THE STATE UNIVERSITY OF ZANZIBAR

DEPARTMENT OF COMPUTER SCIENCE

AND

INFORMATION TECHNOLOGY

COURSE CODE: WT822

COURSE TITTLE: ADVANCE WEBSITE PROGRAMMING

COURSE LECTURER: Mr Massoud Hamad

ACADEMIC YEAR: 2022/2023

TASK: **FINAL PROJECT**

PARTICIPANT:

NAME	REGISTRATION NO:
ABRAHMAN ABDALLAH ZAHRAN	BITA/5/21/031/TZ

PROJECT TITLE: **BODY MASS INDEX**

Introduction

BMI is a measurement of a person's leanness or corpulence based on their height and weight, and is intended to quantify tissue mass. It is widely used as a general indicator of whether a person has a healthy body weight for their height. Specifically, the value obtained from the calculation of BMI is used to categorize whether a person is underweight, normal weight, overweight, or obese depending on what range the value falls between.

Problem Statement

Being overweight increases the risk of a number of serious diseases and health conditions. Below is a list of said risks, according to the Centers for Disease Control and Prevention (CDC). High blood pressure: Higher levels of LDL cholesterol, which is widely considered "bad cholesterol," lower levels of HDL cholesterol, considered to be good cholesterol in moderation, and high levels of triglycerides, Type II diabetes: Coronary heart disease, Stroke.

Objective of the system

Generally Objective

The main objective is to design and implement body mass index system

Challenges of current existing system

The challenges of current system is the no means of technology that are used to inform the person about the body mass index. Lack of efficient resource, direct person go to the hospital to know or measurement their health.

Solution to the current business operations

Having a greater understanding of what make body mass index. It can person to go at hospital and saving money because we are going to design the platform with ability to achieve operational excellence in measure the body mass. It can be more efficient, most effective way for the client, it can take care serious to the client to manage day to day operation to do a better.

Deliverable

The deliverable of body mass index system is to ensure a healthy lifestyle, recommends eating lots of fruit and vegetables, reducing fact, sugar and salt intake and exercising. Based on height and weight, people can check their body mass index (BMI) to see if they are overweight.

Requirements

a. Function requirement

FREQ001	The system shall allow to prompt user to enter the data in numerical.
FREQ002	The system shall give the specific feedback for the current data inserted.

b. Non function requirement

NREQ001	There must be system alert / notification module to administration and clients
NREQ002	System must provide a very proper storage of the data so that not to affect the performance of the system like loading and it's also module based hence their is multiple point of control and failure

c. Security requirement

SREQ001	Data Encryption
SREQ002	Authentication and Authorization

Development methods/ methodologies

Body mass index system is intended to be developed in **Agile methodology**

Advantages:

- ➔ As compared to other methods, the Agile process is quick enough to adjust to changes throughout the development cycle
- ➔ As compared to other methods, the Agile process is quick enough to adjust to changes requested by the clients throughout the development cycle
- ➔ It creates a unique work culture that helps to increase team morale

Disadvantages

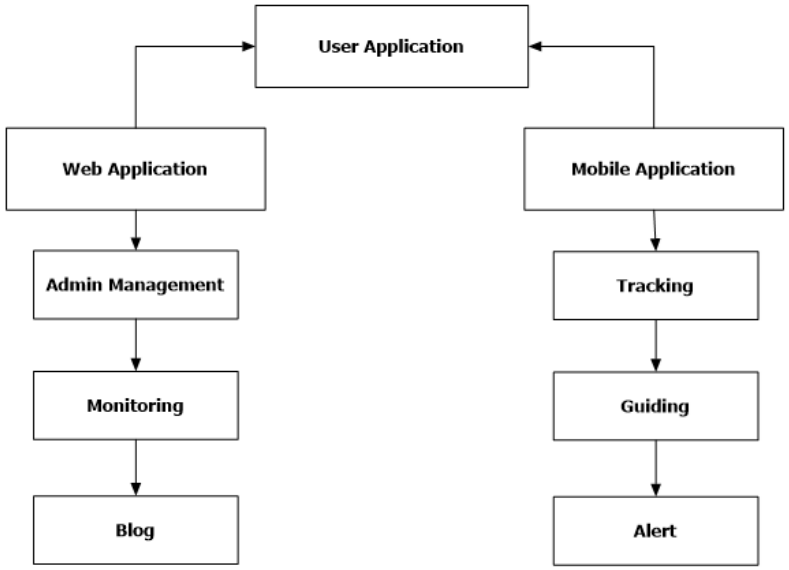
- ➔ Poor resource planning
- ➔ No finite end

➔ Difficult measurement

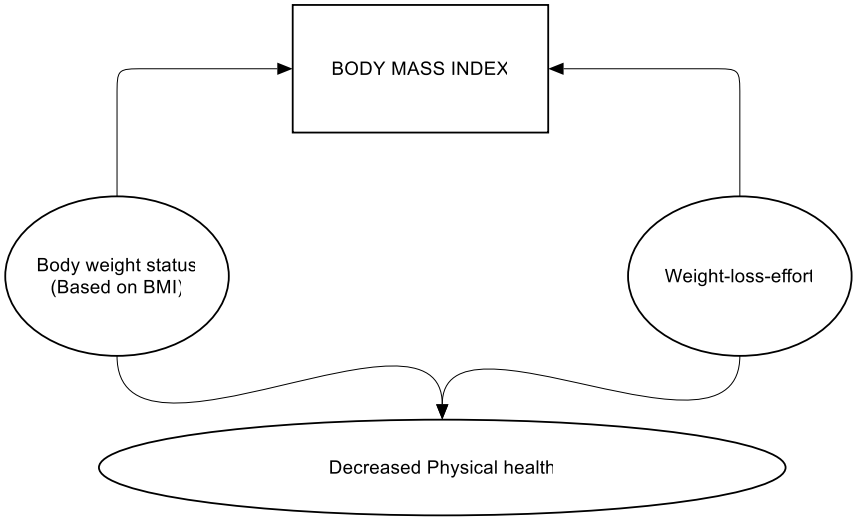
Why decide to use this methods

Because As compared to other methods, the agile process is quick enough to adjust to changes requested by the clients throughout the development cycle.

Architecture of the system (Process flow) which architecture



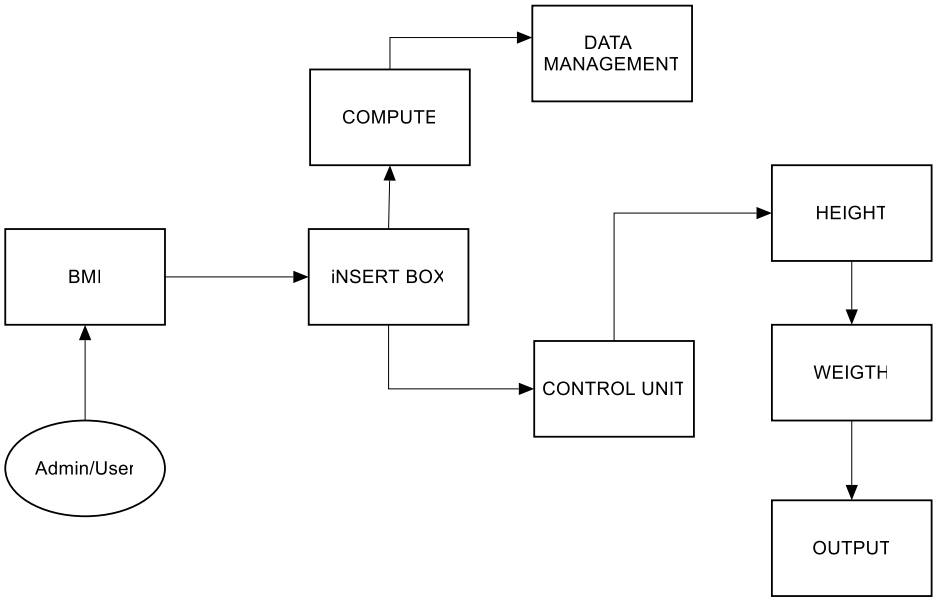
Conceptual design of the system



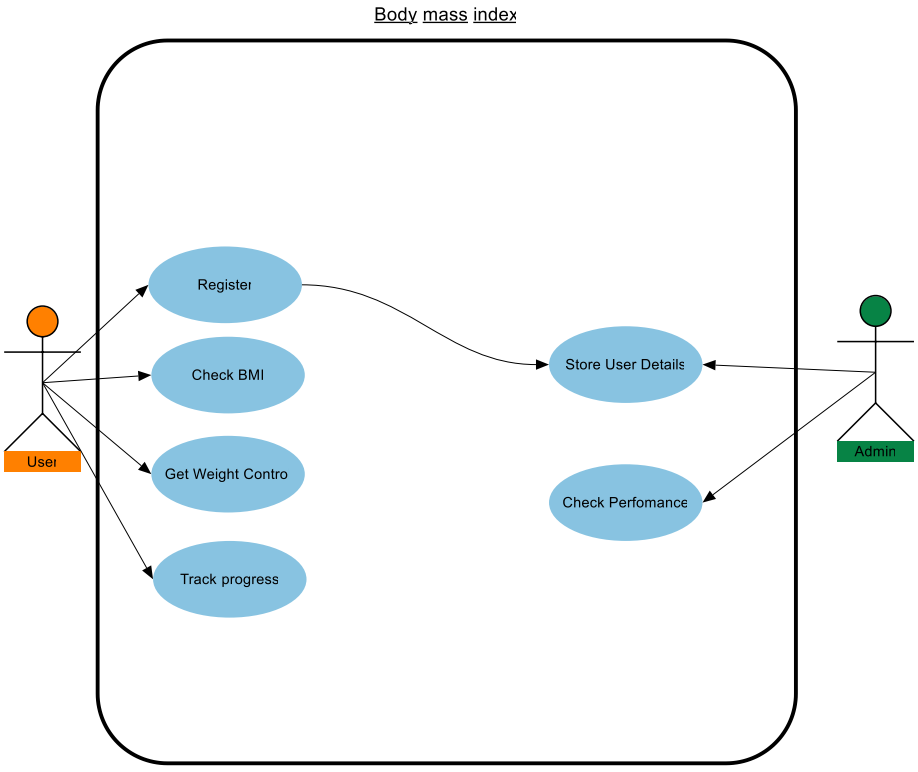
Database Design

	User
	User ID f_Name l_Name Age Gender Weight Height Status

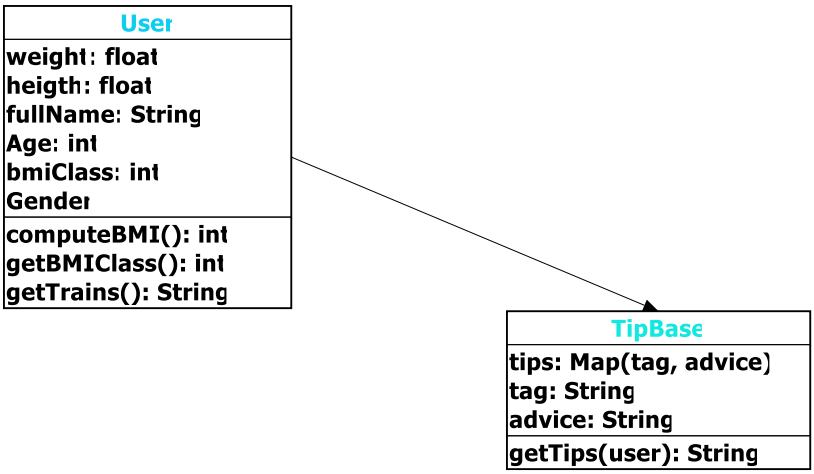
Activity flow of the system



Use case to every functional requirement



Class Diagram



Development Technology (language) Front-end and Back-end

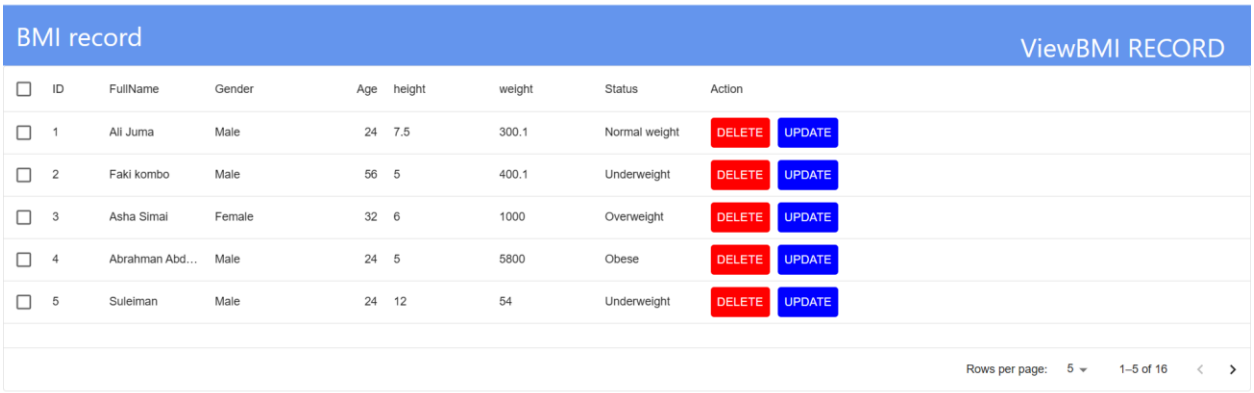
The proposed system will developed by using two technology the Front-end will used the React and back-end will use the Spring Boot.

Deployment Technologies/Methods

Deployment technologies are the various approaches and tools used to deploy software applications to production environments. There are several deployment technologies and methods available for these system the deployment technologies commonly used are:-

- ➔ Cloud computing
- ➔ Wireless Communication
- ➔ Mobile application
- ➔ Geographical Information system
- ➔ APIs and Web Services

Interface web based



Front end code by using react

TableView.jsx

```
import React,{useEffect,useState} from 'react';
import { DataGrid } from '@mui/x-data-grid';
import axios from 'axios';
import {useParams} from 'react';
import { Button } from '@mui/material';


export default function Tables() {

    const [data, setData] = useState([]);
    const [isDeletede, setIsDeleted] = useState(false);


    const fetchData = async () => {
        try {
          const response = await axios.get('http://localhost:8096/api/bmi-
records/listUser');
          setData(response.data);
            // console

        } catch (error) {
            console.error('Error fetching data:', error);
        }
    };
    useEffect(() => {

        fetchData();
    }, []);


        const handleDelete = async (id) =>{
        try{
```

```
            await axios.delete(`http://localhost:8096/api/bmi-
records/deleteData/${id}`);
            setIsDeleted(true);

            fetchData();
        }
        catch(error){
            console.log(error);
        }
        if (isDeletede){
            return alert("deleted");
        }
    }

  const columns = [
  { field: 'id', headerName: 'ID', width: 70 },
  { field: 'fullName', headerName: 'FullName', width: 130 },
  { field: 'gender', headerName: 'Gender', width: 130 },
  {
    field: 'age',
    headerName: 'Age',
    type: 'number',
    width: 90,
  },
  { field: 'height', headerName: 'height', width: 130 },
  { field: 'weight', headerName: 'weight', width: 130 },
  { field: 'bmiCategory', headerName: 'Status', width: 130 },

          {field: 'action', headerName : 'Action', width: 200,

        renderCell: (params) =>{
          return (
              <div className="cellAction" style={{}}>
                  <Button onClick={() => handleDelete(params.id)}
style={{backgroundColor:"Red", color:"white", marginRight:
"10px"}}  >Delete</Button>
                  <Button onClick={() => handleDelete(params.id)}
style={{backgroundColor:"blue", color:"white", marginRight:
"10px"}}  >Update</Button>

              </div>
            );
        }
      },

];

  return (
```

```
    <div style={{ height: 400, width: '100%' }}>
      <DataGrid
        rows={data}
        columns={columns}
        initialState={{
          pagination: {
            paginationModel: { page: 0, pageSize: 5 },
          },
        }}
        pageSizeOptions={[5, 10]}
        checkboxSelection
      />
    </div>
  );
}
```

BACK-END CODE BY USING SPRING-BOOT

/Model

BmiRecodrs.java

```
package com.body.mass.index.bodyMassIndex.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name="bmirecord")
public class BMIRecords {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String fullName;
    @Column(nullable = false)
    private int age;
    @Column(nullable = false)
    private String gender;
    @Column(nullable = false)
    private double weight;
    @Column(nullable = false)
    private double height;
    @Column(nullable = false)
    private String bmiCategory;
```

```
    public BMIRecords() {
    }
    public BMIRecords(String fullName, int age, String gender, double weight,
double height, String bmiCategory) {
        this.fullName = fullName;
        this.age = age;
        this.gender = gender;
        this.weight = weight;
        this.height = height;
        this.bmiCategory = bmiCategory;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getFullName() {
        return fullName;
    }
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public double getWeight() {
        return weight;
    }
    public void setWeight(double weight) {
        this.weight = weight;
    }
    public double getHeight() {
        return height;
    }
    public void setHeight(double height) {
        this.height = height;
```

```
    }
    public String getBmiCategory() {
        return bmiCategory;
    }
    public void setBmiCategory(String bmiCategory) {
        this.bmiCategory = bmiCategory;
    }
}
```

/Controller

BmiRecordsController.java

```
package com.body.mass.index.bodyMassIndex.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.body.mass.index.bodyMassIndex.model.BMIRecords;
import com.body.mass.index.bodyMassIndex.repository.BmiRecordsRepository;

@CrossOrigin
@RestController
@RequestMapping("/api/bmi-records")

public class BmiRecordsController {

    @Autowired
    private final BmiRecordsRepository BmiRecordsRepository;


    public BmiRecordsController(BmiRecordsRepository BmiRecordRepository) {
        this.BmiRecordsRepository = BmiRecordRepository;
    }
    @PostMapping("/adduser")
    public BMIRecords addBmiRecords(@RequestBody BMIRecords bmiRecord){
```

```
        return BmiRecordsRepository.save(bmiRecord);
    }

    @GetMapping("/listUser")
    public List<BMIRecords> getAllBmiRecords() {
        return BmiRecordsRepository.findAll();
    }
    @PutMapping("/updateData/{id}")
    public ResponseEntity<?> update(@PathVariable long id){
        return ResponseEntity.ok(((BmiRecordsController)
BmiRecordsRepository).update(id));
    }

    @DeleteMapping("deleteData/{id}")
    public void delete(@PathVariable long id){
        BmiRecordsRepository.deleteById(id);
    }

}
```

Repository

BmiRecordsRepository.java

```
package com.body.mass.index.bodyMassIndex.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.body.mass.index.bodyMassIndex.model.BMIRecords;

public interface BmiRecordsRepository extends JpaRepository<BMIRecords, Long> {

    // Add any custom query methods if needed
}
```

/Resource

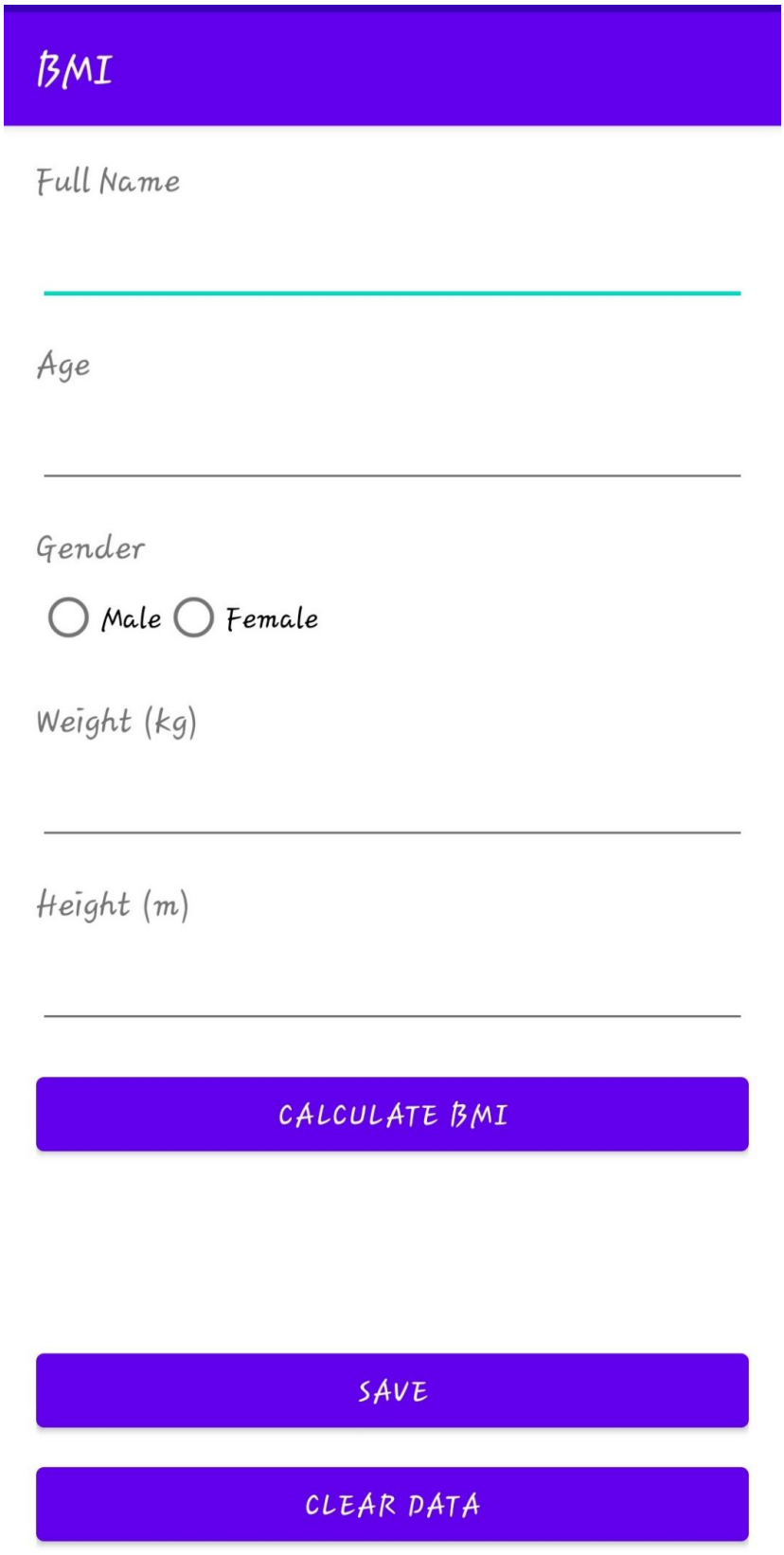
application.properties

```
server.port=8096
spring.datasource.url=jdbc:mysql://localhost:3306/body_mass_index?useSSL=false
spring.datasource.username=root
spring.datasource.password=

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.hibernate.ddl-auto=update
```

/Main Class

```java
package com.body.mass.index.bodyMassIndex;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BodyMassIndexApplication {

    public static void main(String[] args) {
        SpringApplication.run(BodyMassIndexApplication.class, args);
    }

}
```

Interface Mobile Application based



Activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/fullNameTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Full Name"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/fullNameEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/fullNameTextView"
        android:layout_marginTop="8dp" />

    <TextView
        android:id="@+id/ageTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/fullNameEditText"
        android:layout_marginTop="16dp"
        android:text="Age"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/ageEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/ageTextView"
        android:layout_marginTop="8dp"
        android:inputType="number" />

    <TextView
        android:id="@+id/genderTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/ageEditText"
        android:layout_marginTop="16dp"
        android:text="Gender"
        android:textSize="16sp" />

    <RadioGroup
        android:id="@+id/genderRadioGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/genderTextView"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/maleRadioButton"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male" />

        <RadioButton
            android:id="@+id/femaleRadioButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female" />
    </RadioGroup>

    <TextView
        android:id="@+id/weightTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/genderRadioGroup"
        android:layout_marginTop="16dp"
        android:text="Weight (kg)"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/weightEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/weightTextView"
        android:layout_marginTop="8dp"
        android:inputType="numberDecimal" />

    <TextView
        android:id="@+id/heightTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/weightEditText"
        android:layout_marginTop="16dp"
        android:text="Height (m)"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/heightEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/heightTextView"
        android:layout_marginTop="8dp"
        android:inputType="numberDecimal" />

    <Button
        android:id="@+id/calculateButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/heightEditText"
        android:layout_marginTop="16dp"
        android:text="Calculate BMI" />

    <TextView
        android:id="@+id/bmiTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/calculateButton"
        android:layout_marginTop="16dp"
        android:textSize="18sp" />
```

```
    <TextView
        android:id="@+id/categoryTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/bmiTextView"
        android:layout_marginTop="8dp"
        android:textSize="18sp" />

    <Button
        android:id="@+id/saveButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/categoryTextView"
        android:layout_marginTop="16dp"
        android:text="Save" />

    <Button
        android:id="@+id/clearButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/saveButton"
        android:layout_marginTop="8dp"
        android:text="Clear Data" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.bmi;


import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;


import androidx.appcompat.app.AppCompatActivity;


import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;


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

import java.util.HashMap;
import java.util.Map;

public class MainActivity extends AppCompatActivity {
```

```
    private EditText fullNameEditText;
    private EditText ageEditText;
    private RadioGroup genderRadioGroup;
    private EditText weightEditText;
    private EditText heightEditText;
    private Button calculateButton;
    private Button saveButton;
    private Button clearButton;
    private TextView bmiTextView;
    private TextView categoryTextView;

    private Long id;
    String bmiCategory;

    private DatabaseHelper databaseHelper;


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



        fullNameEditText = findViewById(R.id.fullNameEditText);
        ageEditText = findViewById(R.id.ageEditText);
        genderRadioGroup = findViewById(R.id.genderRadioGroup);
        weightEditText = findViewById(R.id.weightEditText);
        heightEditText = findViewById(R.id.heightEditText);
        calculateButton = findViewById(R.id.calculateButton);
        saveButton = findViewById(R.id.saveButton);
        clearButton = findViewById(R.id.clearButton);
        bmiTextView = findViewById(R.id.bmiTextView);
        categoryTextView = findViewById(R.id.categoryTextView);



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

        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                save();
            }
        });
        clearButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                clearData();
            }
        });
    }

    private void calculateBMI() {
        String weightStr = weightEditText.getText().toString();
```

```
        String heightStr = heightEditText.getText().toString();

        if (weightStr.isEmpty() || heightStr.isEmpty()) {
            Toast.makeText(this, "Please enter weight and height.",
Toast.LENGTH_SHORT).show();
            return;
        }

        double weight = Double.parseDouble(weightStr);
        double height = Double.parseDouble(heightStr);

        double bmi = weight / (height * height);
        bmiCategory = getBMICategory(bmi);


        bmiTextView.setText(String.format("Your BMI: %.2f", bmi));
        categoryTextView.setText("Category: " + bmiCategory);
    }

    private void save() {
        Button saveButton = findViewById(R.id.saveButton);
        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String fullName = fullNameEditText.getText().toString();
                int age = Integer.parseInt(ageEditText.getText().toString());
                int selectedRadioButtonId =
genderRadioGroup.getCheckedRadioButtonId();
                RadioButton selectedRadioButton =
findViewById(selectedRadioButtonId);
                String gender = selectedRadioButton.getText().toString();
                String weightStr = weightEditText.getText().toString();
                String heightStr = heightEditText.getText().toString();
                double weight = Double.parseDouble(weightStr);
                double height = Double.parseDouble(heightStr);

                sendDataToBackend(id, fullName, age,weight, gender, height,
bmiCategory);
//                saveDataToDatabase();
            }
        });
    }


    private void clearData() {
        fullNameEditText.setText("");
        ageEditText.setText("");
        weightEditText.setText("");
        heightEditText.setText("");
        bmiTextView.setText("");
        categoryTextView.setText("");
        genderRadioGroup.clearCheck();
    }

    private String getBMICategory(double bmi) {
        if (bmi < 18.5) {
            return "Underweight";
        } else if (bmi >= 18.5 && bmi < 25) {
            return "Normal weight";
        } else if (bmi >= 25 && bmi < 30) {
```

```
            return "Overweight";
        } else {
            return "Obese";
        }
    }

    private void sendDataToBackend(Long id,String fullName, int age, double
weight, String gender, double height, String bmiCategory) {
        // Create a RequestQueue instance
        RequestQueue requestQueue = Volley.newRequestQueue(this);

        // Create the URL for your backend API
        String url = "http://192.168.207.158:8096/api/bmi-records/adduser";

        // Create the request parameters as a JSONObject
        JSONObject jsonObject = new JSONObject();
        try {
            jsonObject.put("id", id);
            jsonObject.put("fullName", fullName);
            jsonObject.put("age", age);
            jsonObject.put("gender", gender);
            jsonObject.put("weight", weight);
            jsonObject.put("height", height);
            jsonObject.put("bmiCategory", bmiCategory);

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

        // Create the request
        JsonObjectRequest request = new
JsonObjectRequest(Request.Method.POST, url, jsonObject,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        // Handle the response from the server
                        Toast.makeText(MainActivity.this, "Data sent to
backend", Toast.LENGTH_SHORT).show();
                    }
                },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        // Handle the error response from the server
                        Toast.makeText(MainActivity.this, "Error sending data
to backend", Toast.LENGTH_SHORT).show();
                    }
                });

        // Add the request to the RequestQueue
        requestQueue.add(request);
    }

//    private void saveDataToDatabase() {
//        String fullName = fullNameEditText.getText().toString();
//        String ageStr = ageEditText.getText().toString();
//        int age = 0;
//
//        if (fullName.isEmpty() || ageStr.isEmpty()) {
//            Toast.makeText(this, "Please enter full name and age.",
Toast.LENGTH_SHORT).show();
```

```
//             return;
//         }
//
//         try {
//             age = Integer.parseInt(ageStr);
//         } catch (NumberFormatException e) {
//             Toast.makeText(this, "Invalid age value.",
Toast.LENGTH_SHORT).show();
//             return;
//         }
//
//         int selectedRadioButtonId =
genderRadioGroup.getCheckedRadioButtonId();
//         RadioButton selectedRadioButton =
findViewById(selectedRadioButtonId);
//         String gender = selectedRadioButton.getText().toString();
//
//         String weightStr = weightEditText.getText().toString();
//         String heightStr = heightEditText.getText().toString();
//
//         if (weightStr.isEmpty() || heightStr.isEmpty()) {
//             Toast.makeText(this, "Please calculate BMI first.",
Toast.LENGTH_SHORT).show();
//             return;
//         }
//
//         double weight = Double.parseDouble(weightStr);
//         double height = Double.parseDouble(heightStr);
//
//         long rowId = databaseHelper.insertRecord(fullName, age, gender,
weight, height, bmiCategory );
//
//         if (rowId != -1) {
//             Toast.makeText(this, "Data saved to database.",
Toast.LENGTH_SHORT).show();
//         } else {
//             Toast.makeText(this, "Failed to save data to database.",
Toast.LENGTH_SHORT).show();
//         }
//     }


}
```