

Energy Conservation Champion
BACHELOR OF
TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING
BY

T. Vimala	(22501A05I4)
P. Lavanya	(22501A05D4)
T. Revanth	(22501A05I5)
M. Sudeeksha	(22501A05H2)

Under the Guidance of
Mr. Michael Sadgun Rao Kona,
Assistant Professor



PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU: Kakinada, Approved by AICTE)

(An NBA & NAAC A++ accredited and ISO 9001:2015 certified institution)

Kanuru, Vijayawada-520007

2024-25

PRASAD V POTLURI

SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU :: Kakinada, Approved by AICTE)

(An NBA & NAAC A++ accredited and ISO 9001:2015 certified institution)

Kanuru, Vijayawada – 520007



CERTIFICATE

This is to certify that the project report title “**Energy Conservation Champion**” is the Bonafide work of **T. Vimala (22501A05I4)**, **P. Lavanya (22501A05D4)**, **T. Revanth (22501A05I5)**, **M. Sudeeksha (22501A05H2)** in partial fulfilment of completing the Academic project in Mobile App Development (20SA8651) during the academic year 2024-25.

Signature of the Incharge

Signature of the HOD

INDEX

S.No.	Contents	Page No. (s)
1	Abstract	1
2	Introduction	2
3	Objectives and Scope of the Project	3
4	Software used - Explanation	4-5
5	Proposed model	6-8
6	Sample Code	9-22
7	Result/Output Screenshots	23-29
8	Conclusion & Future Enhancements	30
9	References	31

1. ABSTRACT

Energy Conservation Champion is an Android application designed to help users monitor and optimize their energy consumption efficiently. The app allows users to track their household or business electricity usage by inputting appliance details, usage hours, and electricity rates. It provides real-time energy consumption calculations, cost estimation, and detailed breakdowns to promote energy-efficient practices.

The application features interactive dashboards, consumption analysis tools, and personalized insights to help users identify high-energy-consuming appliances and implement conservation strategies. Users can set weekly reminders to log their energy usage, ensuring consistent tracking and improved accuracy.

Developed using Android Studio with Java and backed by an SQL database, the app offers a seamless and structured approach to energy tracking. By providing transparent data visualization, real-time analytics, and energy-saving recommendations, Energy Conservation Champion empowers users to adopt responsible energy consumption habits and reduce their electricity costs effectively.



1.1. SDG JUSTIFICATION REPORT

SDG Mapped:

SDG 7 – Affordable and Clean Energy

SDG 11 – Sustainable Cities and Communities

SDG 12 – Responsible Consumption and Production

SDG 13 – Climate Action

SDG 15 – Life on Land

1.1.1 How This Project Supports SDGs 7, 11, 12, 13, and 15

The **Energy Consumption Tracker** is designed to empower individuals and communities by promoting energy efficiency, reducing waste, and fostering a culture of responsible consumption. By providing users with real-time insights and actionable recommendations, this project helps minimize environmental impact while contributing to a cleaner, more sustainable future.

SDG 7 – Affordable and Clean Energy

The project enables users to track and optimize energy use, reducing waste and promoting efficient consumption.

- **Promotes Energy Efficiency:** The tracker helps users monitor their energy consumption, encouraging efficient usage of appliances. By identifying high-energy-consuming devices, users can switch to energy-efficient alternatives like LED bulbs or inverter-based appliances.
- **Reduces Energy Waste:** Provides real-time insights on estimated vs. actual consumption, helping users minimize unnecessary energy use. Sends alerts when excessive energy is consumed, promoting conservation efforts.

SDG 11 – Sustainable Cities and Communities

By encouraging responsible energy habits, the project reduces urban energy demand and enhances sustainability efforts.

- **Supports Smart Energy Management:** Encourages individuals to adopt sustainable energy habits, reducing strain on city power grids. Helps in promoting decentralized and community-driven energy solutions like solar power.
- **Contributes to a Greener Urban Environment:** Lower energy consumption leads to reduced reliance on fossil fuels, cutting down air pollution in urban areas. Supports urban sustainability initiatives by making households more energy-conscious.

SDG 12 – Responsible Consumption and Production

The project fosters responsible energy use by educating users on their consumption patterns and promoting sustainable practices.

- **Encourages Sustainable Consumer Behaviour:** By providing detailed consumption patterns, the tracker makes users more conscious of their energy habits. Encourages switching to renewable energy sources, promoting responsible energy production.
- **Empowers Users with Data-Driven Insights:** Users can track weekly energy trends and adjust their consumption accordingly. Helps businesses and households reduce electricity bills while maintaining a sustainable lifestyle.

SDG 13 – Climate Action

By reducing energy consumption, the project directly contributes to lower carbon emissions and climate change mitigation.

- **Reduces Carbon Footprint:** Lower energy consumption means fewer greenhouse gas emissions, directly supporting climate change mitigation. Promotes awareness about how small household changes can have a significant impact on global energy conservation.
- **Encourages Proactive Environmental Action:** The alert system nudges users to take corrective measures if they exceed their estimated consumption. Promotes a culture of environmental responsibility by encouraging mindful energy use.

SDG 15 – Life on Land

The project promotes sustainable energy consumption habits, reducing environmental degradation and resource depletion.

- **Reduces Strain on Natural Resources:** By encouraging users to consume electricity more efficiently, the project helps lower overall energy demand. This, in turn, reduces the need for excessive energy production, which can lead to habitat destruction and pollution.
- **Minimizes Environmental Impact:** Less electricity consumption leads to lower emissions from power plants, reducing air and soil pollution. Promoting mindful energy use helps decrease the ecological footprint of individuals and households.

This comprehensive approach ensures that the project contributes meaningfully to multiple SDGs, fostering a more sustainable and resilient future.

2. INTRODUCTION

Energy Conservation Champion is an innovative Android-based mobile application designed to help individuals, households, and businesses effectively monitor, analyze, and optimize their energy consumption. With a sleek and intuitive user interface, the app empowers users to track electricity usage in real-time, estimate costs based on their consumption, and implement energy-efficient strategies to reduce waste. By providing structured insights through interactive dashboards, graphical representations, and detailed consumption breakdowns, the app simplifies energy management while encouraging sustainability.

One of the key features of Energy Conservation Champion is its ability to log appliance details and monitor usage patterns, allowing users to understand how different devices contribute to their overall electricity consumption. Through historical data analysis and personalized recommendations, users can make informed decisions to minimize energy costs and adopt eco-friendly habits. The app also includes smart reminders and notifications, ensuring that users regularly log their data and stay updated on their energy-saving progress.

Built using Android Studio with Java and supported by an SQL database, the app guarantees seamless performance, secure data storage, and a smooth user experience. The integration of real-time data tracking, automated insights, and cost estimations makes it an essential tool for anyone looking to take control of their electricity usage. Whether for personal use or business applications, Energy Conservation Champion provides a comprehensive and accessible solution for reducing energy bills, promoting sustainability, and contributing to a greener future.

3. OBJECTIVES AND SCOPE OF THE PROJECT

Energy Conservation Champion is designed to simplify energy tracking and management for individuals and businesses by providing a structured and interactive platform for monitoring consumption, estimating costs, and promoting energy-efficient practices. The project focuses on the following key objectives:

Monitor and Optimize Energy Consumption: Energy Conservation Champion provides a streamlined system for tracking and analyzing electricity usage. Users can input appliance details, usage hours, and electricity rates to estimate total consumption, enabling them to make data-driven decisions on reducing energy waste.

Provide Cost Estimation and Savings Insights: The platform calculates electricity costs based on usage patterns and customizable tariff rates. By displaying estimated expenses and high-energy-consuming appliances, the app helps users plan their energy consumption more efficiently and cut down on unnecessary costs.

Encourage Sustainable Energy Practices: The app fosters energy-conscious behavior by offering actionable recommendations for optimizing energy usage. It provides users with energy-saving tips, personalized insights, and visual analytics to encourage more responsible consumption habits.

Enhance User Engagement with Real-Time Data and Analytics: The application is designed with interactive dashboards and real-time data visualization, allowing users to track their energy usage trends effectively. Features such as pie charts, historical data analysis, and reminders ensure that users stay informed and proactive about their energy conservation efforts.

Enable Future Scalability and Integration: The platform is built with future scalability in mind, allowing enhancements such as smart device integration, AI-driven energy recommendations, and automated energy tracking. Additional features like monthly reports, community challenges, and energy comparison tools will help users improve their conservation efforts over time.

Scope of the Project: Energy Conservation Champion is more than just an energy tracking tool; it is a comprehensive energy management solution for individuals and businesses. By providing real-time tracking, personalized recommendations, and data-driven insights, the app helps users reduce energy waste, cut costs, and adopt sustainable practices. With an intuitive interface and structured analytics, it simplifies energy monitoring, making conservation more accessible, engaging, and effective.

4. SOFTWARE USED – EXPLANATION

Energy Conservation Champion is developed using modern Android technologies to provide a seamless and interactive energy management experience. The key technologies used include:

Frontend Technologies:

Android Studio

- Integrated development environment (IDE) used for building Android applications.
- Provides tools for UI design, debugging, and performance optimization.
- Supports real-time preview and emulator testing for efficient development.



Java

- Primary programming language used for Android app development.
- Provides object-oriented programming features for better code organization and maintainability.
- Ensures high performance and compatibility with various Android devices.



Backend Technologies:

Java (Servlets and APIs)

- Handles user authentication, data processing, and business logic.
- Manages communication between the frontend and database efficiently



Database and Storage Tools:

SQL Database

- Stores user data, appliance details, and energy consumption records.
- Provides structured and efficient data management for quick retrieval and analysis.
- Supports complex queries for generating consumption reports and tracking historical data.



By integrating these technologies, Energy Conservation Champion ensures a seamless and interactive experience, enabling users to efficiently monitor and optimize their energy consumption.

5. PROPOSED MODEL

The proposed model for Energy Conservation Champion aims to create an efficient and user-friendly platform for energy monitoring, cost estimation, and sustainable energy management. The system is structured into various key components:

Home Page & Navigation: The landing page provides an intuitive interface where users can input appliance details, track energy consumption, and access cost estimation features. Streamlined navigation ensures a smooth user experience, allowing users to interact effortlessly with the app's functionalities.

Energy Usage Tracking: The app enables users to log their electrical appliances, specifying power ratings and usage durations. It calculates energy consumption in real time, helping users understand their daily, weekly, and monthly usage patterns.

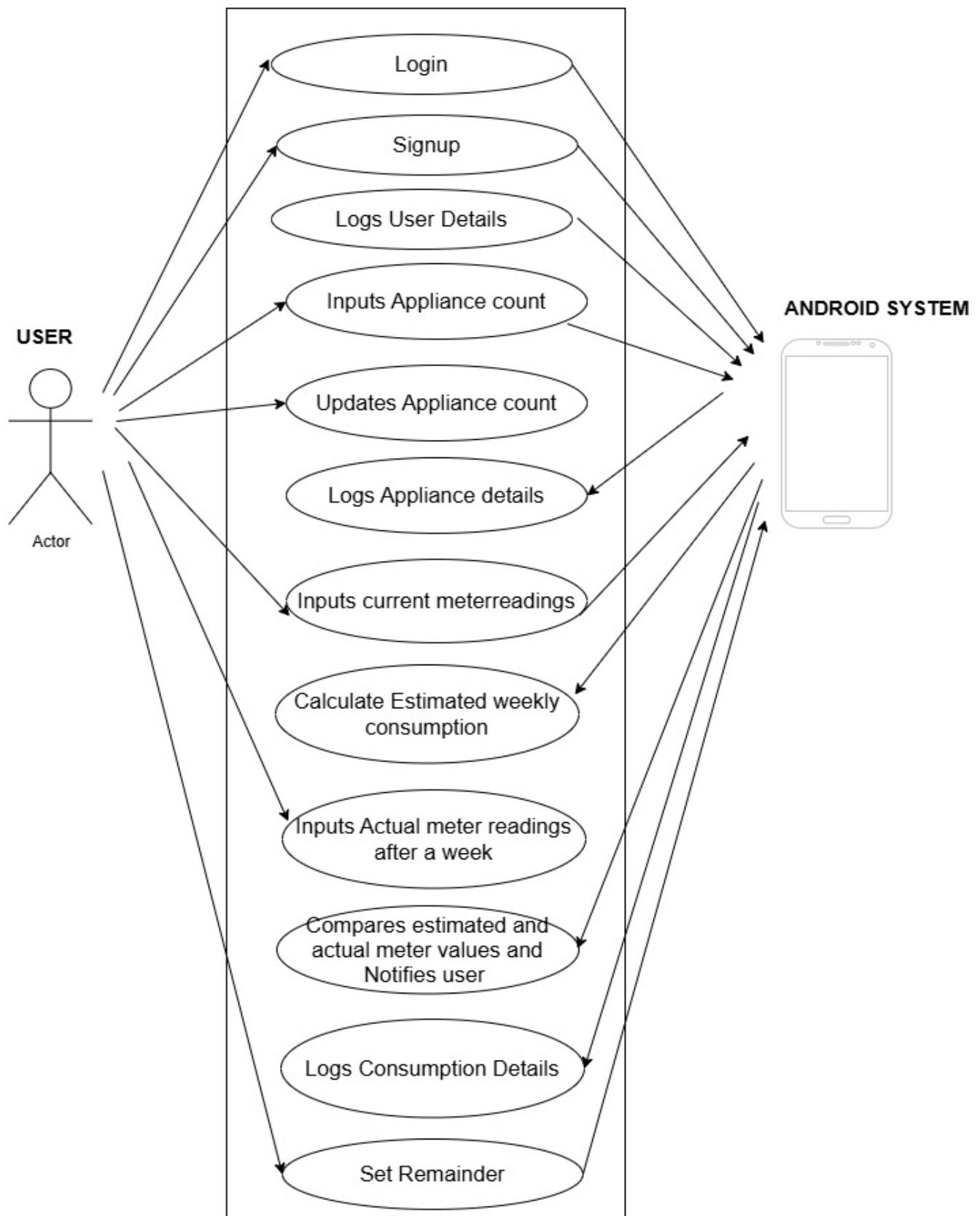
Cost Estimation & Savings Insights: The system estimates electricity costs based on the entered appliance details and customizable electricity rates. Users can analyze their expenses and receive recommendations on reducing unnecessary consumption to save money.

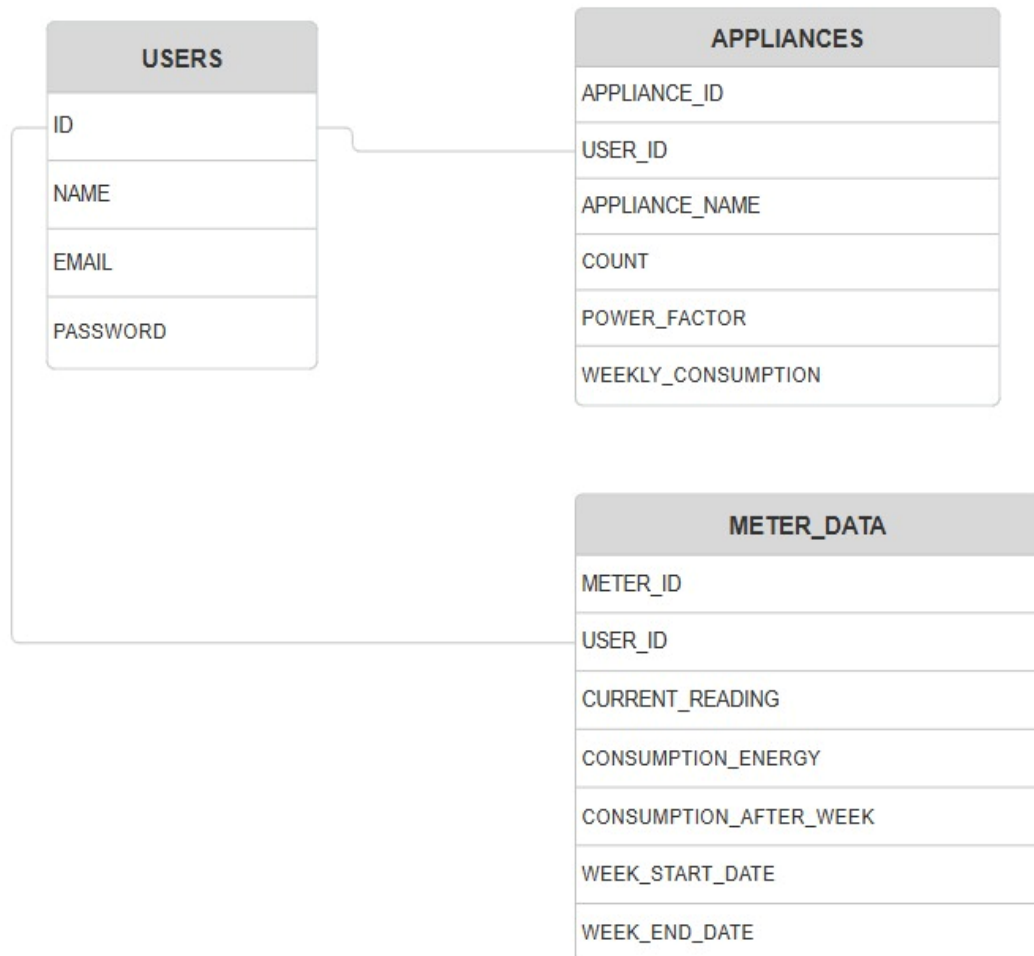
Consumption Analysis & Reports: Users can view detailed breakdowns of energy consumption through interactive charts and reports. High-consumption appliances are highlighted, allowing users to identify areas for improvement and optimize their energy usage.

Reminder & Notification System: The app allows users to set reminders for logging energy consumption, ensuring consistency in tracking and better accuracy in calculations. Notifications help users stay updated on energy-saving tips and personalized insights.

Data Storage & Security: The system securely stores user data, appliance details, and energy usage history in an SQL database. This ensures quick retrieval and structured documentation, enabling users to review past consumption trends.

Responsive Design & Scalability: The application is designed with a mobile-first approach, ensuring smooth performance on Android devices. Future enhancements could include AI-powered recommendations, smart device integration for automated tracking, and community-based challenges to encourage energy conservation.

Use Case Diagram:**Energy Calculator App**

Database Schema Diagram:**ENERGY_CONSERVATION CHAMPION_DB**

6. SAMPLE CODE

GitHub repository links:

http://github.com/Sudeeksha29/Energy_Consevation_Champion_mad

Folder Structure:

```
+---app
| | .gitignore
| | build.gradle.kts
| | DatabaseHelper.java
| | proguard-rules.pro
| |
| \---src
|   +---androidTest
|   |   \---java
|   |       \---com
|   |           \---example
|   |               \---energyapp
|   |                   ExampleInstrumentedTest.java
|   |
|   +---main
|   |   |   AndroidManifest.xml
|   |   |
|   |   +---java
|   |   |   \---com
|   |   |       \---example
|   |   |           \---energyapp
|   |   |               DatabaseHelper.java
|   |   |               InsertActivity.java
|   |   |               MainActivity.java
|   |   |               MeterdataActivity.java
|   |   |               SignupActivity.java
|   |   |               UpdateActivity.java
|   |   |               UserActivity.java
|   |   |
|   |   \---res
|   |       +---drawable
```

```
| | | ic_launcher_background.xml
| | | ic_launcher_foreground.xml
| | |
| | +---layout
| | | activity_insert.xml
| | | activity_main.xml
| | | activity_meterdata.xml
| | | activity_signup.xml
| | | activity_update.xml
| | | activity_user.xml
| | |
| | +---values
| | | colors.xml
| | | strings.xml
| | | themes.xml
| | |
| | +---values-night
| | | themes.xml
| | |
| | \---xml
| | | backup_rules.xml
| | | data_extraction_rules.xml
| | |
| \---test
| | \---java
| | | \---com
| | | | \---example
| | | | | \---energyapp
| | | | | | ExampleUnitTest.java
```


MainActivity.java

```
package com.example.energyapp;

import android.content.Intent;
import android.os.Bundle;
import android.util.Patterns;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    EditText email, password;
    Button btnLogin;
    TextView txtSignup;
    DatabaseHelper db;

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

        // Initialize views
        email = findViewById(R.id.emailLogin);
        password = findViewById(R.id.passwordLogin);
        btnLogin = findViewById(R.id.btnLogin);
        txtSignup = findViewById(R.id.txtSignup);

        // Initialize SQLite Database Helper
        db = new DatabaseHelper(this);

        // Login Button Click Listener
        btnLogin.setOnClickListener(v -> {
            String userEmail = email.getText().toString().trim();
            String userPassword = password.getText().toString().trim();

            // Validate Input Fields
```

```
if (userEmail.isEmpty() || userPassword.isEmpty()) {
    Toast.makeText(MainActivity.this, "Please enter email and password",
Toast.LENGTH_SHORT).show();
} else if (!Patterns.EMAIL_ADDRESS.matcher(userEmail).matches()) {
    Toast.makeText(MainActivity.this, "Enter a valid email", Toast.LENGTH_SHORT).show();
} else {
    boolean isUserValid = db.checkUser(userEmail, userPassword);
    if (isUserValid) {
        Toast.makeText(MainActivity.this, "Login Successful", Toast.LENGTH_SHORT).show();

        // Navigate to User Dashboard
        Intent intent = new Intent(MainActivity.this, UserActivity.class);
        startActivity(intent);
        finish(); // Prevent user from going back to login screen
    } else {
        Toast.makeText(MainActivity.this, "Invalid Credentials", Toast.LENGTH_SHORT).show();
    }
}
});

// Redirect to Signup Page
txtSignup.setOnClickListener(v -> {
    Intent intent = new Intent(MainActivity.this, SignupActivity.class);
    startActivity(intent);
});
}
}
```

MainActivity.xml

```
<?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:gravity="center"
    android:orientation="vertical"
    android:padding="20dp"
    android:background="#F9F9F9">

    <EditText
        android:id="@+id/emailLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress" />

    <EditText
        android:id="@+id/passwordLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/btnLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Login" />

    <TextView
        android:id="@+id/txtSignup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Don't have an account? Sign Up"
        android:textColor="@android:color/holo_blue_dark"
        android:paddingTop="10dp"/>
</LinearLayout>
```

InsertActivity.java

```
package com.example.energyapp;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;

public class InsertActivity extends AppCompatActivity {
    DatabaseHelper dbHelper;
    int userId;
    EditText meterReadingInput, consumptionValueInput;
    Button calculateButton, nextButton;

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

        dbHelper = new DatabaseHelper(this);
        userId = getUserId(); // Retrieve user ID dynamically

        Button insertButton = findViewById(R.id.insert_button);
        insertButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                insertApplianceData();
            }
        });
    }
}
```

```
meterReadingInput = findViewById(R.id.meter_reading);
consumptionValueInput = findViewById(R.id.consumption_value);
calculateButton = findViewById(R.id.calculate_button);
nextButton = findViewById(R.id.next_button);
```

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

```
nextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        saveMeterData();
    }
});
```

```
}
```

```
private int getUserId() {
    SharedPreferences sharedPreferences = getSharedPreferences("UserPrefs", Context.MODE_PRIVATE);
    return sharedPreferences.getInt("userId", -1); // Default -1 if no user is found
}
```

```
private void insertApplianceData() {
    insertAppliance("Fans", R.id.fans, 0.75);
    insertAppliance("Fridge", R.id.fridge, 1.2);
    insertAppliance("AC", R.id.ac, 3.5);
    insertAppliance("Lights", R.id.lights, 0.5);
    insertAppliance("Oven", R.id.oven, 2.0);
    insertAppliance("TV", R.id.tv, 0.2);
    insertAppliance("Bulbs", R.id.bulbs, 0.1);
    insertAppliance("Cooler", R.id.cooler, 1.5);
    insertAppliance("Water Filter", R.id.water_filter, 0.3);
    insertAppliance("Chargers", R.id.chargers, 0.05);
    insertAppliance("Inverter", R.id.inverter, 1.0);
    insertAppliance("Router", R.id.router, 0.1);
}
```

```
insertAppliance("Computer", R.id.computer, 0.4);
insertAppliance("Heater", R.id.heater, 2.5);
insertAppliance("Washing Machine", R.id.washing_machine, 1.8);

Toast.makeText(this, "Data Inserted Successfully!", Toast.LENGTH_SHORT).show();
}

private void insertAppliance(String applianceName, int editTextId, double powerFactor) {
    EditText editText = findViewById(editTextId);
    String countStr = editText.getText().toString();
    int count = countStr.isEmpty() ? 0 : Integer.parseInt(countStr);

    if (count > 0) {
        dbHelper.insertAppliance(userId, applianceName, count, powerFactor);
    }
}

private void calculateEnergyConsumption() {
    double currentMeterReading = Double.parseDouble(meterReadingInput.getText().toString());
    double weeklyConsumption = dbHelper.getTotalWeeklyConsumption(userId);
    double totalConsumption = currentMeterReading + weeklyConsumption;

    consumptionValueInput.setText(String.valueOf(totalConsumption));
    Toast.makeText(this, "Consumption Energy Updated", Toast.LENGTH_SHORT).show();
}

private void saveMeterData() {
    try {
        if (userId == -1) {
            Toast.makeText(this, "Invalid User ID. Please log in again.", Toast.LENGTH_SHORT).show();
            return;
        }

        String meterReadingStr = meterReadingInput.getText().toString();
        String consumptionStr = consumptionValueInput.getText().toString();

        if (meterReadingStr.isEmpty() || consumptionStr.isEmpty()) {
            Toast.makeText(this, "Please enter both meter reading and consumption values",
```

```
Toast.LENGTH_SHORT).show();
    return;
}

double currentMeterReading = Double.parseDouble(meterReadingStr);
double consumptionEnergy = Double.parseDouble(consumptionStr);

if (currentMeterReading < 0 || consumptionEnergy < 0) {
    Toast.makeText(this, "Values cannot be negative", Toast.LENGTH_SHORT).show();
    return;
}

Calendar calendar = Calendar.getInstance();
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
String startDate = dateFormat.format(calendar.getTime());
calendar.add(Calendar.DAY_OF_YEAR, 7);
String endDate = dateFormat.format(calendar.getTime());

boolean success = dbHelper.insertMeterData(userId, currentMeterReading, consumptionEnergy, startDate,
endDate);
if (success) {
    Toast.makeText(this, "Meter Data Saved Successfully!", Toast.LENGTH_SHORT).show();
    Intent intent = new Intent(InsertActivity.this, MeterdataActivity.class);
    startActivity(intent);
} else {
    Toast.makeText(this, "Failed to save meter data. Please try again.", Toast.LENGTH_SHORT).show();
}
} catch (NumberFormatException e) {
    Toast.makeText(this, "Invalid input! Please enter valid numeric values.",
Toast.LENGTH_SHORT).show();
} catch (Exception e) {
    e.printStackTrace();
    Toast.makeText(this, "An error occurred while saving data", Toast.LENGTH_SHORT).show();
}
}
}
```

UpdateActivity.java

```
package com.example.energyapp;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Locale;

public class UpdateActivity extends AppCompatActivity {
    DatabaseHelper dbHelper;
    EditText meterReadingInput, consumptionValueInput;
    Button calculateButton, nextButton;
    int userId = 1; // Replace with actual logged-in user ID
    HashMap<String, Integer> applianceIds = new HashMap<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update); // Ensure you use the correct layout

        dbHelper = new DatabaseHelper(this);
        loadApplianceData(); // Fetch and display existing values

        Button updateButton = findViewById(R.id.update_button);
        updateButton.setText("UPDATE DATA"); // Set button text
        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
```



```
public void onClick(View v) {
    updateApplianceData(); // Update data in the database
    moveToMeterData(); // Move to MeterData activity
}
});

meterReadingInput = findViewById(R.id.meter_reading);
consumptionValueInput = findViewById(R.id.consumption_value);
calculateButton = findViewById(R.id.calculate_button);
nextButton = findViewById(R.id.next_button);

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

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

private int getUserId() {
    SharedPreferences sharedPreferences = getSharedPreferences("UserPrefs", Context.MODE_PRIVATE);
    return sharedPreferences.getInt("userId", -1); // Default -1 if no user is found
}

private void loadApplianceData() {
    Cursor cursor = dbHelper.getApplianceData(userId);
    if (cursor != null) {
        while (cursor.moveToNext()) {
            int applianceId = cursor.getInt(0);
            String applianceName = cursor.getString(1);
        }
    }
}
```

```
        int count = cursor.getInt(2);

        applianceIds.put(applianceName, applianceId);
        setEditTextValue(applianceName, count);
    }
    cursor.close();
}

private void setEditTextValue(String applianceName, int count) {
    int editTextId = getApplianceEditTextId(applianceName);
    if (editTextId != -1) {
        EditText editText = findViewById(editTextId);
        editText.setText(String.valueOf(count));
    }
}

private int getApplianceEditTextId(String applianceName) {
    switch (applianceName) {
        case "Fans": return R.id.fans;
        case "Fridge": return R.id.fridge;
        case "AC": return R.id.ac;
        case "Lights": return R.id.lights;
        case "Oven": return R.id.oven;
        case "TV": return R.id.tv;
        case "Bulbs": return R.id.bulbs;
        case "Cooler": return R.id.cooler;
        case "Water Filter": return R.id.water_filter;
        case "Chargers": return R.id.chargers;
        case "Inverter": return R.id.inverter;
        case "Router": return R.id.router;
        case "Computer": return R.id.computer;
        case "Heater": return R.id.heater;
        case "Washing Machine": return R.id.washing_machine;
        default: return -1;
    }
}
```

```
private void updateApplianceData() {
    for (String applianceName : applianceIds.keySet()) {
        int applianceId = applianceIds.get(applianceName);
        int editTextId = getApplianceEditTextId(applianceName);
        if (editTextId != -1) {
            EditText editText = findViewById(editTextId);
            String countStr = editText.getText().toString();
            int count = countStr.isEmpty() ? 0 : Integer.parseInt(countStr);

            dbHelper.updateAppliance(applianceId, count);
        }
    }
    Toast.makeText(this, "Data Updated Successfully!", Toast.LENGTH_SHORT).show();
}

private void moveToMeterData() {
    Intent intent = new Intent(UpdateActivity.this, MeterdataActivity.class);
    startActivity(intent);
    finish(); // Close the update activity
}

private void calculateEnergyConsumption() {
    double currentMeterReading = Double.parseDouble(meterReadingInput.getText().toString());
    double weeklyConsumption = dbHelper.getTotalWeeklyConsumption(userId);
    double totalConsumption = currentMeterReading + weeklyConsumption;

    consumptionValueInput.setText(String.valueOf(totalConsumption));
    Toast.makeText(this, "Consumption Energy Updated", Toast.LENGTH_SHORT).show();
}

private void saveMeterData() {
    try {
        if (userId == -1) {
            Toast.makeText(this, "Invalid User ID. Please log in again.", Toast.LENGTH_SHORT).show();
            return;
        }

        String meterReadingStr = meterReadingInput.getText().toString();
        String consumptionStr = consumptionValueInput.getText().toString();
    }
}
```

```
        if (meterReadingStr.isEmpty() || consumptionStr.isEmpty()) {
            Toast.makeText(this, "Please enter both meter reading and consumption values",
Toast.LENGTH_SHORT).show();
            return;
        }

        double currentMeterReading = Double.parseDouble(meterReadingStr);
        double consumptionEnergy = Double.parseDouble(consumptionStr);

        if (currentMeterReading < 0 || consumptionEnergy < 0) {
            Toast.makeText(this, "Values cannot be negative", Toast.LENGTH_SHORT).show();
            return;
        }

        Calendar calendar = Calendar.getInstance();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
        String startDate = dateFormat.format(calendar.getTime());
        calendar.add(Calendar.DAY_OF_YEAR, 7);
        String endDate = dateFormat.format(calendar.getTime());

        boolean success = dbHelper.insertMeterData(userId, currentMeterReading, consumptionEnergy, startDate,
endDate);
        if (success) {
            Toast.makeText(this, "Meter Data Saved Successfully!", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(UpdateActivity.this, MeterdataActivity.class);
            startActivity(intent);
        } else {
            Toast.makeText(this, "Failed to save meter data. Please try again.", Toast.LENGTH_SHORT).show();
        }
    } catch (NumberFormatException e) {
        Toast.makeText(this, "Invalid input! Please enter valid numeric values.",
Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        e.printStackTrace();
        Toast.makeText(this, "An error occurred while saving data", Toast.LENGTH_SHORT).show();
    }
}
```

7. RESULT/OUTPUT SCREENS

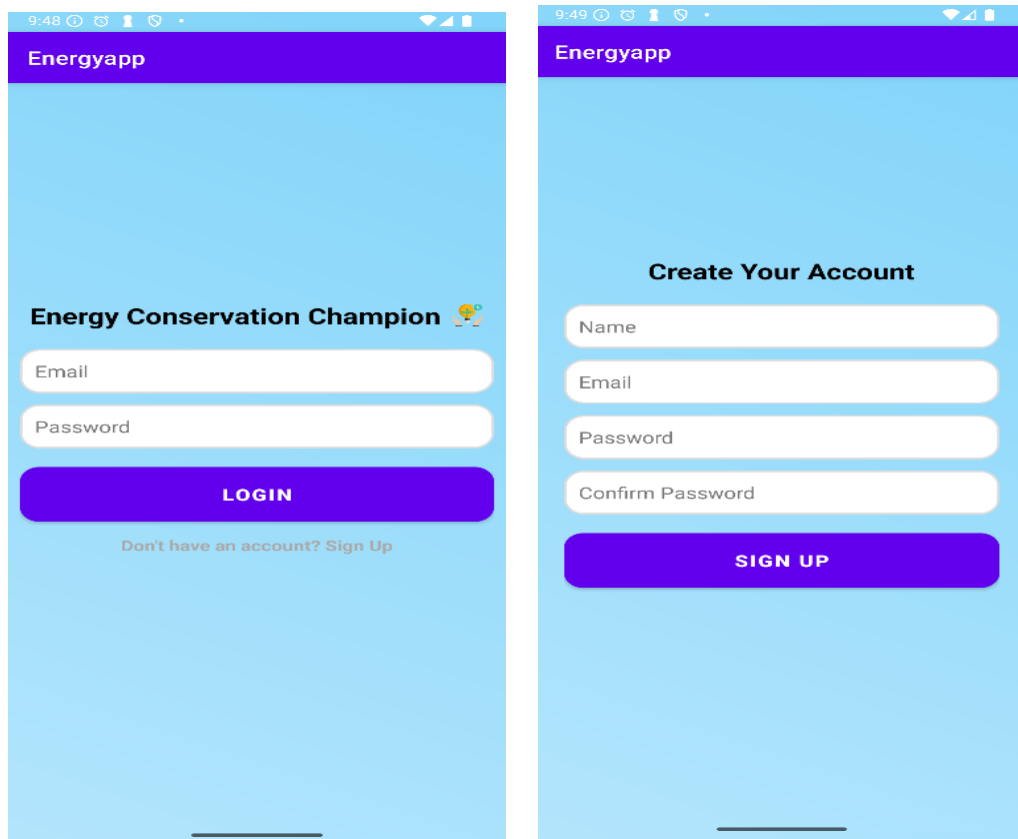


Figure 1: Sign Up and Login - These are signup and login pages. The signup page allows users to create an account and use the app, while the login page with password authentication.

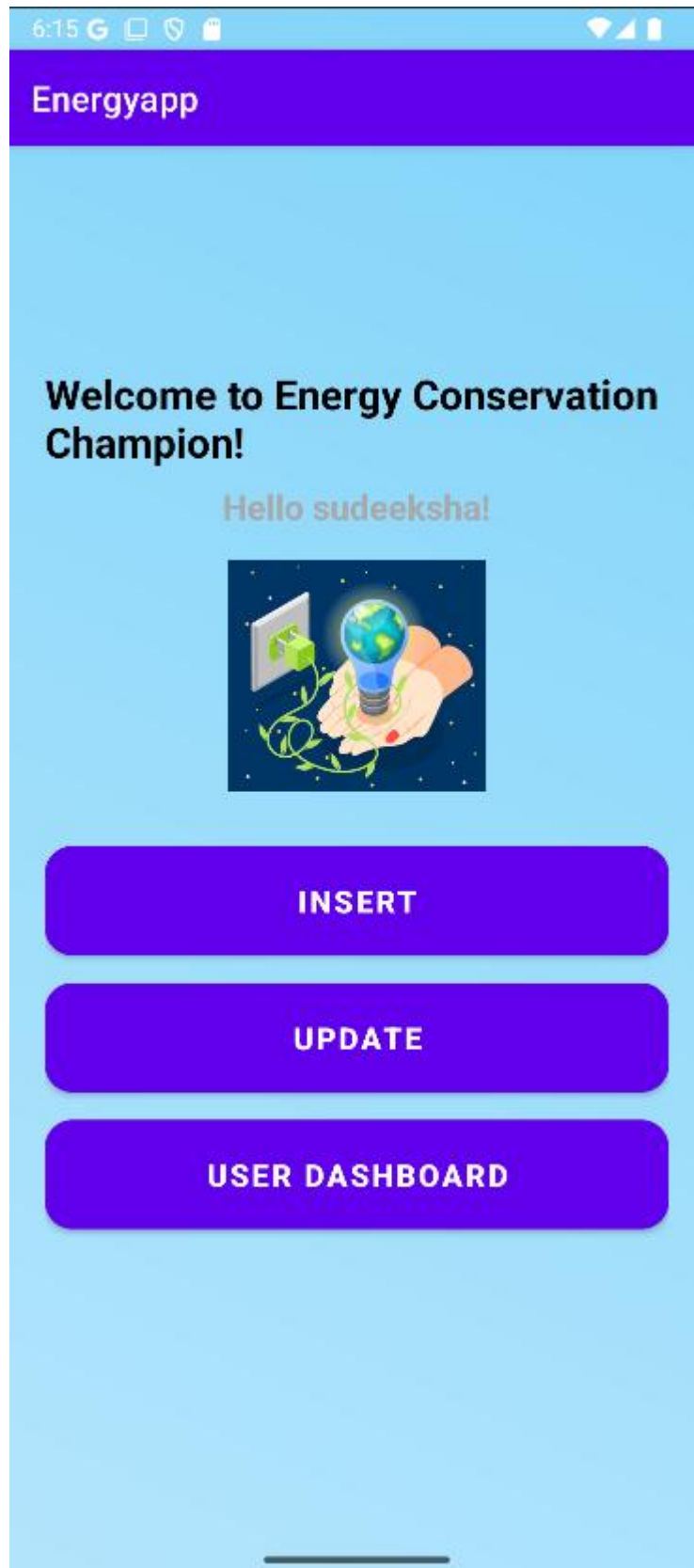


Figure 2: Home Page: This is the home page where users can navigate to different sections like inserting, updating, and viewing the user dashboard.

The figure displays two screenshots of the 'Energyapp' interface, which is used for managing energy consumption data. Both screens feature a purple header with the app name 'Energyapp' and a light blue background.

Left Screenshot (Insert Data): This screen allows users to input the number of electrical appliances. The form includes input fields for AC (2), Lights (5), Oven (1), TV (2), Bulbs (5), Cooler (2), Water Filter (1), Chargers (4), Inverter (2), Router (1), Computer (2), Heater (2), and Washing Machine (1). Below these fields is a purple button labeled 'INSERT DATA'. A 'Current meter reading' field with the placeholder 'Enter the number' and a purple 'CALCULATE' button are also present. At the bottom, there is a 'Can Consume:' field and a large purple 'NEXT' button. A success message 'Data Inserted Successfully!' is displayed at the bottom.

Right Screenshot (Update Data): This screen allows users to update the data. It features the same form as the left screenshot, but with a purple button labeled 'UPDATE DATA' instead of 'INSERT DATA'. The success message at the bottom reads 'Data Updated Successfully!'.

Figure 3: Energy Consumption Page: This page allows users to input the number of electrical appliances, calculate energy consumption based on the current meter reading, and insert data or update data . A success message confirms on data insertion and Updation.

The screenshot displays the 'Energyapp' interface on a mobile device. The app has a purple header with the title 'Energyapp'. Below the header, there is a light blue background with a grid of input fields for various electrical appliances. The inputs are as follows:

Appliance	Count	Appliance	Count
AC:	2	Lights:	5
Oven:	1	TV:	2
Bulbs:	5	Cooler:	2
Water Filter:	1	Chargers:	4
Inverter:	2	Router:	1
Computer:	2	Heater:	2
Washing Machine:	1		

Below the input fields is a purple button labeled 'INSERT DATA'. Underneath this button is a text field for 'Current meter reading' with the value '120'. Below the text field is another purple button labeled 'CALCULATE'. Below the 'CALCULATE' button, the result 'Can Consume: 461.6' is displayed in red text. Below the result is a wide purple button labeled 'NEXT'. At the bottom of the screen, there is a white notification bubble with a green icon and the text 'Consumption Energy Updated'. The status bar at the top shows the time as 9:52 and various system icons.

Figure 4: Energy Calculation Page: Users input the number of electrical appliances and the current meter reading, then calculate their energy consumption. The result is displayed, and a confirmation message appears upon successful update.

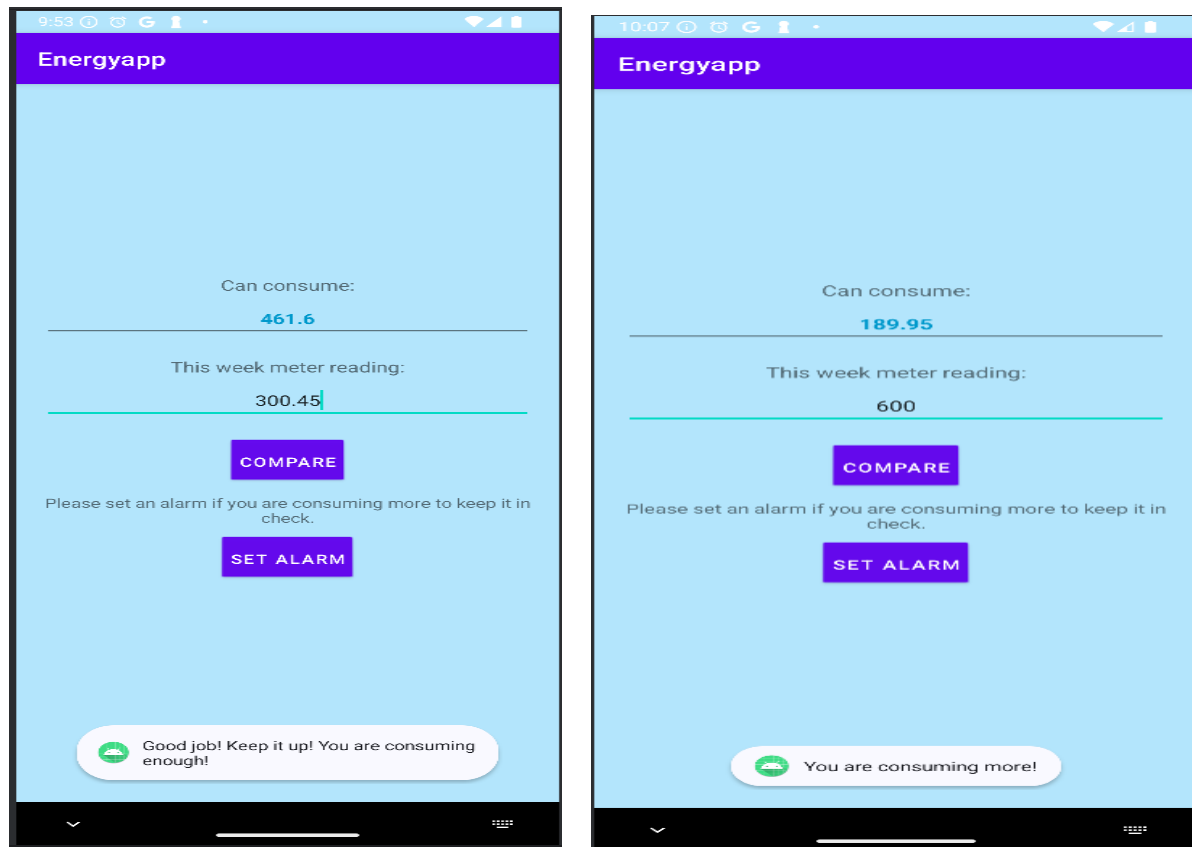
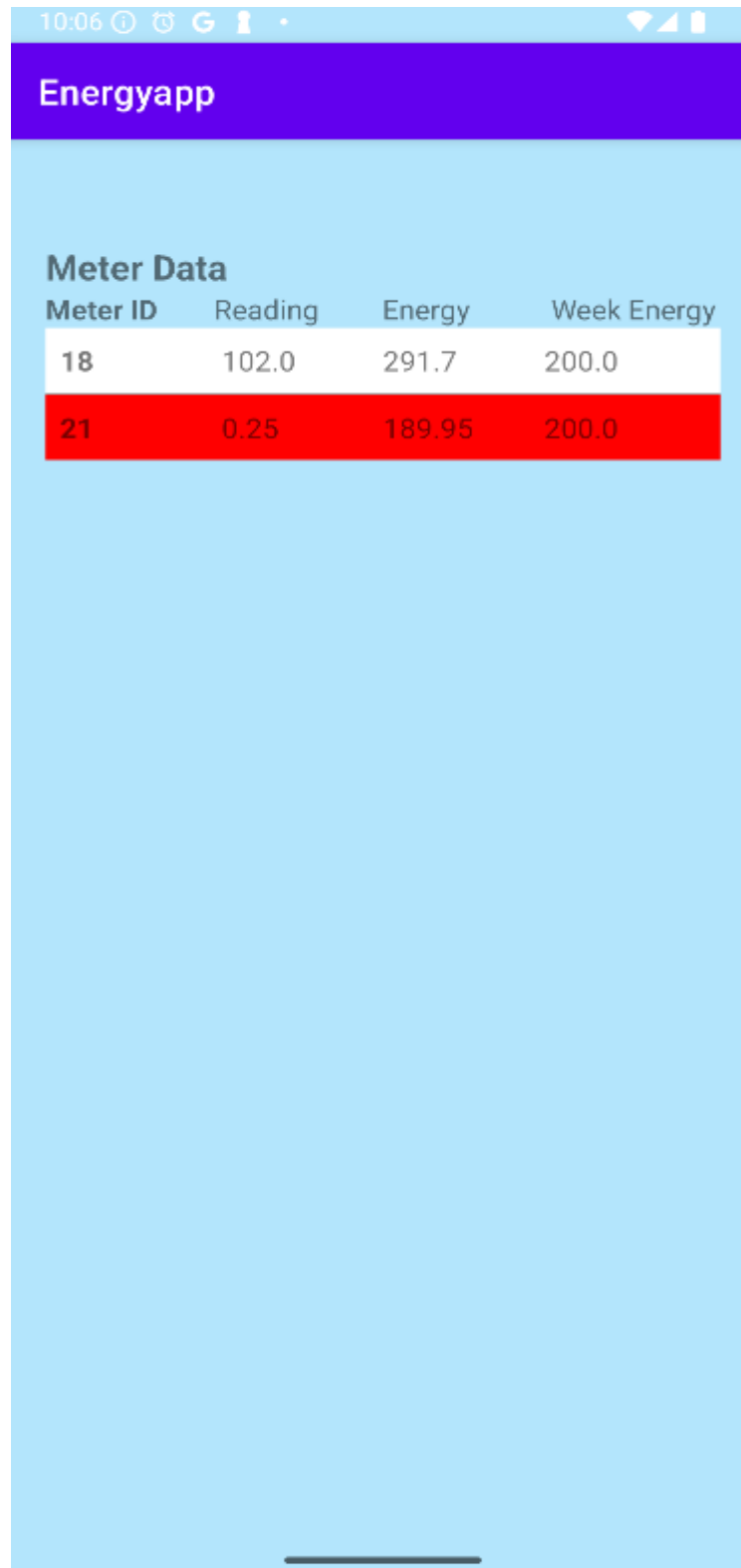


Figure 5: This screens lets users compare energy usage, set an alarm and confirms saved meter data



The screenshot shows a mobile application interface for 'Energyapp'. At the top, there is a purple header bar with the app name. Below it, a light blue background contains a table titled 'Meter Data'. The table has four columns: 'Meter ID', 'Reading', 'Energy', and 'Week Energy'. There are two data rows. The first row, with Meter ID 18, has a white background. The second row, with Meter ID 21, has a red background, indicating high energy consumption. The status bar at the top shows the time as 10:06 and various system icons.

Meter ID	Reading	Energy	Week Energy
18	102.0	291.7	200.0
21	0.25	189.95	200.0

Figure 6: When you click on "User Dashboard," this activity is displayed. The table highlights energy consumption data, with rows turning red if the consumption exceeds the estimated value, while they remain white if the usage is within the expected range.

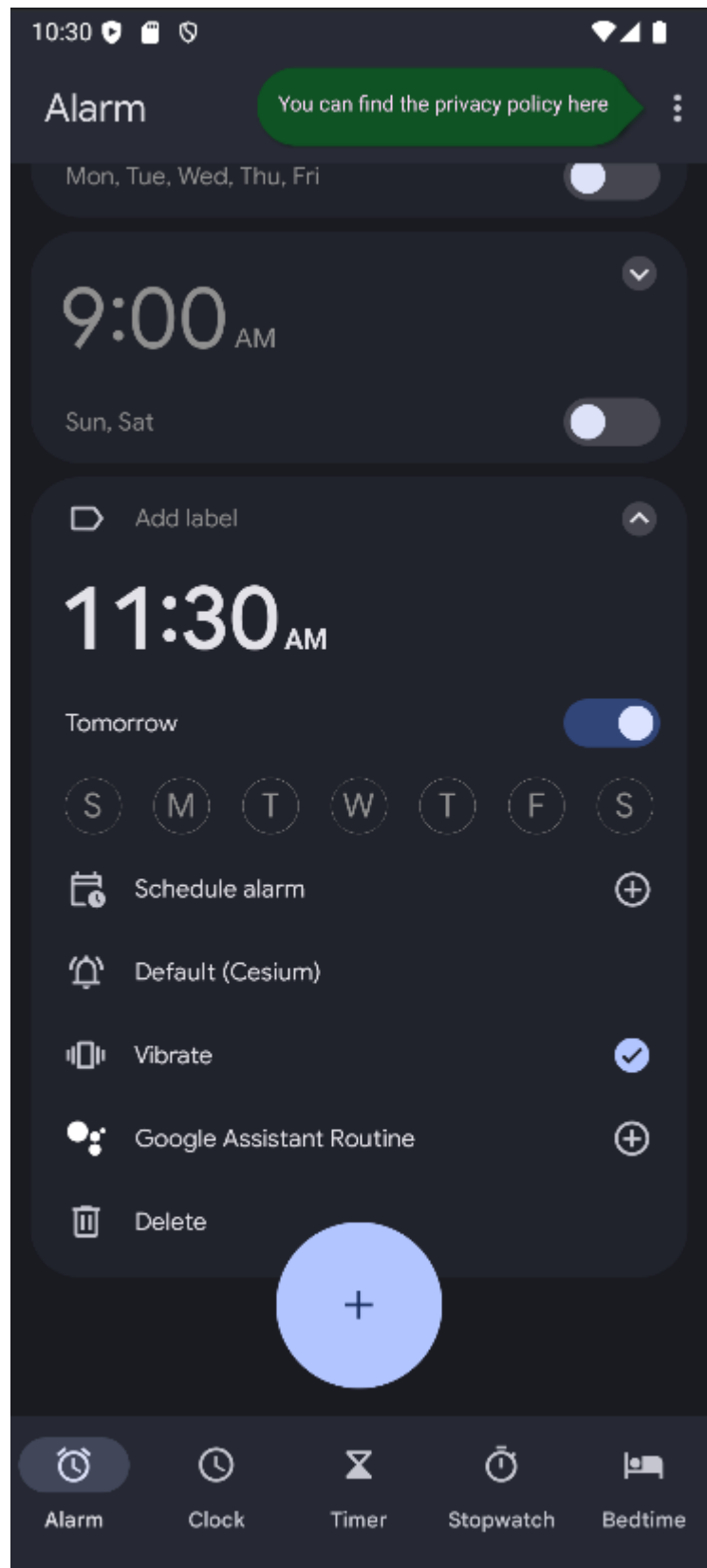


Figure 7: This screen displays a scheduled alarm

8. CONCLUSION & FUTURE ENHANCEMENTS

Energy Conservation Champion is a seamless and efficient energy management application designed to simplify energy tracking and optimization. By offering a user-friendly interface and robust functionality, it enables users to monitor energy consumption, estimate costs, and adopt energy-efficient practices effortlessly. The system ensures accurate data analysis and actionable insights for individuals and businesses.

Built with Android Studio using Java and an SQL database, Energy Conservation Champion delivers a highly responsive and scalable experience. The streamlined energy tracking enhances efficiency, making it an ideal solution for households, offices, and industries aiming to reduce electricity costs and minimize energy waste.

Future enhancements will focus on smart device integration, AI-driven energy-saving recommendations, real-time energy consumption alerts, and cloud-based data storage for enhanced accessibility. Features like push notifications for reminders, historical consumption trends, and interactive reports will further improve usability. These improvements will position Energy Conservation Champion as a powerful tool for sustainable energy management.

9. REFERENCES (WEB SITE URLS)

GitHub repository links:

https://github.com/Sudeeksha29/Energy_Conversation_Champion_mad

- **Android Studio Documentation** - <https://developer.android.com/studio>
- **Java Documentation** - <https://docs.oracle.com/en/java/>
- **SQL Database Documentation** - <https://www.mysql.com/doc/>
- **Android Jetpack Components** - <https://developer.android.com/jetpack>
- **Energy Efficiency Guidelines** - <https://www.energy.gov/>