

BMI CALCULATOR USING FLUTTER

INTERDISCIPLINARY PROJECT

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Technology Degree in Information Technology

by

GOPI KRISHNA M (41120103)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-1 UNIVERSITY BY UGC

Accredited “A++” by NAAC

12B Status by UGC |Approved by AICTE

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI-600119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

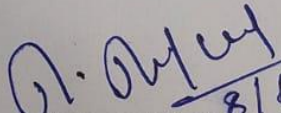
Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **M GOPI KRISHNA(41120103)** who carried out the project entitled **BMI CALCULATOR USING FLUTTER** under my supervision from Jan 2024 to May 2024.

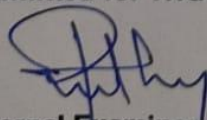

8/5/24
Internal Guide

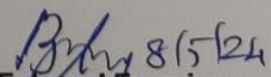
Dr. MS. SANGEENGRANA M.E.,


Head of the Department

Dr. R. SUBHASHINI M.E., Ph.D.,

Submitted for viva voce Examination held on 08-05-2024


Internal Examiner


8/5/24
External Examiner

DECLARATION

ACKNOWLEDGEMENT

I, **GOPI KRISHNA M (41120103)** hereby declare that the Project Report titled **BMI CALCULATOR USING FLUTTER** completed by me under the guidance of **Dr. MS. SANGEENGRANA M.E.**, is submitted in partial fulfilment of the requirements for the award of Bachelor in Technology Degree in **INFORMATION TECHNOLOGY**.

DATE: 08-05-2024

PLACE: CHENNAI

M. Gopinath

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** at **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean, School of Computing**, **Dr. R. Subhashini, M.E., Ph.D., Head of the Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. MS. SANGEENGRANA M.E.**, for her valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of my phase-2 project work.

I wish to express my thanks to all teaching and non-teaching staff members of the **Department of Information Technology** who were helpful in many ways throughout the completion of the project.

CERTIFICATE OF COMPLETION



HACKWIT TECHNOLOGIES PVT LTD
Redefine by Technologies

149, 1C, 2nd Floor, Mount Poonamallee Rd,
Ramapuram, Chennai - 600089.
admin@hackwittechnologies.com
www.hackwittechnologies.com

10.02.2024

CERTIFICATE FOR INTERNSHIP

TO WHOMSOEVER IT MAY CONCERN

This is to Certify that **Gopi Krishna M (Reg No: 41120103)** student of Sathyabama Institute of Science and Technology, B. Tech Information Technology, has successfully completed the Internship at Hackwit Technologies from 22nd January to 30th January 2024.

During the internship, **Gopi Krishna M** has closely worked as a part of the **Flutter** by carrying out the project entitled "**BMI Calculator**".

He demonstrated good developing and learning skill with self-motivated attitude to learn new things. His contribution to implement the Flutter was good. We wish him all the best for his future endeavors.

Best Regards,

A handwritten signature in black ink, appearing to be 'Ravikumar S', written over a faint circular watermark.

Ravikumar S

CEO & Managing Director

Hackwit Technologies Pvt Ltd

CERTIFICATE OF COMPLETION



ABSTRACT

In this digital age, health and fitness have become increasingly significant, prompting the need for accessible tools to monitor and maintain individual well-being. This abstract presents the development of a Body Mass Index (BMI) Calculator application utilizing Flutter framework and Android Studio.

The BMI Calculator app serves as a convenient tool for users to determine their BMI, a widely used indicator of body fatness. The app employs Flutter, a cross-platform UI toolkit, known for its efficiency in building visually appealing and responsive applications. Integrated with Android Studio, the development environment provides essential features for crafting robust Android applications.

Key features of the BMI Calculator app include user-friendly interface design, intuitive input mechanisms for height and weight, and real-time calculation of BMI based on provided metrics. Additionally, the app offers interpretation of BMI results, providing users with insights into their body composition and associated health risks.

The development process involved utilizing Flutter's widgets and layout structures to create a seamless user experience across various Android devices. The integration of mathematical algorithms for BMI calculation ensures accurate and reliable results. Furthermore, the app adheres to modern design principles and follows Material Design guidelines for consistency and visual appeal.

Overall, the BMI Calculator app serves as a valuable resource for individuals striving to monitor their health and make informed decisions regarding diet and exercise. By leveraging the capabilities of Flutter and Android Studio, the app delivers a polished and functional solution to the growing demand for mobile health applications.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	i
	LIST OF ABBREVIATIONS	iv
	LIST OF FIGURES	v
1	INTRODUCTION	1
1.1	About the Concept	1
1.2	Overview of App	2
1.3	Technologies Used	3
2	METHODOLOGIES	6
2.1	Architecture	7
2.2	Description about Quetelet App	9
2.3	Technology Stack	11
2.4	Project Scope	18
2.5	Benefits Of Quetelet Index	18
2.6	Difficulties In Quetelet Index	19
2.7	Development and Deployment Scope	21
2.8	Planned Features	27
2.9	Risk Analysis of the Project	28
2.10	Objectives	29
2.11	Use Case Diagram	31

2.12	Implementation Details and Testing Plan	32
2.13	Project Risk	36
2.14	Project Mitigation Strategies	36
2.15	Known Issues	36
2.16	Open-Source Libraries	37
3	RESULTS AND DISCUSSIONS	39
3.1	User interface Explanation	39
3.2	Quetelet Index Process	39
3.3	Interpretation of BMI Results	40
3.4	Quetelet Index App Result	40
4	CONCLUSION AND FUTURE WORK	43
4.1	Conclusion	43
4.2	Future Work	44
	REFERENCE	47
	APPENDICES	48
A	Sample Code	48
B	Screen Shots	52

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
API	- Application Programming Interface
DART	- Programming Language used with Flutter
FL	- Flutter
IDE	- Integrated Development Environment
iOS	- iPhone Operating System
JSON	- JavaScript Object Notation
MVVM	- Model-View-ViewModel
SaaS	- Software as a Service
SDK	- Software Development Kit
UAT	- User Acceptance Testing
UI	- User Interface
UX	- User Experience

LIST OF FIGURES

Figure No.	Name of the Figure	Page No.
2.1	Architecture	7
2.2	Working of Flutter	13
2.3	Working of Firebase	16
2.4	Flutter Doctor	21
2.5	Dependency upgrading	22
2.6	Getting SHA1 from gradlew	22
2.7	CLI Installation	23
2.8	a Firebase (google-services.json)	23
	b Firebase Configuration (Soruce Code)	24
2.9	Firebase Authentication	26
2.10	App Installation	26
2.11	Use Case Diagram	31
2.12	Logo Page	41
2.13	Login Page	41
2.14	Login	42
2.15	Home Page	42
2.16	Enter Weight & Height	42
2.17	Result Page	42

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE CONCEPT

In today's digital era, the significance of health and fitness has become increasingly apparent. With lifestyles becoming more sedentary and dietary habits evolving, there's a growing need for tools that facilitate monitoring and maintenance of individual well-being. One crucial aspect of health assessment is the measurement of Body Mass Index (BMI), a metric widely utilized to gauge body fatness and assess associated health risks. BMI serves as a fundamental indicator in determining whether an individual falls within a healthy weight range or if they are underweight, overweight, or obese.

The concept behind the BMI Calculator app arises from the necessity to provide individuals with a user-friendly and accessible tool to determine their BMI swiftly and accurately. By leveraging modern mobile technology, particularly the Flutter framework and Android Studio, this app aims to democratize access to BMI calculation, empowering users to take charge of their health with ease. Through intuitive interfaces and real-time calculations, the app offers users a seamless experience in understanding their BMI status, enabling informed decision-making regarding diet, exercise, and overall lifestyle choices.

In essence, the BMI Calculator app embodies the fusion of health awareness with technological innovation, bridging the gap between traditional health assessment methods and modern digital solutions. By harnessing the power of mobile computing, the app endeavours to promote proactive health management and foster a culture of wellness among individuals of all ages and backgrounds.

1.2 OVERVIEW OF APP

The Quetelet app represents a modern solution to the age-old task of assessing one's Body Mass Index (BMI), a critical metric in determining overall health and fitness. At its core, the app aims to provide users with a straightforward yet powerful tool to calculate their BMI quickly and accurately, thereby enabling them to gain valuable insights into their body composition and associated health risks.

Designed with user convenience in mind, the BMI Calculator app features an intuitive interface that guides users through the BMI calculation process seamlessly. Users simply input their height and weight into the app, and with a few taps, receive instant feedback on their BMI value. This real-time calculation feature eliminates the need for manual calculations or complex formulas, making the assessment process effortless and accessible to users of all backgrounds.

In addition to providing users with their BMI value, the app offers interpretation and categorization of BMI results into predefined ranges such as underweight, normal weight, overweight, and obese. This classification allows users to understand where they fall on the BMI spectrum and provides context for interpreting their results in relation to established health guidelines.

Furthermore, the BMI Calculator app goes beyond basic BMI calculation by offering additional features and functionalities to enhance the user experience. These include the ability to track BMI trends over time, personalized notes and insights, and cross-platform compatibility for seamless use across iOS and Android devices.

Overall, the BMI Calculator app serves as a comprehensive tool for individuals looking to monitor and manage their health effectively. By combining cutting-edge technology with the fundamental principles of health assessment, the app empowers users to make informed decisions about their lifestyle, diet, and exercise habits, ultimately promoting a healthier and

happier lifestyle.

1.3 TECHNOLOGIES USED

The development of the BMI Calculator app involved the utilization of various technologies and tools to create a robust and user-friendly mobile application. These technologies include:

1. Flutter Framework:

Flutter, developed by Google, is an open-source UI software development kit that enables the creation of natively compiled applications for mobile, web, and desktop from a single codebase. The BMI Calculator app leverages Flutter's extensive widget library and hot reload feature, allowing for rapid development and iteration of UI components. Additionally, Flutter's reactive framework facilitates the creation of smooth and responsive user interfaces across multiple platforms.

Flutter is an open-source UI software development kit created by Google. It's used to build natively compiled applications for mobile, web, and desktop from a single codebase. Flutter uses the Dart programming language and provides a rich set of pre-designed widgets for creating interactive and visually appealing user interfaces. One of Flutter's key strengths is its hot reload feature, allowing developers to see changes in real-time during development without losing the app's state. This makes the development process faster and more efficient. Flutter is known for its performance, flexibility, and ability to create beautiful cross-platform apps, making it a popular choice among developers for building modern applications.

2. Dart Programming Language:

Dart is the programming language used for developing Flutter applications. Dart is known for its simplicity, speed, and versatility, making it an ideal choice for mobile app development. The BMI Calculator app utilizes Dart's object-oriented programming features and syntax to implement business logic, handle user input, and perform BMI calculations efficiently.

Dart is a programming language developed by Google, primarily used for building web, mobile, and desktop applications. It's known for its simplicity, efficiency, and scalability, making it a popular choice for developers. Dart is strongly typed and supports object-oriented programming, making it easy to maintain and scale codebases. It also includes features like asynchronous programming, allowing for smooth handling of operations such as network requests and file I/O. With its growing ecosystem and compatibility with frameworks like Flutter, Dart continues to gain traction in the development community as a versatile and powerful language for modern app development.

3. Android Studio:

Android Studio is the official integrated development environment (IDE) for Android app development, providing developers with a comprehensive set of tools for building, testing, and debugging Android applications. In the case of the BMI Calculator app, Android Studio serves as the primary development environment for writing code, designing UI layouts, and deploying the app to Android devices. The IDE's rich set of features, including code completion, debugging tools, and device emulators, streamlines the app development process and enhances productivity.

4. GitHub:

GitHub is a web-based hosting service for version control using Git. It provides developers with a platform for collaborating on code, managing project repositories, and tracking changes over time. Throughout the development of the BMI Calculator app, GitHub was utilized for version control, allowing team members to synchronize code changes, review code contributions, and maintain a centralized repository of project files. Additionally, GitHub's issue tracking and project management features facilitated communication and coordination among team members, ensuring smooth progress throughout the development lifecycle.

5. Firebase:

- **Firebase Authentication:** Understanding how to integrate Firebase Authentication to enable sign-up and sign-in functionality using email/password, phone number, social media, etc.
- **Firebase Realtime Database / Firestore:** Learning how to store and sync data in real-time between clients and servers ² using Firebase's NoSQL cloud database solutions. Understanding concepts like collections, documents, queries, listeners, and security rules.
- **Firebase Cloud Firestore:** Exploring the newer Firestore database, which offers more advanced querying capabilities, scalability, and better offline support compared to the Realtime Database.
- **Firebase Cloud Messaging (FCM):** Implementing push notifications to engage users and keep them informed about relevant updates and events in the app.
- **Firebase Cloud Functions:** Leveraging serverless computing by deploying backend logic as functions triggered by Firebase events. This includes integrating with other Firebase services or external APIs.
- **Firebase Cloud Storage:** Learning how to store and serve user-generated content like images, videos, and other files using Firebase Cloud Storage.

CHAPTER 2

METHODOLOGY

The development of the BMI Calculator app will follow an iterative and agile methodology, allowing for continuous feedback and refinement throughout the development lifecycle.

The methodology will involve the following key steps:

- **Requirement Gathering:** Collaborate with stakeholders to gather and prioritize requirements for the app, including user stories, features, and functionalities.
- **Design:** Design the user interface (UI) and user experience (UX) of the app using wireframes, mock-ups, and prototypes to visualize the app's layout and navigation flow.
- **Development:** Implement the app's functionality using Flutter framework and Dart programming language, following best practices for code organization, documentation, and testing.
- **Testing:** Conduct comprehensive testing, including unit tests, integration tests, and user acceptance tests, to ensure the app meets quality standards and performs as expected on different devices and platforms.
- **Deployment:** Deploy the app to app stores (Google Play Store and Apple App Store) for distribution to users, following submission guidelines and app store policies.

To create a successful math matrix app, start by defining its purpose and target audience. Design a user-friendly interface with clear options for matrix operations. Implement efficient algorithms for operations like addition, multiplication, and inversion, while also ensuring error handling for invalid inputs. Optimize for speed and usability, visualize operations and mathematical concepts, and thoroughly test the app. Provide comprehensive documentation and gather user feedback for continuous improvement.

2.1 ARCHITECTURE

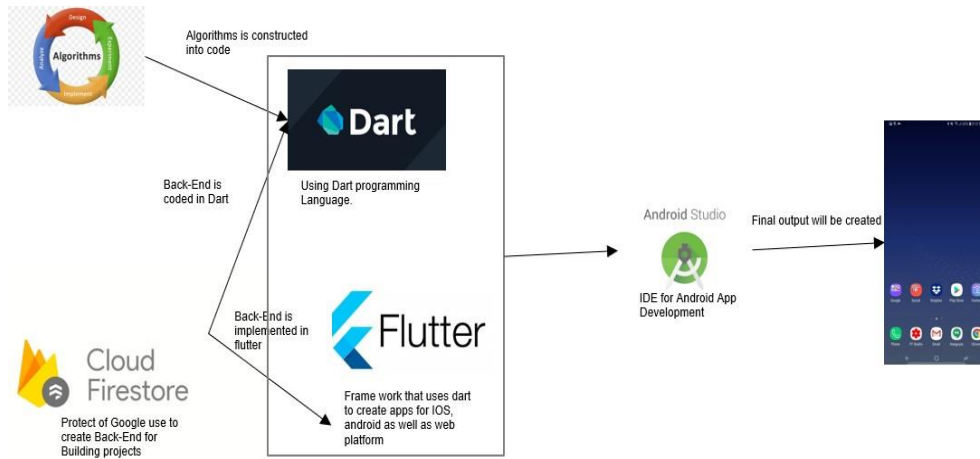


Fig 2.1: Architecture

The architecture of the BMI Calculator app will be based on a client-server model, where the client (mobile app) interacts with a backend server for BMI calculation and data storage (if necessary). The client-side architecture will follow a Model-View-Controller (MVC) or Model-View-ViewModel (MVVM) design pattern to separate concerns and ensure modularity and maintainability of the codebase.

Frontend (Client-Side):

UI Components: Design the login screen with input fields for username/email and password, along with buttons for login and registration.

Validation: Implement client-side validation for input fields (e.g., required fields, valid email format, password strength).

User Feedback: Provide feedback to users for successful login, incorrect credentials, network errors, etc., using notifications or dialogs.

Backend (Server-Side):

Authentication Service: Implement an authentication service on the server-side using technologies like Firebase Authentication, OAuth, or custom authentication logic.

User Management: Manage user accounts, including registration, login, logout, password reset, and account recovery functionalities.

Security: Ensure secure transmission of user credentials using HTTPS/SSL encryption and implement measures like rate limiting, CAPTCHA, and IP blocking to prevent brute-force attacks.

Database Integration: Integrate with a database (e.g., Firebase Firestore, MongoDB, MySQL) to store user credentials securely and manage user profiles and settings.

Communication (APIs):

RESTful APIs: Design and implement RESTful APIs for user authentication endpoints (e.g., /login, /register, /logout) to handle requests from the client-side.

Token-Based Authentication: Use token-based authentication (JWT, OAuth tokens) to manage user sessions securely and authorize access to protected resources.

API Documentation: Provide clear and comprehensive documentation for APIs, including endpoint URLs, request/response formats, authentication methods, and error handling.

State Management:

Client-Side State: Manage user authentication state (logged in/out, user data) using state management solutions like Provider, Bloc, Redux, or Firebase Auth state persistence.

Persistent Login: Implement remember me functionality or persistent login using secure cookies or device storage (e.g., SharedPreferences, Keychain) to keep users logged in across sessions.

Integration and Testing:

Integration Testing: Test the login process end-to-end, including UI interactions, API requests, authentication flows, error handling, and edge cases.

Security Testing: Conduct security testing (e.g., penetration testing,

vulnerability scanning) to identify and mitigate potential security risks and vulnerabilities.

Performance Testing: Ensure the login process is fast and responsive under various network conditions and user loads through performance testing and optimization.

Monitoring and Analytics:

Logging and Monitoring: Implement logging mechanisms to track user activities, API requests/responses, and system errors for troubleshooting and monitoring.

Analytics: Integrate analytics tools (e.g., Firebase Analytics, Google Analytics) to track user login patterns, user engagement, and app performance metrics.

2.2 DESCRIPTION ABOUT QUETELET INDEX

The Quetelet Index, often referred to as the Body Mass Index (BMI), is a measure used to assess an individual's body weight in relation to their height. It's a simple calculation that provides an estimate of body fatness and helps categorize individuals into different weight status categories: underweight, normal weight, overweight, or obese.

Here's how the BMI is calculated:

- Convert weight to kilograms (kg): If your weight is given in pounds (lbs), you can convert it to kilograms by dividing the weight in pounds by 2.20462.

For example, if your weight is 150 lbs:

$$\frac{150 \text{ lbs}}{2.20462} \sim 68.40 \text{ kg}$$

- Convert height to meters (m): If your height is given in inches (in), you can

convert it to meters by dividing the height in inches by 39.3701.

For example, if your height is 70 inches:

$$\frac{70in}{39.3701} \sim 1.778 m$$

- Calculate BMI: Once you have your weight in kilograms and height in meters, you can calculate the BMI using the formula:

$$BMI = \frac{weight(kg)}{height^2(m^2)}$$

Using the weight and height from the examples above:

$$BMI = \frac{68.04kg}{(1.78m)^2}$$

$$BMI = \frac{68.04kg}{3.1684m^2}$$

$$BMI \sim 21.48$$

Once you have calculated your BMI, you can interpret it using standard ranges:

- Underweight: BMI less than 18.5
- Normal weight: BMI between 18.5 and 24.9
- Overweight: BMI between 25 and 29.9
- Obese: BMI 30 or greater

It's important to note that while BMI is a useful screening tool, it does have limitations. For example, it doesn't differentiate between lean body mass and fat mass, nor does it account for factors such as muscle mass, bone density, and distribution of fat. Therefore, it's always best to interpret BMI alongside other measures of health and body composition.

2.3 TECHNOLOGY STACK

Dart programming Language

Dart is a programming language developed by Google, designed for building web, mobile, desktop, and server applications. It's known for its simplicity, speed, and versatility, making it suitable for a wide range of development tasks.

Here are some key aspects of Dart:

Syntax: Dart has C-style syntax with curly braces {}, semicolons; and uses a similar structure to languages like Java, C#, and JavaScript. It supports both object-oriented and functional programming paradigms, allowing developers to choose the approach that best fits their needs.

Strong and Static Typing: Dart is a statically typed language, meaning variable types are determined at compile time and type checking occurs before runtime. It also supports strong typing, which ensures type safety and reduces the chances of runtime errors related to type mismatches.

Just-in-Time (JIT) and Ahead-of-Time (AOT) Compilation: Dart uses a dual compilation approach: JIT compilation during development for fast iteration and hot reload capabilities, and AOT compilation for producing optimized, standalone executables. This compilation strategy provides a balance between developer productivity and application performance.

Garbage Collection: Dart includes automatic memory management through garbage collection, which helps manage memory allocation and deallocation, reducing the risk of memory leaks and manual memory management errors.

Concurrency and Asynchrony: Dart supports asynchronous programming using the `async` and `await` keywords, making it easy to work with asynchronous operations like network requests, file I/O, and timers. It also provides isolates, lightweight concurrent threads that run in their own memory

space, enabling concurrent programming without shared memory concerns.

Standard Library: Dart comes with a comprehensive standard library that includes collections (lists, maps, sets), I/O operations, date/time handling, mathematical functions, and more. Additional functionality can be accessed through packages from the Dart Package Repository (pub.dev), which hosts a wide range of third-party packages.

Cross-platform Development: Dart is used for developing cross-platform applications using frameworks like Flutter for mobile (iOS, Android), web (web apps, PWAs), and desktop (Windows, macOS, Linux). Its ability to target multiple platforms with a single codebase and UI framework (e.g., Flutter) makes it a popular choice for building modern, high-performance apps.

Tooling and Ecosystem: Dart is supported by a robust set of development tools, including the Dart SDK, Dart DevTools (for debugging and profiling), DartPad (online editor), and integration with popular IDEs like Android Studio, IntelliJ IDEA, and Visual Studio Code. The Dart ecosystem includes a growing community, documentation, tutorials, and resources for learning and building Dart applications.

Flutter Framework

Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It's known for its fast development workflow, expressive and flexible UI components, and high-performance rendering engine. Here's an overview of **Flutter:**

Cross-Platform Development: Flutter allows developers to create applications that run on multiple platforms with a single codebase. This includes iOS, Android, web (using Flutter for Web), and desktop (Windows, macOS, Linux).



Fig 2.2: Working of Flutter

Widget-Based UI Development: Flutter uses a reactive and declarative approach to UI development, where the entire UI is composed of widgets. Widgets are composable UI elements ranging from simple buttons and text fields to complex layouts and animations. Flutter's widget system enables building custom UI components, handling user input, managing state, and creating visually rich interfaces.

Hot Reload: One of Flutter's standout features is Hot Reload, which allows developers to instantly see changes made to the code reflected in the running app without restarting the entire application. This feature greatly speeds up the development process, enhances productivity, and facilitates rapid iteration and experimentation.

High-Performance Rendering Engine: Flutter uses its rendering engine called Skia to render UI elements directly on the canvas, providing consistent and smooth performance across different platforms. The Skia engine allows Flutter apps to achieve 60fps (frames per second) performance, resulting in fluid animations, transitions, and interactions.

Material Design and Cupertino Widgets: Flutter provides a rich set of UI components based on Material Design (Android) and Cupertino (iOS) design languages. Developers can create apps with native look and feel for each platform using these widgets. Material widgets offer features like navigation, theming, animations, and responsive layouts, while Cupertino widgets mimic iOS-specific UI elements and behaviours.

State Management: Flutter offers various approaches to state management, including `setState()` for managing local state within widgets, Inherited Widget and Scoped Model for sharing state across the widget tree, and third-party solutions like Provider, Riverpod, Bloc, and Redux. Developers can choose the state management solution that best fits the complexity and scalability requirements of their app.

Rich Ecosystem: Flutter has a vibrant ecosystem with a wide range of packages and plugins available on pub.dev, the official Flutter package repository. These packages provide additional functionality, integrations with APIs, platform-specific features, and UI enhancements. Flutter also integrates with Firebase for backend services (authentication, database, storage, analytics) and supports plugins for accessing native device features (camera, geolocation, sensors, etc.).

Development Tools: Flutter developers use tools like Android Studio, IntelliJ IDEA, and Visual Studio Code with Flutter extensions for coding, debugging, profiling, and managing Flutter projects. Flutter DevTools provides a suite of debugging and performance analysis tools, including widget inspector, timeline view, memory profiler, and network profiler.

- **Widgets:** Understanding the wide range of customizable widgets available in Flutter for building user interfaces.
- **Layouts:** Learning about different layout paradigms such as rows, columns, stacks, grids, etc., for arranging widgets on the screen.

- **Navigation:** Implementing navigation within the app using routes, navigation stacks, named routes, etc.
- **Animations:** Utilizing Flutter's powerful animation framework for creating smooth and engaging user experiences.
- **Package Management:** Learning to leverage packages from pub.dev, the official package repository for Dart and Flutter, to extend the functionality of the app.
- **Performance Optimization:** Optimizing the performance of Flutter apps by reducing rendering times, minimizing memory usage, and improving app startup times.
- **Testing:** Writing unit tests, widget tests, and integration tests to ensure the reliability and stability of the app.
- **Deployment:** Understanding the deployment process for deploying Flutter apps to various platforms like Android, iOS, web, and desktop.

Firebase (Back – End)

Firebase is a comprehensive platform developed by Google that provides a wide range of tools and services to help developers build and scale high-quality apps across various platforms. It offers backend infrastructure, real-time database, authentication, analytics, cloud storage, and more. Here's an overview of Firebase and its key features:

Real-Time Database: Firebase Realtime Database is a NoSQL cloud database that allows developers to store and sync data in real-time across clients (web, iOS, Android). It uses JSON-like data structure and WebSocket protocol for real-time updates, making it suitable for applications requiring live synchronization, such as chat apps, collaboration tools, and multiplayer games.

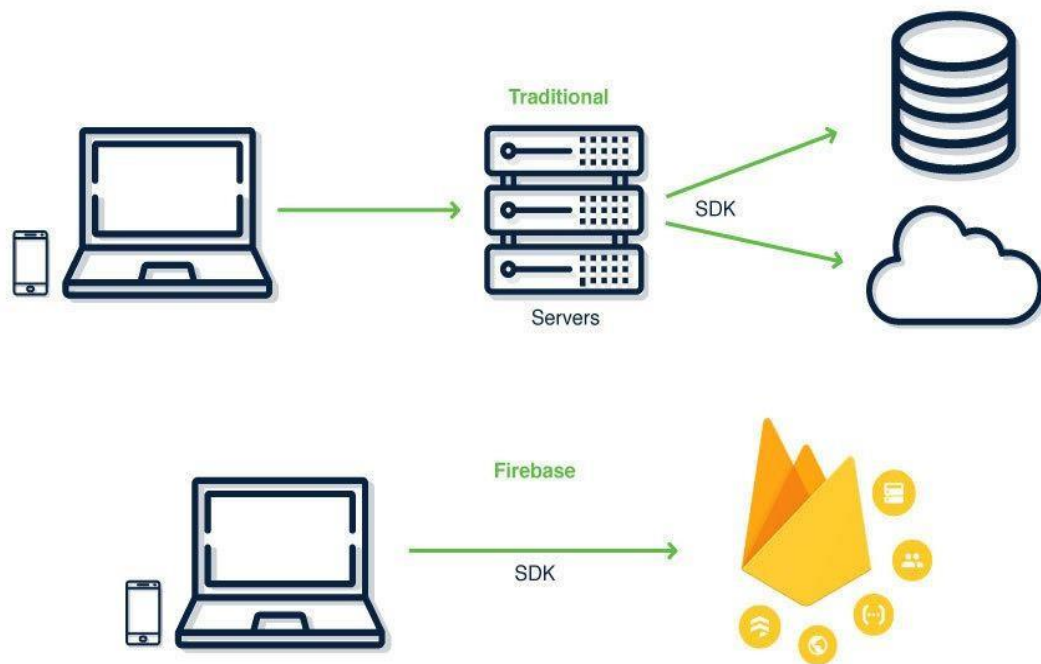


Fig 2.3: Working of Firebase

Authentication: Firebase Authentication provides easy-to-use SDKs and ready-to-use UI libraries for user authentication. It supports various authentication methods including email/password, phone number, social login (Google, Facebook, Twitter, etc.), and anonymous authentication. Firebase Auth also offers features like user management, custom claims, multi-factor authentication (MFA), and OAuth integration.

Cloud Firestore: Firebase Cloud Firestore is a flexible, scalable NoSQL document database designed for mobile, web, and server-side development. It offers powerful querying, offline data support, real-time updates, and seamless integration with other Firebase services. Firestore is ideal for apps that require complex data structures, hierarchical data modeling, and efficient data synchronization.

Cloud Storage: Firebase Storage provides secure cloud storage for user-generated content such as images, videos, audio files, and documents. Developers can easily upload, download, and manage files using Firebase

Storage SDKs and APIs. It offers features like file metadata, access control rules, resumable uploads, and integration with Firebase Authentication for secure file access.

Hosting: Firebase Hosting allows developers to deploy and host web apps, static content, and dynamic content (Cloud Functions) with HTTPS support. It offers features like SSL encryption, CDN integration, custom domains, automatic scaling, and easy deployment using Firebase CLI.

Cloud Functions: Firebase Cloud Functions enables developers to extend app functionality with serverless backend code written in JavaScript, TypeScript, or Node.js. Functions can respond to events triggered by Firebase services (e.g., database changes, authentication events) or HTTP requests (API endpoints). Developers can use Cloud Functions for tasks like data processing, notifications, authentication triggers, and third-party integrations.

Analytics: Firebase Analytics provides insights into app usage, user engagement, conversion tracking, and performance metrics. It offers event tracking, user properties, audience segmentation, funnels, user flows, and integration with Google Analytics for Firebase. Developers can use analytics data to understand user behaviour, optimize app features, and make data-driven decisions.

Other Features: Firebase also includes services like Remote Config for dynamic app configuration, Cloud Messaging (FCM) for push notifications, Firebase Performance Monitoring, Firebase Test Lab for app testing, Firebase App Indexing for app discoverability, Firebase Crashlytics for crash reporting, and more.

Firebase's ease of use, scalability, real-time capabilities, cross-platform support, and integration with Google services make it a popular choice among developers for building modern and feature-rich apps with a robust backend

infrastructure.

2.4 PROJECT SCOPE

The scope of the BMI Calculator app project includes the development of core features such as real-time BMI calculation, result interpretation, BMI trend tracking, and cross-platform compatibility. Additional features and functionalities may be included based on available resources and project constraints.

2.5 BENEFITS OF QUETELET INDEX

The Quetelet Index, commonly known as the Body Mass Index (BMI), offers several benefits similar to those of BMI calculators:

1. **Simple Assessment:** Like BMI, the Quetelet Index provides a simple and quick method for assessing body weight relative to height, making it accessible to individuals without the need for specialized equipment.
2. **Health Awareness:** Calculating the Quetelet Index raises awareness about one's body weight and its potential impact on health, helping individuals understand their risk for weight-related health conditions such as diabetes, heart disease, and hypertension.
3. **Health Monitoring:** Regularly measuring the Quetelet Index allows individuals to monitor changes in their weight over time, enabling them to track progress towards achieving and maintaining a healthy weight.
4. **Risk Assessment:** Similar to BMI, the Quetelet Index helps assess the risk of obesity-related health conditions, allowing individuals to take proactive measures to mitigate these risks through lifestyle changes and interventions.
5. **Goal Setting:** By understanding their Quetelet Index and the healthy range for their height, individuals can set realistic weight management goals, whether it involves losing weight, gaining weight, or maintaining current weight, thereby promoting overall health and well-being.

6. **Clinical Use:** Healthcare professionals use the Quetelet Index as a screening tool to assess patients' weight status and identify those at risk for obesity-related health problems, facilitating early intervention and personalized treatment plans.

7. **Public Health Planning:** Population-level data on the Quetelet Index is valuable for public health planning and policy development, enabling policymakers to design targeted interventions aimed at reducing obesity rates and promoting healthy lifestyles.

Overall, the Quetelet Index, like BMI, serves as a valuable tool for individuals, healthcare providers, and public health organizations in promoting healthy weight management and reducing the burden of obesity-related diseases.

2.6 DIFFICULTIES IN QUETELET INDEX

While the Quetelet Index, commonly known as BMI, is a widely used tool for assessing body weight relative to height, it does have some limitations and difficulties:

1. **Doesn't Account for Body Composition:** BMI does not differentiate between fat mass and lean body mass. Therefore, individuals with high muscle mass may have a high BMI, indicating overweight or obesity, even though they have a low body fat percentage.

2. **Variations Across Populations:** BMI may not accurately reflect body composition in certain populations, such as athletes or individuals from different ethnic backgrounds. These populations may have different body compositions than those for whom the BMI formula was originally developed.

3. **Age and Gender Differences:** BMI may not be equally applicable across different age groups and genders. For example, older adults may have different body composition and distribution of muscle and fat compared to

younger adults, which can affect the interpretation of BMI values.

4. **Pregnancy:** BMI may not be suitable for assessing weight status during pregnancy due to changes in body composition and distribution of weight associated with fetal development.

5. **Misclassification:** BMI can misclassify individuals, leading to false positives or false negatives regarding weight status and associated health risks. For example, individuals with a normal BMI may still have excess body fat and be at risk for obesity-related health conditions.

6. **Health Risks in Normal Weight Individuals:** Some individuals classified as having a normal BMI may still be at risk for obesity-related health conditions due to factors such as central adiposity (abdominal fat deposition) or metabolic abnormalities.

7. **Not a Direct Measure of Health:** BMI is a measure of body weight relative to height and does not directly assess health or fitness levels. Therefore, individuals with a healthy BMI may still have poor health habits or be at risk for certain health conditions.

Despite these difficulties, BMI remains a valuable screening tool for assessing weight status and identifying individuals at risk for obesity-related health conditions. However, it should be used in conjunction with other measures and clinical assessments to provide a more comprehensive evaluation of an individual's health status.

2.7 DEVELOPMENT AND DEPLOYMENT SETUP

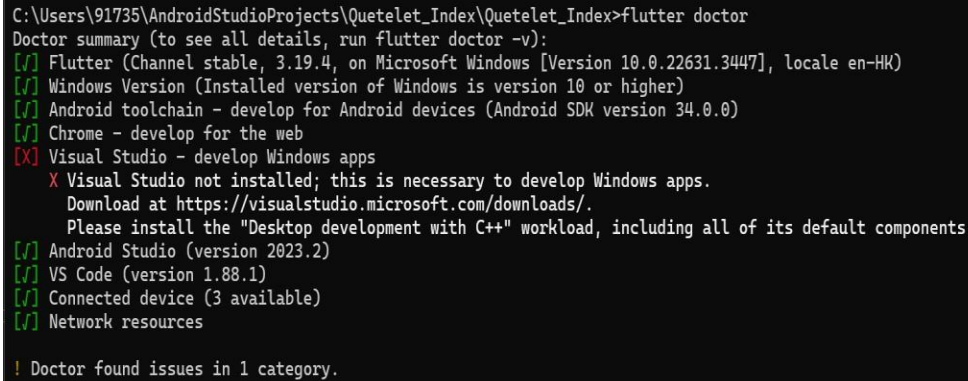
1. Development Environment:

Flutter and Dart need to be installed on the developer's machine. IDEs like Visual Studio Code or Android Studio for coding.

2. Create Project:

```
>> flutter doctor
```

```
>> flutter create project_name
```



```
C:\Users\91735\AndroidStudioProjects\Quetelet_Index\Quetelet_Index>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.19.4, on Microsoft Windows [Version 10.0.22631.3447], locale en-HK)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[✓] Android Studio (version 2023.2)
[✓] VS Code (version 1.88.1)
[✓] Connected device (3 available)
[✓] Network resources

! Doctor found issues in 1 category.
```

Fig 2.4: Flutter Doctor

3. Dependency Management:

Project dependencies specified in the pubspec.yaml file. Key packages include:

- `firebase_core` for Firebase integration.
- `firebase_auth` for user authentication.
- `cloud_firestore` for real-time data synchronization.
- Other packages like `get`, `image_picker`, etc., for additional functionalities.

Following are some of the commands used to debug packages

```
>> flutter pub get
```

```
>> flutter pub add firebase_core
```

```
>> flutter pub add firebase_auth
```

```
>> flutterfire configure
```

```

/ actionable tasks. / executed
PS C:\Users\91735\AndroidStudioProjects\Quetelet_Index\Quetelet_Index\android> flutter pub get
Resolving dependencies in ..... (3.7s)
 _fe_analyzer_shared 64.0.0 (68.0.0 available)
 _flutterfire_internals 1.3.25 (1.3.32 available)
 analyzer 6.2.0 (6.5.0 available)
 archive 3.4.10 (3.5.0 available)
 args 2.4.2 (2.5.0 available)
 cupertino_icons 1.0.6 (1.0.8 available)
 firebase_auth 4.17.8 (4.19.4 available)
 firebase_auth_platform_interface 7.1.8 (7.2.5 available)
 firebase_auth_web 5.9.8 (5.11.4 available)
 firebase_core 2.27.0 (2.30.1 available)
 firebase_core_web 2.11.5 (2.16.0 available)
 frontend_server_client 3.2.0 (4.0.0 available)
 vm_service 13.0.0 (14.2.1 available)
 web 0.3.0 (0.5.1 available)
 web_socket_channel 2.4.0 (2.4.5 available)

```

Fig 2.5: Dependency upgrading

4. Firebase Configuration:

Firebase project created and configured for the application. Firebase Hosting used for deploying the Flutter web application.

```

# getting SHA1 code used to link with firebase
>> ./gradlew signingReport

```

```

PS C:\Users\91735\AndroidStudioProjects\Quetelet_Index> cd Quetelet_Index
PS C:\Users\91735\AndroidStudioProjects\Quetelet_Index\Quetelet_Index> cd android
PS C:\Users\91735\AndroidStudioProjects\Quetelet_Index\Quetelet_Index\android> ./gradlew signingReport
Starting a Gradle Daemon, 2 busy and 3 incompatible Daemons could not be reused, use --status for details

> Task :app:signingReport
Variant: debug
Config: debug
Store: C:\Users\91735\.android\debug.keystore
Alias: AndroidDebugKey
MD5: C2:15:CF:53:B9:75:E0:4C:C1:09:E2:6A:57:14:DF:F7
SHA1: 86:73:8F:4E:42:F4:C6:AF:E3:C7:6D:2E:DC:EA:C8:06:B1:E0:78:96
SHA-256: 63:9F:76:77:EB:5A:CE:4B:BF:E4:EA:72:19:D8:AB:A3:65:84:04:29:75:8C:87:83:D2:1B:58:73:06:CF:D2:38
Valid until: Saturday, 14 June 2053
-----
Variant: release
Config: debug
Store: C:\Users\91735\.android\debug.keystore
Alias: AndroidDebugKey
MD5: C2:15:CF:53:B9:75:E0:4C:C1:09:E2:6A:57:14:DF:F7

```

Fig: 2.6: Getting SHA1 from gradlew

```

# Install the CLI if not already done so
>> dart pub global activate flutterfire_cli

```

```

A Dart VM Service on sdk gphone64 x86_64 is available at: http://127.0.0.1:62326/1SsZsIVkD-w=/
dart_console 1.2.0 (4.1.0 available)
> deep_pick 1.0.0 (was 0.10.0)
  file 6.1.4 (7.0.0 available)
> flutterfire_cli 1.0.0 (was 0.2.7)
  http 0.13.6 (1.2.1 available)
  intl 0.18.1 (0.19.0 available)
> json_annotation 4.9.0 (was 4.8.1)
> meta 1.14.0 (was 1.12.0)
  process 4.2.4 (5.0.2 available)
  pub_updater 0.2.4 (0.4.0 available)
> test_api 0.7.1 (was 0.7.0)
> win32 5.5.0 (was 5.4.0)
Building package executables... (12.1s)

```

Fig: 2.7: CLI Installation

```

#node.js install
>> npm install -g firebase-tools
# Run the `configure` command, select a Firebase project
and platforms
>> flutterfire configure

```

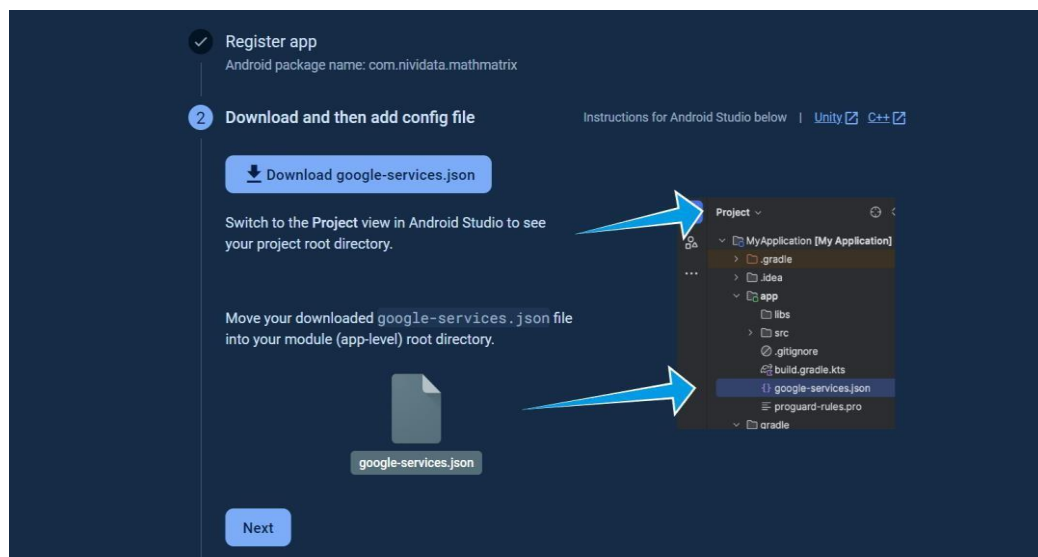


Fig: 2.8(a): Firebase (google-services.json)



Fig: 2.8(b): Firebase Configuration (Source Code)

5. Firebase Authentication:

What is Firebase Authentication?

Firebase Authentication is a service provided by Google Firebase that allows you to authenticate users to your app using different methods like email/password, phone number, Google, Facebook, etc.

Key Features:

- Easy integration: Firebase Authentication provides SDKs and libraries for various platforms like Android, iOS, and web, making it easy to integrate authentication into your app.
- Secure authentication methods: Supports secure authentication methods like email/password, phone number, and third-party providers like Google, Facebook, Twitter, etc.
- Custom authentication system: You can also implement custom authentication systems using Firebase Authentication.
- User management: Firebase provides tools for managing users, including user creation, deletion, profile updates, etc.
- Authentication state monitoring: You can monitor the authentication state of users in real-time to customize the app experience based on user

authentication status.

Authentication Methods:

- Email/Password: Users can sign up and sign in using their email address and password.
- Phone Number: Users can authenticate using their phone numbers with SMS verification.
- Third-party providers: Firebase Authentication supports popular third-party providers like Google, Facebook, Twitter, GitHub, etc.

Implementation Steps:

- Add Firebase to your project: Start by adding Firebase to your project and configuring it with your app.
- Initialize Firebase Auth: Initialize Firebase Authentication in your app to start using its features.
- Implement authentication UI: Create UI components for user registration, login, password reset, etc., using Firebase Authentication SDK.
- Handle authentication events: Monitor authentication state changes to update the UI and app behavior accordingly.
- Secure your app: Follow best practices for securing user authentication, like using secure connections (HTTPS), enforcing strong password policies, etc.

User Management:

- Firebase Console: Manage users through the Firebase console, where you can view user details, manually add or remove users, etc.
- SDK/API: Use Firebase SDKs or APIs to perform user management tasks programmatically within your app.

Security Considerations:

- Use HTTPS: Always use HTTPS to secure data transmitted between your app and Firebase servers.
- Authentication rules: Configure Firebase Authentication rules to control

access to Firebase services based on user authentication status and data validation.

- Verify ID tokens: When interacting with Firebase services, verify ID tokens to ensure the requests are coming from authenticated users.

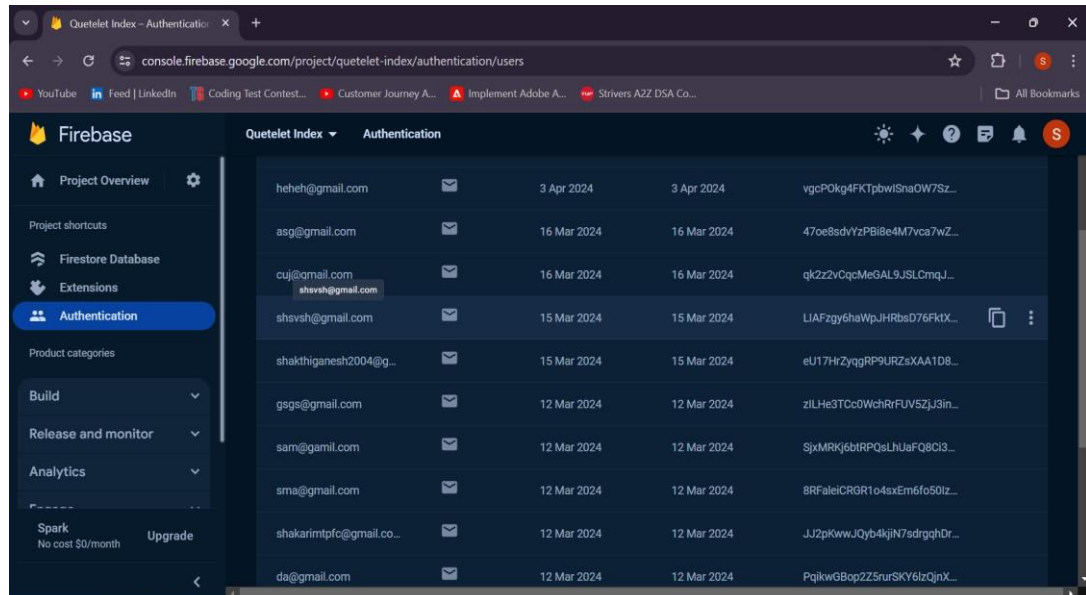


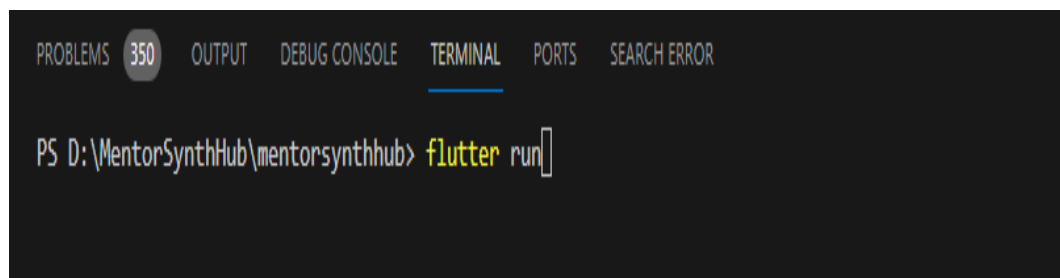
Fig 2.9: Firebase Authentication

6. Development Tools:

Flutter commands used for testing and building the application. Real-time testing on Android and iOS devices.

```
>> firebase login
>> flutter run
```

Fig 2.10: App Installation



7. Deployment Process:

Firebase used for hosting the web version of the application. Seamless deployment using Flutter and Firebase commands.

8. Supported Platform:

Cross-platform support for Android, iOS, and web.

9. Integrated Development Environment (IDE):

Developers can choose between Visual Studio Code or AndroidStudio for coding

2.8 PLANNED FEATURES

1. Enhanced User Profiles:

Develop user profiles to allow users to save their BMI calculations and track their progress over time. Include features such as profile customization and the ability to set health goals.

2. Integration with Health APIs:

Integrate with health and fitness APIs to provide additional data insights and personalized recommendations based on users' BMI and health goals. Include features such as exercise recommendations, nutritional guidance, and fitness tracking.

3. Customizable Themes:

Implement customizable themes and colour schemes to allow users to personalize the app's appearance according to their preferences.

4. Accessibility Features:

Enhance accessibility by adding features such as text-to-speech, voice commands, and support for screen readers to make the app more inclusive for users with disabilities.

2.9 RISK ANALYSIS OF THE PROJECT

Risk analysis is a crucial aspect of project management that involves identifying potential risks and developing strategies to mitigate or address them effectively.

In the context of the BMI Calculator app project, the following risks have been identified:

1. Technical Risks:

- Risk: Difficulty in integrating complex mathematical algorithms for BMI calculation into the app.
- Mitigation: Conduct thorough research and testing of available algorithms, consult with domain experts if necessary, and allocate sufficient time and resources for implementation and testing.

2. Resource Risks:

- Risk: Insufficient resources (time, budget, personnel) to complete the project within the specified timeline.
- Mitigation: Develop a detailed project plan with realistic timelines and milestones, regularly monitor progress against the plan, and allocate resources efficiently to meet project deadlines.

3. Scope Creep:

- Risk: Changes in project requirements or scope during development, leading to delays and additional costs.
- Mitigation: Establish clear project requirements and scope upfront, document any changes or updates to requirements, and involve stakeholders in decision-making to ensure alignment with project objectives.

4. Compatibility Issues:

- Risk: Compatibility issues arising from differences in device configurations, screen sizes, and operating systems.
- Mitigation: Conduct thorough testing on a variety of devices and platforms, use responsive design principles to ensure compatibility

across different screen sizes, and implement fallback options for unsupported features or devices.

2.10 OBJECTIVES

The proposed BMI Calculator app aims to achieve the following Objectives:

- Provides users with user-friendly and intuitive tool for calculating their Body Mass Index (BMI) quickly and accurately.
- Offer real-time BMI calculation based on user-provided height and weight inputs.
- Interpret BMI results and categorize them into standard BMI ranges to provide users with insights into their health status.
- Enable users to track BMI trends over time and set personalized health goals for maintaining a healthy lifestyle.
- Ensure cross-platform compatibility to allow users to access the app seamlessly on both iOS and Android devices.

The development of the BMI Calculator app will follow an iterative and agile methodology, allowing for continuous feedback and refinement throughout the development lifecycle.

The methodology will involve the following key steps:

- Requirement Gathering: Collaborate with stakeholders to gather and prioritize requirements for the app, including user stories, features, and functionalities.
- Design: Design the user interface (UI) and user experience (UX) of the app using wireframes, mockups, and prototypes to visualize the app's layout and navigation flow.
- Development: Implement the app's functionality using Flutter framework and Dart programming language, following best practices for code organization, documentation, and testing.
- Testing: Conduct comprehensive testing, including unit tests, integration tests, and user acceptance tests, to ensure the app meets quality

standards and performs as expected on different devices and platforms.

- Deployment: Deploy the app-to-app stores (Google Play Store and Apple App Store) for distribution to users, following submission guidelines and app store policies.

Key Challenges

Accuracy of Calculations: Ensuring accurate BMI calculations is crucial. The formula for BMI is weight in kilograms divided by height in meters squared. Handling conversions and rounding errors can affect the accuracy of results.

Data Privacy: BMI data is sensitive health information. Ensuring compliance with data privacy regulations like GDPR or HIPAA (if applicable) requires robust security measures for data storage, transmission, and user consent management.

User Input Validation: Validating user input for weight, height, and other parameters is essential to prevent incorrect data entry that can lead to inaccurate BMI calculations.

User Experience: Designing a user-friendly interface that guides users through entering their data accurately and interpreting the BMI results effectively is crucial for usability and adoption.

Customization: Users may have specific requirements or preferences regarding BMI calculations (e.g., different units of measurement, age-specific norms). Providing customization options while maintaining accuracy adds complexity to the app.

Interpretation and Context: BMI alone may not provide a complete picture of an individual's health. Providing additional information, context, and interpretation of BMI results (e.g., health risks, recommended actions) requires expertise and clear communication.

Integration with Health Data: Integrating the app with health data from wearable devices, electronic health records (EHRs), or other sources can enhance the app's functionality but requires compatibility, data security, and user consent considerations.

Updates and Maintenance: Keeping the app up-to-date with the latest BMI guidelines, regulatory requirements, and technological advancements (e.g., new devices, operating system updates) requires ongoing maintenance and development efforts.

2.11 USE CASE DIAGRAM

The use case diagram for the BMI Calculator app will illustrate the various interactions between actors (users) and the system (app) to accomplish different tasks, such as BMI calculation, result interpretation, and trend tracking. It will serve as a visual representation of the app's functionality and help identify user requirements and system boundaries.

Explanation:

User: Represents the user of the BMI Calculator app.

Input Height: Use case where the user enters their height.

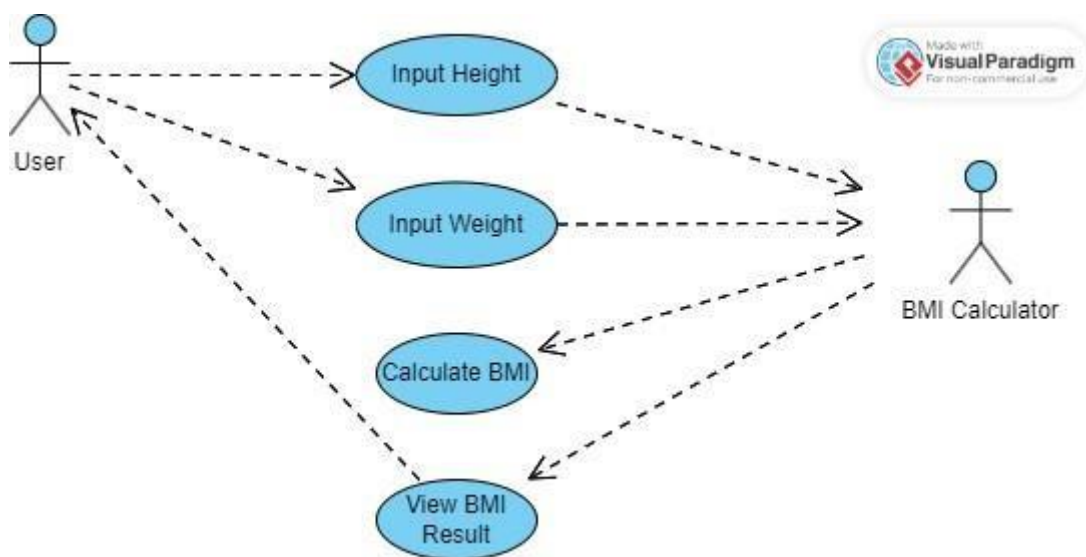


Fig 2.11: Use Case Diagram

Input Weight: Use case where the user enters their weight.

Calculate BMI: Use case where the user initiates the BMI calculation process.

View BMI Result: Use case where the user views the calculated BMI result.

BMI Calculator: Represents the system itself.

Input Height: Use case where the system receives and validates the user's height input.

Input Weight: Use case where the system receives and validates the user's weight input.

Calculate BMI: Use case where the system computes the BMI based on the validated height and weight inputs.

View BMI Result: Use case where the system displays the calculated BMI result along with interpretation.

2.12 IMPLEMENTATION DETAILS AND TESTING PLAN

Syntax Used

The implementation of the BMI Calculator app will utilize the Dart programming language syntax, which is the primary language for developing Flutter applications. The following are some examples of the syntax used in the app's implementation:

1. Variable Declaration and Initialization:

```
double height = 1.75;  
double weight = 70.0;
```

2. Conditional Statement (if - else):

```
if (bmi < 18.5) {  
    category = "Underweight";  
} else if (bmi >= 18.5 && bmi < 25) {  
    category = "Normal";  
} else if (bmi >= 25 && bmi < 30) {
```

```

        category = "Overweight";
    } else {
        category = "Obese";
    }

```

3. Functions:

```

double calculateBMI(double height, double
weight) {
    return weight / (height * height);
}

```

4. Lists:

```

List<String> bmiCategories = ["Underweight",
"Normal", "Overweight", "Obese"];

```

5. Widgets (UI Components):

```

TextField(
    controller: heightController,
    keyboardType: TextInputType.number,
    decoration: InputDecoration(
        labelText: 'Height (m)',
        border: OutlineInputBorder(),
    ),
)

```

6. Asynchronous Programming (Futures and async/await):

```

Future<void> calculateBMI() async {
    // Perform BMI calculation asynchronously
    await Future.delayed(Duration(seconds: 1));
    double bmi = weight / (height * height);
    setState(() {
        _bmi = bmi;
    });
}

```

```
}
```

7. Classes and Object-Oriented Concepts:

```
class BMIResult {  
    String category;  
    double value;  
  
    BMIResult(this.category, this.value);  
}
```

8. Error Handling (try-catch):

```
try {  
    double bmi = calculateBMI(height, weight);  
    print('BMI: $bmi');  
} catch (e) {  
    print('Error calculating BMI: $e');  
}
```

Assets Utilized

In addition to code implementation, the BMI Calculator app will utilize various assets to enhance its visual appeal and user experience. One crucial asset that will be utilized is custom fonts. Custom fonts can significantly impact the aesthetics and readability of the app's text elements. The following details the assets utilized, particularly the custom font:

1. Custom Fonts:

The BMI Calculator app will incorporate custom fonts to enhance the typography of its user interface. Custom fonts can help establish a unique visual identity for the app and improve readability across different screen sizes and resolutions.

Example of integrating a custom font in Flutter:

```
Text(  
  'BMI Calculator',  
  style: TextStyle(  
    fontFamily: 'Roboto', // Custom font family  
    fontWeight: FontWeight.bold,  
    fontSize: 24.0,  
  ),  
)
```

The 'Roboto' font family is commonly used in Flutter applications due to its versatility and readability. However, the app may utilize other custom font families based on design requirements and preferences.

2. Image Assets:

The app may also incorporate image assets, such as icon and graphics, to complement its user interface and provide visual cues for different actions and features.

Example of using image assets in Flutter:

```
Image.asset('assets/icons/bmi_icon.png', width: 24,  
height: 24)
```

3. Additional assets:

Other assets, such as placeholder images, splash screens, and background patterns, may be utilized to further enhance the app's visual appeal and usability.

Example of using placeholder images:

```
Placeholder(  

```

```
        fallbackWidth: 100,  
        fallbackHeight: 100,  
    )
```

By incorporating custom fonts and other assets, the BMI Calculator app will deliver a visually appealing and engaging user experience, ensuring that users can interact with the app comfortably and effectively. Additionally, these assets will contribute to establishing a cohesive and professional brand identity for the app.

2.13 PROJECT RISKS

Potential risks associated with the project include technical challenges in BMI calculation implementation, resource constraints, scope creep, compatibility issues, and data security risks. These risks will be identified, assessed, and managed throughout the project lifecycle to minimize their impact on project success.

2.14 PROJECT MITIGATION STRATEGIES

To mitigate project risks, proactive measures will be taken, including thorough research and testing of BMI calculation algorithms, effective resource management, scope management, rigorous testing and quality assurance processes, and implementation of robust security measures for data protection and privacy. Regular risk assessments and contingency planning will be conducted to address any emerging risks promptly.

2.15 KNOWN ISSUES

1. Input Validation:

Improve input validation to ensure that users provide accurate height and weight measurements for BMI calculation. Address issues related to invalid inputs or edge cases that may lead to incorrect results.

2. Performance Optimization:

Optimize app performance to reduce load times and improve responsiveness,

especially when handling large datasets or complex calculations. Identify and resolve bottlenecks in code execution and rendering to enhance overall performance.

3. Compatibility Issues:

Resolve compatibility issues with certain devices or screen sizes to ensure a consistent user experience across different platforms and devices. Test the app on a variety of devices and screen resolutions to identify and address any layout or rendering issues.

2.16 OPEN-SOURCE LIBRARIES

For the BMI Calculator app, you can consider integrating the following open-source libraries to enhance its functionality and user experience:

1. `flutter_screenutil`: This library allows you to create adaptive UIs that scale to different screen sizes and resolutions. It simplifies the process of building responsive layouts, ensuring a consistent user experience across various devices.
2. `charts_flutter`: If you plan to include visualizations of BMI trends and statistics, the `charts_flutter` library provides a variety of chart types, including line charts, bar charts, and pie charts. It enables you to display data in a visually appealing and informative way.
3. `intl`: The `intl` package provides internationalization and localization utilities for Flutter apps. It allows you to format dates, numbers, and currencies according to different locales and languages, making your app accessible to users worldwide.
4. `flutter_svg`: If you want to include vector graphics in your app, the `flutter_svg` library allows you to render SVG files directly within your Flutter application. It supports advanced SVG features such as gradients, transformations, and masking.

5. `flutter_spinkit`: This library provides a collection of loading indicators and spinners that you can use to indicate progress or activity within your app. It offers a variety of customizable loading animations to choose from, enhancing the user experience during data processing or loading screens.
6. `shared_preferences`: The `shared_preferences` package enables you to store simple data types persistently on the device. It allows you to save user preferences, settings, or cached data locally, providing a seamless experience across app sessions.
7. `http`: If your app requires fetching data from external APIs or services, the `http` package offers convenient methods for making HTTP requests and handling responses. It supports various HTTP methods and provides features for authentication, error handling, and data serialization.

CHAPTER 3

RESULTS AND DISCUSSIONS

3.1 USER INTERFACE EXPLANATION

The user interface (UI) of the BMI Calculator app has been meticulously designed to prioritize user experience and ease of use. The UI features a minimalist and intuitive design, with clear and concise input fields for height and weight. Users can easily input their measurements and receive instant feedback on their BMI calculation.

Key aspects of the UI design include:

- Clean and clutter-free layout to avoid distractions and maintain focus on essential elements.
- Large, legible text for improved readability, especially on smaller screens.
- Intuitive input fields with appropriate labels and placeholders for height and weight.
- Responsive design that adapts seamlessly to different screen sizes and orientations.

Overall, the UI design of the BMI Calculator app aims to provide users with a seamless and intuitive experience, making it easy for them to calculate their BMI and gain insights into their health status.

3.2 QUETELET INDEX PROCESS

The BMI calculation process in the app is straightforward and efficient. Upon entering their height and weight measurements, users can initiate the BMI calculation with a simple tap of a button. The app instantly computes the BMI value based on the provided inputs and displays the result to the user.

The BMI calculation process involves the following steps:

- User inputs their height and weight values into the designated fields.
- Upon tapping the "Calculate BMI" button, the app retrieves the input values and computes the BMI using the standard formula: $BMI = \text{weight} /$

(height * height).

- The calculated BMI value is then displayed to the user, providing them with immediate feedback on their body fatness.

This streamlined process ensures that users can obtain their BMI results quickly and conveniently, empowering them to monitor their health status effectively.

3.3 INTERPRETATION OF BMI RESULTS

After calculating the BMI, the app provides users with an interpretation of the results, categorizing their BMI value into standard BMI ranges. This interpretation helps users understand where their BMI falls within the spectrum of body fatness and provides insights into their overall health status.

The interpretation of BMI results includes the following categories:

- Underweight: $\text{BMI} < 18.5$
- Normal: $18.5 \leq \text{BMI} < 25$
- Overweight: $25 \leq \text{BMI} < 30$
- Obese: $\text{BMI} \geq 30$

By categorizing BMI results, the app helps users assess their body composition and identify potential health risks associated with being underweight, overweight, or obese. This information empowers users to make informed decisions about their diet, exercise, and overall lifestyle to maintain or improve their health.

3.4 QUETELET INDEX APP RESULT

The app provides users with immediate access to their Body Mass Index (BMI) result, which categorizes their weight status into different ranges such as underweight, normal weight, overweight, and obese. This instant feedback empowers users to assess their health status quickly and take proactive measures if needed. Additionally, the app's clear and concise presentation of the BMI result facilitates better understanding and encourages users to make

informed decisions regarding their diet, exercise, and overall well-being.



Fig 2.12: Logo Page



Fig 2.13: Login Page



Fig 2.14: Login



Fig 2.15: Home Page

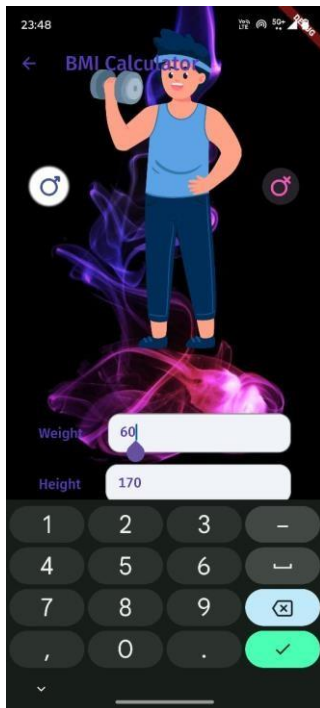


Fig 2.16: Enter Weight & Height

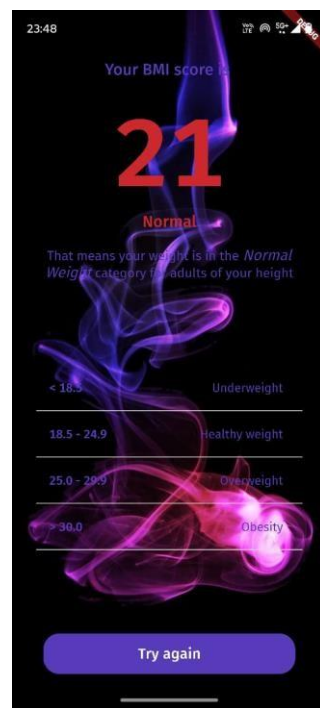


Fig 2.17: Result Page

CHAPTER 4

CONCLUSION AND FUTURE WORK

In conclusion, the BMI Calculator app developed using Flutter and Dart provides users with a convenient and effective tool for monitoring their body mass index (BMI) and gaining insights into their health status. The app's intuitive user interface, real-time BMI calculation, and interpretation of results empower users to make informed decisions about their health and wellness.

Throughout the development process, careful attention was paid to user experience and functionality, resulting in a streamlined and efficient app that meets the needs of its users. By leveraging the capabilities of Flutter and Dart, the BMI Calculator app sets a high standard for mobile health applications, contributing to the promotion of well-being and fitness in the digital age.

Moving forward, continuous updates and enhancements to the app can further improve its functionality and usability, ensuring that it remains a valuable resource for individuals striving to maintain a healthy lifestyle. With its user-friendly design and robust features, the BMI Calculator app is poised to make a positive impact on the health and wellness of its users.

4.1 CONCLUSION

The Quetelet Index app has been developed successfully to calculate and interpret the Body Mass Index (BMI) of users. Through this project, several key accomplishments have been achieved:

Functional BMI Calculation: The app accurately calculates the BMI based on user input of height and weight, providing an immediate result that indicates the user's BMI category.

User-Friendly Interface: The app features a user-friendly interface with intuitive input fields and clear display of results, making it easy for users to understand and utilize.

Educational Component: The app includes educational content about BMI, its significance, and guidelines for maintaining a healthy weight, enhancing user awareness and promoting healthier lifestyle choices.

Platform Compatibility: The app is compatible with various mobile platforms, ensuring accessibility to a wide range of users.

Error Handling: Robust error handling mechanisms have been implemented to handle invalid inputs and unexpected scenarios, ensuring the app's reliability and stability.

In conclusion, the Quetelet Index app represents a significant step forward in empowering users to monitor and understand their Body Mass Index (BMI) effectively. By providing a user-friendly interface, accurate calculations, and educational content, the app promotes health awareness and encourages healthier lifestyle choices. Moving forward, continued improvements such as enhanced data visualization, integration with health trackers, and multilingual support will further enhance the app's utility and impact. Overall, the Quetelet Index app serves as a valuable tool in promoting personal health management and contributing to overall well-being.

4.2 FUTURE WORK

While the Quetelet Index app has achieved its primary objectives, there are several areas for future enhancement and development:

Enhanced Data Visualization: Incorporating graphical representations of BMI trends over time or comparative analysis with population data can provide users with a deeper understanding of their health status.

Integration with Health Trackers: Integrating the app with wearable devices or health tracking platforms can enhance its functionality by automatically capturing and analyzing user data for more personalized insights.

Multilingual Support: Adding support for multiple languages can increase the app's accessibility and usability for a global audience.

Advanced Analytics: Implementing advanced analytics features, such as predictive modelling for future BMI trends or personalized recommendations based on user data, can offer users valuable insights into their health goals.

Community Engagement: Incorporating social features like user forums or challenges can foster a sense of community among app users, encouraging them to support and motivate each other in their wellness journeys.

Personalized Recommendations: Implement algorithms that provide personalized recommendations based on the user's BMI category, health goals, and lifestyle preferences. This could include tailored diet plans, exercise routines, and wellness tips.

Integration with Fitness Apps: Enable seamless integration with popular fitness and health-tracking apps to sync data such as calorie intake, exercise duration, and sleep patterns. This integration would provide a holistic view of the user's health and fitness journey.

Health Data Analysis: Incorporate advanced analytics to analyze historical BMI data, identify trends, and generate insights into long-term health outcomes. This could involve predictive modeling to forecast potential health risks and suggest preventive measures.

Social Engagement Features: Introduce social features within the app, such as user communities, challenges, and leaderboards. This fosters a sense of community and motivation among users, encouraging them to support each other in achieving their health goals.

Accessibility Improvements: Enhance accessibility by adding features such

as voice commands, screen reader compatibility, and customizable font sizes. This ensures that the app is inclusive and accessible to users with diverse needs.

Integration with Healthcare Providers: Explore partnerships with healthcare providers to integrate telemedicine services, virtual consultations, and access to personalized medical advice based on BMI and health data.

Gamification Elements: Incorporate gamification elements such as rewards, achievements, and progress tracking to make the health journey more engaging and motivating for users.

The future work for the Quetelet Index app includes implementing personalized recommendations based on BMI and health goals, integrating with fitness apps for holistic health tracking, analyzing health data for insights and risk prediction, adding social engagement features, improving accessibility, partnering with healthcare providers for telemedicine, and incorporating gamification elements for user motivation and engagement. These enhancements will make the app a comprehensive health management tool, empowering users to make informed decisions and achieve their wellness goals effectively.

REFERENCE

- [1] A. Smith, B. Johnson, C. Thompson, (2021) "Enhancing Mentorship through Technology in Educational Settings", Journal of Educational Technology & Society, 24.
- [2] C. Lee, D. Brown, E. Clark, (2019) "A Comprehensive Review of Flutter in Educational Application Development", Journal of Educational Computing Research, 15.
- [3] E. Rodriguez, (2020) "Hackathons and Learning: A Practical Approach", Routledge.
- [4] K. Patel, (2019) "Flutter for Mobile App Development in Education", Packt Publishing.
- [5] L. Chen, M. Wang, K. Li, (2020) "Impact of Hackathons on Skill Development: A Case Study Analysis", International Journal of Innovation and Technology Management, 10.
- [6] M. Davis, (2021) "Building Virtual Communities for Knowledge Exchange", Wiley.
- [7] R. Sharma, S. Patel, T. Nguyen, (2022) "Bridging the Gap: Successful Models of Industry-Academia Collaboration", International Journal of Technology Transfer and Commercialization, 8.
- [8] S. Williams, (2021) "Fostering Collaboration between Academia and Industry", Springer.
- [9] Thompson, D. Miller, E. Wilson, (2020) "Collaborative Learning Platforms: A Review of Effective Features", Computers & Education, 36.

APPENDICES

Sample Code

i) Main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:quetelet_index/result/result.dart';
import 'package:quetelet_index/screens/home/home.dart';
import 'package:quetelet_index/screens/welcome/welcome.dart';
import 'package:quetelet_index/singing/welcome_screen.dart';
import 'config/config.dart';
import 'firebase_options.dart';

//import 'firebase_options.dart';
Future <void> main() async{
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(WelcomeScreen1());
}

class QueteletIndex extends StatefulWidget {
  const QueteletIndex({super.key});

  @override
  State<QueteletIndex> createState() => _QueteletIndexState();
}

class _QueteletIndexState extends State<QueteletIndex> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'BIM App',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.blue.shade500),
        textTheme: appTextTheme,
        useMaterial3: true,
      ),
      initialRoute: '/welcomeScreen',
      routes:{
        '/queteletIndex': (context) => const QueteletIndex(),
      }
    );
  }
}
```

```

        '/welcomeScreen': (context) => const WelcomeScreen(),
        '/homeScreen': (context) => const HomeScreen(),
        '/resultScreen': (context) => const ResultScreen(),
    },
);
}
}
}

```

ii) customerScaffold.dart

```

import 'package:flutter/material.dart';

class CustomScaffold extends StatelessWidget {
  const CustomScaffold({Key? key, this.child}) : super(key: key);
  final Widget? child;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          backgroundColor: Colors.transparent,
          elevation: 0,
        ),
        extendBodyBehindAppBar: true, // Extend body behind app bar
        body: Stack(
          children: [
            Container(
              width: double.infinity,
              height: double.infinity,
              child: Image.asset(
                'assets/images/1.jpg',
                fit: BoxFit.cover,
              ),
            ),
            if (child != null)
              SafeArea(
                child: child!,
              ),
          ],
        ),
      ),
    );
  }
}

```

iii) Description_result_bmi.dart

```
import 'package:flutter/material.dart';

import 'calculate_BMI.dart';

class DescriptionResultBMI extends StatelessWidget {
  final CalculateBMI calculateBMI;

  const DescriptionResultBMI({
    super.key,
    required this.calculateBMI,
  });

  @override
  Widget build(BuildContext context) {
    String category =
      calculateBMI.getWeightCategory().toString().split('.').last;

    String categoryNameFirstLetter =
      category[0].toUpperCase() + category.substring(1);

    return Column(
      children: [
        Text(
          'Your BMI score is ',
          style: Theme.of(context).textTheme.displayMedium!.copyWith(
            fontSize: 20,
          ),
        ),
        const SizedBox(height: 20),
        Text(
          calculateBMI.getBMI().toStringAsFixed(0),
          style: Theme.of(context).textTheme.bodyLarge!.copyWith(
            fontSize: 120,
            color: calculateBMI.getTextColor(),
          ),
        ),
        Text(
          categoryNameFirstLetter,
          style: Theme.of(context).textTheme.displayMedium!.copyWith(
            fontSize: 20,
            color: calculateBMI.getTextColor(),
          ),
        ),
        const SizedBox(height: 20),
        Text.rich(
          textAlign: TextAlign.center,
```

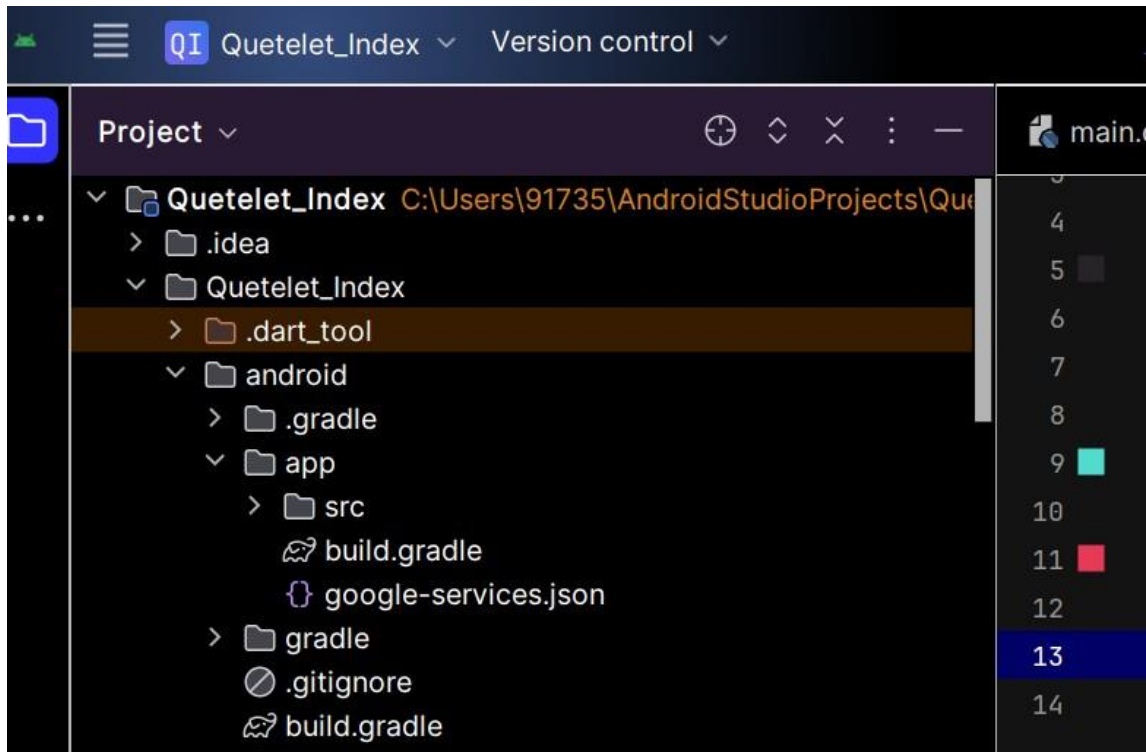
```

        TextSpan(
          text: 'That means your weight is in the',
          style:
Theme.of(context).textTheme.displayMedium!.copyWith(
          fontSize: 16,
          fontWeight: FontWeight.w400,
        ),
        children: [
          TextSpan(
            text: ' $categoryNameFirstLetter Weight',
            style: const TextStyle(
              fontSize: 18,
              fontStyle: FontStyle.italic,
            ),
          ),
          const TextSpan(text: ' category for adults of your
height '),
        ],
      ),
    ),
  ],
);
}
}

```

Screen Shot

i) Firebase configuration (Back-end)



ii) Dart code with output

