-reported 24-hour recalls in low-income countries and strategies for their prevention” *Advances in Nutrition*. 2017;8[6]:980-91.
- [9] Ventura. AK, Loken. E, Mitchell. DC, Smiciklas-Wright. H, Birch LL, “Understanding reporting bias in the dietary recall data of 11-year-old girls” *Obesity [Silver Spring]*. 2006;14[6]:1073-84.
- [10] Hjartåker. A, Andersen. LF, Lund. E, “Comparison of diet measures from a food-frequency questionnaire with measures from repeated 24-hour dietary recalls” *The Norwegian Women and Cancer Study. Public health nutrition*. 2007;10[10]:1094-103.