CHAPTER 1

INTRODUCTION

1.1: Overview

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013, at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

Fig 1.1.1: Android Studio

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

1.2: Applications

Weight Management: The app helps individuals monitor their BMI and track changes in their weight over time. By regularly using the BMI Calculator, users can set weight goals, track progress, and make informed decisions about their diet and exercise routines.

Health Screening: The app serves as a quick and convenient tool for health screening, allowing users to assess their weight status and identify potential risks associated with underweight, overweight, or obesity. It can serve as an initial step in recognizing the need for further medical evaluation or lifestyle changes.

Fitness and Exercise Planning: For individuals who engage in fitness and exercise activities, the BMI Calculator app provides a baseline measurement to gauge their progress. Users can monitor how their BMI changes as they follow exercise regimens, helping them adjust their routines and optimize their fitness goals.

Weight Loss Programs: The app can be integrated into weight loss programs and initiatives. It allows users to set realistic weight loss goals, track their progress, and receive reminders and tips to stay motivated and on track.

Health Education: The BMI Calculator app can serve as an educational tool, providing users with information about the importance of maintaining a healthy weight, the impact of weight on overall health, and strategies for weight management. It can offer resources, articles, and tips related to nutrition, exercise, and lifestyle modifications.

Healthcare Provider Collaboration: Users can share their BMI history and progress with healthcare professionals, facilitating informed discussions and decisions during medical appointments. The app can also integrate with electronic health records to provide a comprehensive view of the individual's health profile.

1.3: Problem Statement

Despite the increasing awareness of the importance of maintaining a healthy weight, many individuals struggle to accurately assess their body mass index (BMI) and understand its implications on their overall health. Existing BMI calculators often lack user-friendly interfaces, fail to provide personalized guidance, and do not integrate seamlessly with other health-related tools and resources. Consequently, there is a need for a comprehensive and user-centric BMI calculator app that addresses these limitations and empowers individuals to make informed decisions about their weight management.

Many people struggle to accurately determine their body mass index (BMI) and understand its significance for their health. Existing BMI calculators are often complex and lack user-friendly interfaces, making it difficult for individuals to input their height and weight accurately. Additionally, these calculators fail to provide clear interpretations of the BMI results and personalized guidance for weight management. There is a need for a simple and comprehensive BMI calculator app that is easy to use, provides understandable results, and offers personalized recommendations to help individuals make informed decisions about their weight and overall health.

1.4: Objectives

Accurate and Convenient BMI Calculation: The app aims to provide users with an accurate and efficient calculation of their BMI based on their height and weight. It should offer a user-friendly interface that allows for easy input of these parameters.

Clear Interpretation of Results: The app should provide clear and understandable interpretations of the BMI results. It should categorize the BMI into different weight categories (underweight, normal weight, overweight, or obese) and provide concise explanations of what each category means for the individual's health.

Personalized Guidance and Recommendations: The app should offer personalized guidance and recommendations based on the user's BMI results and individual goals. It may include suggestions for healthy weight management strategies, such as diet modifications, exercise routines, and lifestyle changes, tailored to the user's specific needs.

Integration with Health Trackers and Resources: The app should integrate seamlessly with popular health trackers and wearable devices to incorporate additional health data, such as physical activity levels and heart rate. It should also provide access to educational resources, nutritional information, exercise guidelines, and other relevant content to support users in their weight management journey.

Collaboration with Healthcare Providers: The app should facilitate collaboration between users and healthcare professionals. It should allow users to share their BMI history and progress with healthcare providers, enabling informed discussions and personalized recommendations during medical consultations.

CHAPTER 2:

SYSTEM REQUIREMENT SPECIFICATION

2.1: Software Requirements

- **IDE:** Android Studio
- **Technologies:** Java, XML, Android
- **Database**: SQLite

2.2: Hardware Requirements

- **Disk space:** 250MB or more
- **Processor:** i3 or higher
- **Operating System:** Windows 7 or higher, Mac OS
- **Java version:** JDK 8
- **Disk space for Android Studio:** 500 MB
- **Space for Android SDK:** 1.5 GB or higher
- **Processor speed:** 2.3 GHz
- **RAM:** 8GB or higher
- Android phone with API level 16 or higher (Jelly bean)

CHAPTER 3:

SYSTEM ANALYSIS AND DESIGN

3.1: System Analysis

Functional Requirements:

- **User Registration:** The app should allow users to create an account or register using their email or social media accounts to access personalized features.
- **BMI Calculation:** The app should provide a BMI calculation feature that takes user inputs of height and weight and calculates the BMI using the appropriate formula.
- **Integration with Health Trackers:** The app should integrate with popular health trackers or wearable devices to incorporate additional health data, such as physical activity levels and heart rate.
- **Reminders and Notifications:** The app should include customizable reminders and notifications to remind users to input their weight regularly, stay motivated, and track their progress.
- **Data Sharing with Healthcare Providers:** The app should allow users to share their BMI history and ensuring collaboration and informed discussions during medical consultations.

Non-Functional Requirements:

- **User-Friendly Interface:** The app should have an intuitive and user-friendly interface that is easy to navigate, allowing users to input their data effortlessly.
- **Privacy and Security:** The app should prioritize user data privacy and employ security measures to protect personal information.
- **Compatibility:** The app should be compatible with a variety of mobile devices and operating systems to reach a wide range of users.
- **Scalability:** The app should be designed to handle a growing number of users and data without compromising performance or functionality.
- **Accessibility:** The app should consider accessibility standards and guidelines to ensure inclusivity for users with disabilities.
- **Error Handling:** The app should handle errors gracefully, providing informative messages and guidance to users in case of input errors or system failures.

3.2: System Design

User Interface (UI):

The UI should have a clean and intuitive design, making it easy for users to navigate and interact with the app.Visual representations, such as charts or graphs, can be used to display BMI trends and weight progress.

User Registration and Authentication:

The app should include a registration and login system to allow users to create and manage their accounts.Authentication mechanisms, such as email verification or social media login, can be implemented for user security.

BMI Calculation and Interpretation:

The app should have a module for BMI calculation using the standard formula: BMI = weight (kg) / (height (m))^2.The calculated BMI should be categorized into weight categories (underweight, normal weight, overweight, or obese) with clear explanations provided to the user.

Personalized Recommendations:

Based on the user's BMI results and goals, the app should provide personalized recommendations for weight management.Recommendations can include suggested calorie intake, meal plans, exercise routines, and lifestyle modifications.

Integration with Health Trackers and Resources:

The app can integrate with popular health trackers or wearable devices to gather additional health data, such as physical activity levels or heart rate.Access to educational resources, including articles, videos, or links to reputable sources, should be provided to support users in making informed decisions about their weight management.

Reminders and Notifications:

The app should include customizable reminders and notifications to prompt users to input their weight regularly, stay motivated, and track their progress.

Integration with Healthcare Providers:

The app should provide functionality for users to share their BMI history and progress with healthcare professionals, facilitating collaboration and informed discussions during medical consultations.

Error Handling and User Feedback:

The app should handle input errors gracefully, providing informative messages and guidance to users when incorrect or incomplete data is entered. within the app.

CHAPTER 4:

IMPLEMENTATION

4.1: Description of Implementation

The implementation of the BMI Calculator app involves translating the system design into actual software using appropriate programming languages, frameworks, and tools. Here is a description of the implementation process for the app:

User Interface (UI) Development:

Develop the UI screens using frontend technologies such as HTML, CSS, and JavaScript. Implement user registration/login screens with appropriate form validations and authentication mechanisms.

Backend Development:

Choose a suitable backend technology stack, such as Node.js, Python, or Ruby on Rails, based on the project requirements and team expertise.Develop algorithms for personalized recommendations based on BMI results and user goals.

Database Integration and Data Management:

Design and implement database schemas and models to represent the necessary data structures.

Implement data access and manipulation methods to store and retrieve user-specific information, BMI history, and goals.

Integration with External APIs and Services:

If integrating with health trackers or wearable devices, utilize relevant APIs and SDKs provided by the respective manufacturers.Implement API calls and data synchronization mechanisms to retrieve and process additional health data, such as physical activity levels or heart rate.

Notification and Reminder Systems:

Integrate notification services, such as Firebase Cloud Messaging or Pusher, to send customizable reminders and notifications to users.Implement scheduling mechanisms to trigger reminders based on user preferences and input frequency.

Error Handling and User Feedback:

Implement error handling mechanisms to gracefully handle input errors, exceptions, and edge cases.Provide appropriate error messages and feedback to users when necessary, ensuring a smooth user experience.

Security and Privacy Considerations:

Implement necessary security measures, such as encryption and secure transmission protocols, to protect user data and ensure privacy.Adhere to industry-standard security practices to safeguard user information from unauthorized access or breaches.

Testing and Quality Assurance:

Conduct comprehensive testing, including unit testing, integration testing, and user acceptance testing, to identify and resolve any issues or bugs.Perform usability testing to ensure a seamless user experience and validate the accuracy of BMI calculations and interpretations.

Deployment and Maintenance:

Deploy the app to a production environment, such as cloud hosting platforms like AWS, Azure, or Google Cloud, or app stores for mobile platforms.Continuously monitor and maintain the app, addressing any bugs, performance issues, or user feedback through regular updates and improvements.

4.2: XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BMI Calculator"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="16dp"/>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enter your height (in cm):"
    android:textSize="18sp"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="16dp"/>

<EditText
    android:id="@+id/editTextHeight"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="number"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enter your weight (in kg):"
    android:textSize="18sp"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginTop="16dp"/>

<EditText
    android:id="@+id/editTextWeight"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="number"/>

<Button
    android:id="@+id/calculateButton"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Calculate"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"/>

    <TextView
        android:id="@+id/resultTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"/>

</LinearLayout>
```

4.3: JAVA Code

```
package com.example.bmicalculator;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RelativeLayout;
```

```
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {


    TextView mcurrentheight;
    TextView mcurrentweight,mcurrentage;
    ImageView mincrementage,mdecrementage,mincrementweight,mdecrementweight;
    SeekBar mseekbarforheight;
    Button mcalculatebmi;
    RelativeLayout mmale,mfemale;

    int intweight=55;
    int intage=22;
    int currentprogress;
    String mintprogress="170";
    String typerofuser="0";
    String weight2="55";
    String age2="22";

    @SuppressLint("ResourceAsColor")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        getSupportActionBar().hide();
        mcurrentage=findViewById(R.id.currentage);
        mcurrentweight=findViewById(R.id.currentweight);
```

```
        mcurrentheight=findViewById(R.id.currentheight);
        mincrementage=findViewById(R.id.incrementage);
        mdecrementage=findViewById(R.id.decrementage);
        mincrementweight=findViewById(R.id.incremetweight);
        mdecrementweight=findViewById(R.id.decrementweight);
        mcalculatebmi=findViewById(R.id.calculatebmi);
        mseekbarforheight=findViewById(R.id.seekbarforheight);
        mmale=findViewById(R.id.male);
        mfemale=findViewById(R.id.female);



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

mmale.setBackground(ContextCompat.getDrawable(getApplicationContext(),R.drawable.malefe
malefocus));

mfemale.setBackground(ContextCompat.getDrawable(getApplicationContext(),R.drawable.malef
emalenotfocus));
                        typerofuser="Male";


            }
        });



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

```
mfemale.setBackground(ContextCompat.getDrawable(getApplicationContext(),R.drawable.malef
emalefocus));

mmale.setBackground(ContextCompat.getDrawable(getApplicationContext(),R.drawable.malefe
malenotfocus));
            typerofuser="Female";
        }
    });

     mseekbarforheight.setMax(300);
     mseekbarforheight.setProgress(170);
     mseekbarforheight.setOnSeekBarChangeListener(new  SeekBar.OnSeekBarChangeListener()
{
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

           currentprogress=progress;
           mintprogress=String.valueOf(currentprogress);
           mcurrentheight.setText(mintprogress);

        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
```

```
});


mincrementweight.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        intweight=intweight+1;
        weight2=String.valueOf(intweight);
        mcurrentweight.setText(weight2);
    }
});

mincrementage.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        intage=intage+1;
        age2=String.valueOf(intage);
        mcurrentage.setText(age2);
    }
});


mdecrementage.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        intage=intage-1;
        age2=String.valueOf(intage);
        mcurrentage.setText(age2);
    }
});
```

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

        intweight=intweight-1;
         weight2=String.valueOf(intweight);
        mcurrentweight.setText(weight2);
    }
});



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

        if(typerofuser.equals("0"))
        {
            Toast.makeText(getApplicationContext(),"Select Your Gender First",Toast.LENGTH_SHORT).show();
        }
        else if(mintprogress.equals("0"))
        {
            Toast.makeText(getApplicationContext(),"Select Your Height First",Toast.LENGTH_SHORT).show();
        }
        else if(intage==0 || intage<0)
        {
            Toast.makeText(getApplicationContext(),"Age is Incorrect",Toast.LENGTH_SHORT).show();
```

```
            }

            else if(intweight==0|| intweight<0)
            {
               Toast.makeText(getApplicationContext(),"Weight Is
Incorrect",Toast.LENGTH_SHORT).show();
            }
            else {

               Intent intent = new Intent(MainActivity.this, bmiactivity.class);
               intent.putExtra("gender", typerofuser);
               intent.putExtra("height", mintprogress);
               intent.putExtra("weight", weight2);
               intent.putExtra("age", age2);
               startActivity(intent);

            }

         }
      });

   }
}
```

CHAPTER 5:

TESTING

5.1: Types of Testing

For a BMI calculator app, various types of testing can be conducted to ensure its functionality, usability, and reliability. Here are some types of testing that can be performed for a BMI calculator app:

Unit Testing: This type of testing focuses on testing individual components or units of code in isolation. It ensures that each function or module of the app works as expected. For a BMI calculator app, unit testing can be performed on the BMI calculation logic and other critical functions.

Integration Testing: Integration testing verifies the interaction and compatibility between different modules or components of the app. It ensures that the BMI calculator app functions seamlessly when integrated with other modules, such as user authentication, database management, and external APIs for health data integration.

Functional Testing: Functional testing validates that the app meets the specified functional requirements. It involves testing the app's features, such as inputting height and weight, calculating BMI, displaying weight categories, and providing personalized recommendations. This testing ensures that the core functionality of the app works as intended.

Usability Testing: Usability testing evaluates the user-friendliness and intuitiveness of the app's interface. It involves observing users as they interact with the app to accomplish tasks, such as entering height and weight, setting goals, and interpreting results. Usability testing helps identify any usability issues, navigation difficulties, or confusing elements that may impact the user experience.

Performance Testing: Performance testing assesses the responsiveness and efficiency of the app under varying workloads. It includes testing the app's speed, responsiveness, and resource utilization when performing BMI calculations, handling data input, and generating results. Performance testing ensures that the app functions smoothly and provides a seamless user experience.

Compatibility Testing: Compatibility testing verifies that the BMI calculator app works correctly across different devices, platforms, and screen sizes. It involves testing the app on various mobile devices, operating systems, and screen resolutions to ensure compatibility and consistent behavior.

Security Testing: Security testing assesses the app's resistance to unauthorized access, data breaches, or other security vulnerabilities. It involves testing the app's authentication mechanisms, data encryption, and secure transmission of sensitive information. Security testing ensures the protection of user data and compliance with privacy regulations.

Regression Testing: Regression testing is performed to ensure that new changes or updates to the app do not introduce any unintended issues or regressions in previously working features. It involves retesting previously tested functionalities to ensure they still work as expected after implementing new changes or fixes.

User Acceptance Testing (UAT): UAT involves testing the app with end-users to validate its usability, functionality, and overall satisfaction. It allows real users to provide feedback, identify any issues, and validate that the app meets their expectations and requirements.

By conducting these types of testing, the BMI calculator app can be thoroughly evaluated, ensuring its accuracy, functionality, usability, performance, security, and compatibility with different devices and user requirements.

5.2: Test Cases

Test Case: Valid BMI Calculation

- Description: Verify that the app calculates the BMI correctly when valid height and weight inputs are provided.
- Inputs: Height = 170 cm, Weight = 70 kg
- Expected Output: BMI = 24.22 (Normal weight)

Test Case: Invalid Height Input

- Description: Test the app's behavior when an invalid height input is provided (e.g., negative value or non-numeric characters).
- Inputs: Height = -160 cm

Test Case: Invalid Weight Input

- Description: Test the app's behavior when an invalid weight input is provided (e.g., negative value or non-numeric characters).
- Inputs: Weight = -75 kg
- Expected Output: Display an error message indicating an invalid weight input.

Test Case: Display of Weight Category

- Description: Verify that the app correctly displays the weight category based on the calculated BMI.
- Inputs: Height = 180 cm, Weight = 90 kg
- Expected Output: Display "BMI = 27.78 (Overweight)"

Test Case: Personalized Recommendations

- Description: Check if the app provides personalized recommendations based on the user's BMI and weight category.
- Inputs: Height = 160 cm, Weight = 75 kg
- Expected Output: Display recommendations for weight management, such as exercise routines or dietary suggestions for the "Obese" category.

Test Case: Goal Setting and Progress Tracking

- Description: Test the app's ability to set weight goals and track progress over time.
- Inputs: Height = 175 cm, Weight = 80 kg, Set goal weight = 70 kg
- Expected Output: Allow the user to set the goal weight and track progress through visual representations, such as a progress bar or graph.

Test Case: Integration with Health Trackers

- Description: Validate the integration with external health trackers to retrieve additional health data for BMI calculation.

- Inputs: Height = 165 cm, Weight = 68 kg, Sync with fitness tracker to get daily step count.
- Expected Output: Successfully retrieve step count data from the fitness tracker and incorporate it into the BMI calculation or personalized recommendations.

Test Case: User Authentication and Data Privacy

- Description: Verify the app's user authentication system and ensure the privacy of user data.
- Inputs: Test user login credentials
- Expected Output: Allow authenticated access to user-specific data and ensure data privacy measures are in place to protect sensitive information.

Test Case: Usability and Interface Testing

- Description: Evaluate the app's user interface, navigation, and overall usability.
- Inputs: Interact with various UI elements, enter height and weight, navigate through screens.
- Expected Output: Ensure the app's interface is intuitive, elements are properly aligned, and navigation is smooth and user-friendly.

Test Case: Performance Testing

- Description: Test the app's performance under various workloads and stress conditions.
- Inputs: Perform multiple calculations within a short time frame or simulate high user traffic.
- Expected Output: Ensure the app responds quickly, performs calculations accurately, and remains stable under high load conditions.

CHAPTER 6:

RESULTS & SNAPSHOTS

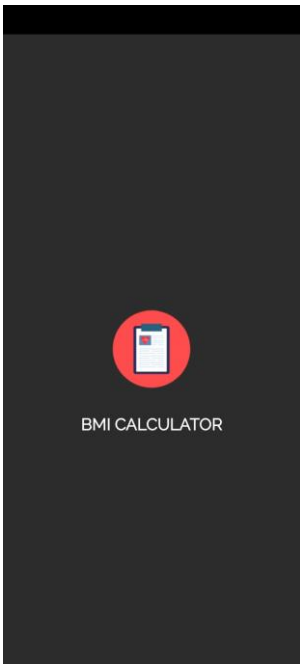

Fig 6.1:Opening page of BMI calculator app

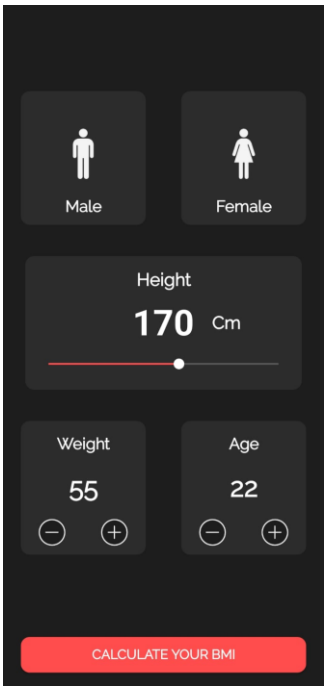

Fig 6.2:Enter the details to calculate your BMI

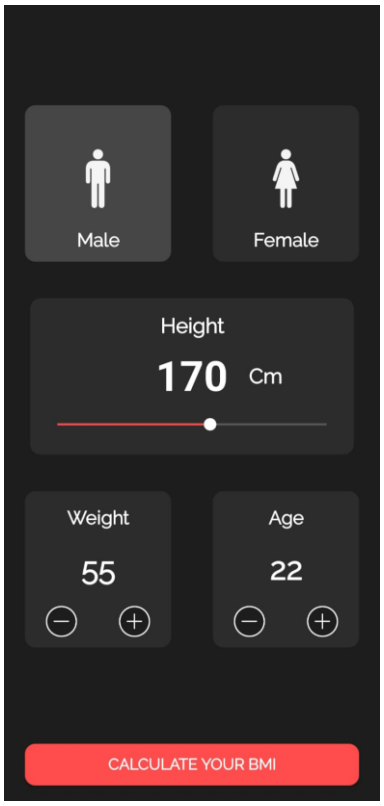

Fig 6.3:Calculating the BMI of Male

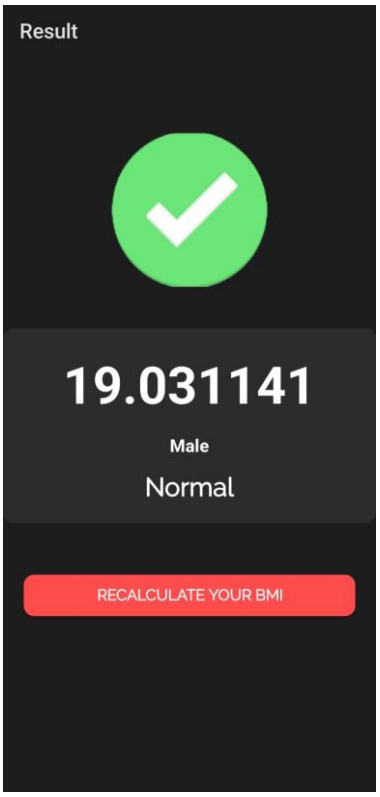

Fig 6.4: Calculated value of Male

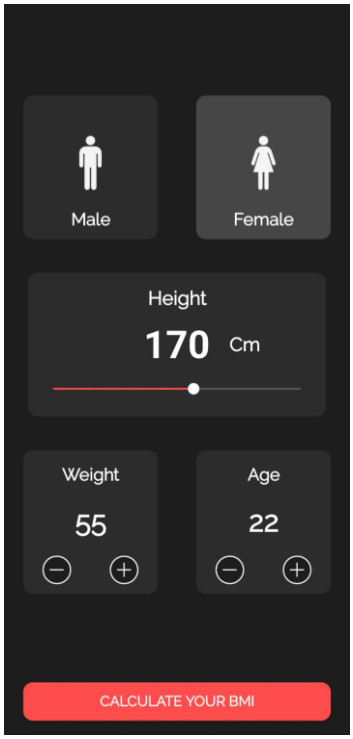

Fig 6.5:Calculating the BMI of Female

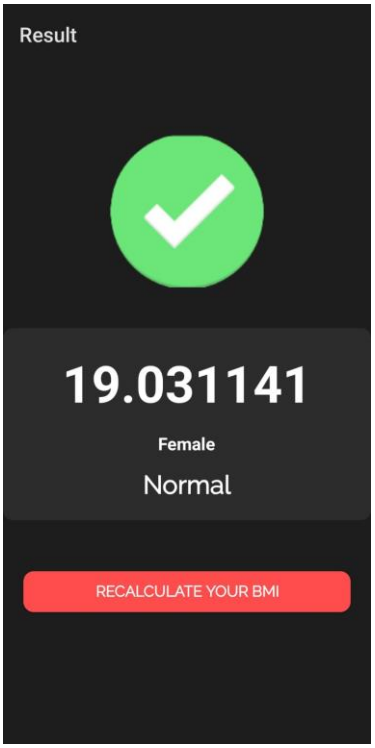

Fig 6.6: Calculated value of Female

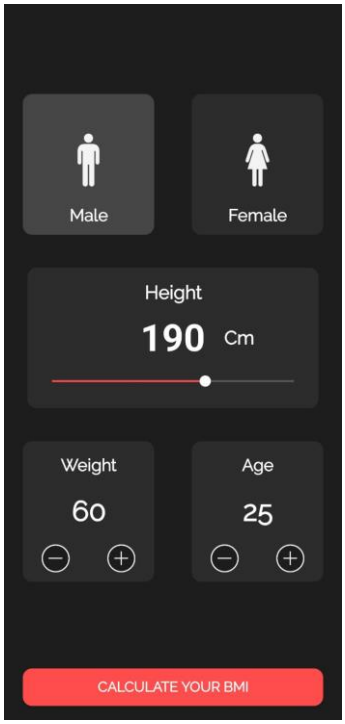

Fig 6.7:Calculating the BMI of male

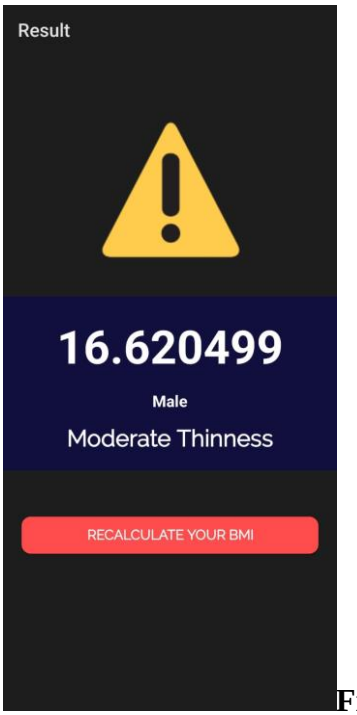

Fig 6.8: Value of male with moderate thinness

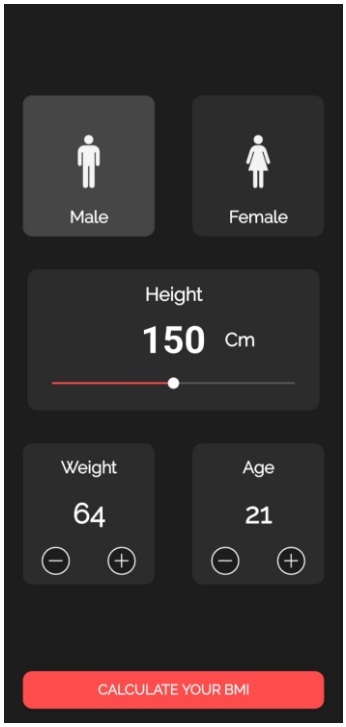

Fig 6.9:Calculating the BMI of male

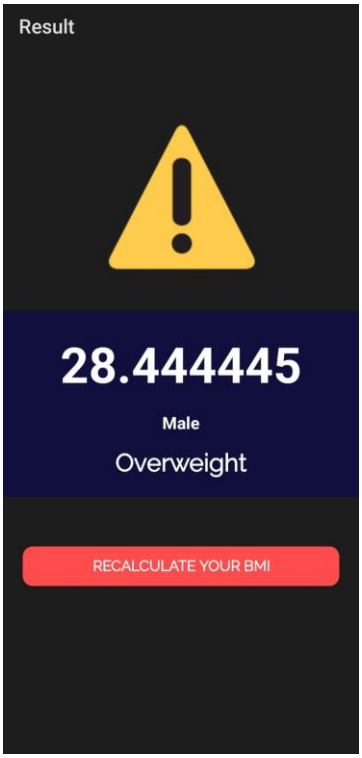

Fig 6.10:Value of male with Overweight

CONCLUSION

In conclusion, the BMI calculator app serves as a valuable tool for individuals to assess their body mass index (BMI) and gain insights into their weight status. By providing a simple and convenient way to calculate BMI based on height and weight inputs, the app empowers users to understand the implications of their weight on their overall health.Through clear interpretations and weight category classifications, the app offers users a meaningful understanding of their BMI results. Whether it's categorizing individuals as underweight, normal weight, overweight, or obese, this information allows users to gauge their weight status and potential health risks associated with different weight categories.

While the BMI calculator app is a valuable tool, it's important to remember that BMI is just one aspect of assessing health. It does not account for factors such as muscle mass, body composition, or specific health conditions. Therefore, it's advisable to consult healthcare professionals for a comprehensive evaluation and to develop personalized weight management strategies.

Overall, the BMI calculator app serves as a supportive companion on an individual's weight management journey. By providing accurate calculations, clear interpretations, personalized recommendations, and access to additional resources, the app empowers users to make informed decisions and take control of their weight and overall well-being.

REFERENCES

[1]. "The Body Mass Index: Uses and Limitations" by George A. Bray and Claude Bouchard

[2]."Weight Management: A Practitioner's Guide" by Anne E. McTiernan and Suzanne Phelan

[3]."Nutrition for Health, Fitness, and Sport" by Melvin H. Williams and Eric Rawson

[4]."The Complete Guide to Sports Nutrition" by Anita Bean