A Project Report

On

The Track Master

Submitted in partial fulfillment of the

requirement for the award of the degree of

MASTER OF COMPUTER APPLICATION

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

MCA

Session 2023-2024
in

MCA

By

Shashikant Kumar Sharma(23SCSE2150001)
Philis Felistas Muthamba(23SCSE2140043)

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY

GALGOTIAS UNIVERSITY, GREATER NOIDA

INDIA

JUNE, 2024

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY

GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We, Shashikant Sharma and Philis Felistas Muthamba a student of MCA at Galgotias University, School of Computing and Information Technology (CNCS), hereby declare that the Android application titled **" The Track Master "** has been developed as part of my academic coursework under the guidance of Dr. Lalit Sharma. This project is a culmination of my original work and has not been previously submitted for any degree or examination.

Shashikant Kumar Sharma(23SCSE2150001)

Philis Felistas Mutamba (23SCSE2140043)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Lalit Sharma

TABLE OF CONTENTS

Abstract:

The "Track Master" Android application is a versatile tool designed to help users build, monitor, and maintain habits by tracking daily routines and personal goals. This application allows users to log daily activities, set personal goals, and track their progress across various categories, including walking, calorie intake, hydration, yoga, sleep, and reading.

Upon logging in, users are greeted with a personalized message and presented with their current habit data. The app features intuitive increment buttons to facilitate easy logging of daily achievements and includes checkboxes for users to visually confirm their goal completion. Users can also access a progress report to review their performance over time, allowing for informed adjustments to their habits.

The application utilizes a robust backend database to securely store user information and habit data, ensuring a seamless experience. The Habit Tracker promotes accountability and motivation, empowering users to cultivate positive habits that enhance their overall well-being.

Developed under the guidance of Dr. Lalit Sharma as part of the Master of Computer Applications (MCA) program at Galgotias University, this project aims to provide a user-friendly platform for personal growth and habit management. Whether for students, professionals, or individuals striving to adopt new habits, the app is designed to be a valuable companion in achieving personal development goals efficiently.

Introduction

The "Track Master" Android application is designed to help users effectively build, track, and manage their habits and daily routines. It serves as a personal development tool, enabling users to cultivate positive habits while monitoring their progress over time. Key functionalities include:

- **Habit Creation**: Users can create personalized habits, set start dates, and define how often they want to track the habits (daily, weekly, or on specific days).
- **Habit Tracking**: The app tracks each habit's progress, showing streaks of consecutive completions and highlighting milestones achieved.
- **Progress Analytics**: Detailed reports and charts offer visual insights into habit performance over time, helping users understand their consistency and adjust goals accordingly.
- **Calendar View**: A built-in calendar allows users to monitor their habit activity and see a daily, weekly, or monthly overview of their progress.
- **Goal Setting**: Users can set short-term and long-term goals to stay motivated and work towards larger objectives.
- **Dark Mode**: The app includes a dark mode option to enhance usability and reduce eye strain, particularly during evening use.

Tools:

- **Progress Statistics**: Offers statistics like total completions, success rates, and other metrics to help users analyze their habit-building journey.
- **Backup and Sync**: Ensures user data is safely backed up and can be synced across devices for uninterrupted tracking.

The goal of this project is to design and develop a habit tracking application, **Track Master**, that provides users with an efficient, intuitive platform to establish and maintain good habits. By integrating reminders, motivational tools, and progress tracking features, the app aims to help users achieve personal growth, develop positive routines, and reach their goals with ease.

This app caters to users seeking to improve their time management and productivity, providing an easy-to-use interface suitable for both beginners and experienced habit trackers.

Literature Survey:

The development of the **Track Master** habit-tracking application is informed by research in personal productivity, behavior change, and habit formation, leveraging modern mobile app technologies to assist users in achieving their personal goals. The key concepts drawn from existing literature include:

1. **Habit Formation and Behavior Change**: Extensive research, including works by behavioral psychologists like B.J. Fogg and James Clear, explores how habits are formed and maintained through cues, routines, and rewards. These studies emphasize the importance of tracking progress and receiving feedback, which are core features of Track Master.
2. **Gamification and Motivation**: Gamification principles, such as the use of streaks, milestones, and progress bars, have been well-researched for their effectiveness in motivating users to maintain consistent habits. Studies have shown that visual feedback and rewards increase user engagement and persistence in achieving goals.
3. **Mobile Health and Wellness Applications**: Research into wellness apps such as Fabulous, Habitica, and Strides reveals the growing trend of mobile applications that support users in tracking daily routines and achieving personal goals. These apps emphasize simplicity and customization, allowing users to set personalized habits and view their progress through user-friendly interfaces.
4. **Reminders and Notifications**: Studies on behavior prompts show that well-timed notifications and reminders can significantly influence habit completion and user engagement. Research into notification systems in mobile applications highlights the balance needed between helpful reminders and over-notification, ensuring user retention without causing fatigue.
5. **Data Visualization and Progress Tracking**: Research on data visualization demonstrates that presenting progress visually, through charts and graphs, enhances users' understanding of their performance over time. Studies suggest that visual feedback improves habit formation by making progress more tangible and motivating users to maintain their routines.
6. **User-Centered Design and Accessibility**: In the field of mobile app development, user-centered design principles are critical for ensuring the application is easy to use for a wide range of users. Research on accessibility and UX design informs Track Master’s interface, ensuring that it is simple and intuitive, allowing users to track habits efficiently.

Through this literature survey, **Track Master** integrates established psychological theories of habit formation with mobile app technology, offering an accessible and user-friendly platform for personal development. The app enhances productivity by combining habit-tracking methodologies with motivational tools, helping users stay consistent in their efforts toward building better habits.

Proposed Methodology:

1. Requirement Analysis:

- Define Essential Features: Identify the key features based on user needs, such as the ability to add, edit, and delete habits, track progress with visual feedback (streaks, charts), set reminders, and analyze habit performance through reports.
- Research APIs: Explore Android APIs for scheduling notifications (e.g., AlarmManager, WorkManager) and managing local data storage (e.g., Room database for habit tracking history) to ensure seamless data persistence and user interaction.

2. System Design:
    - Modular Architecture: Develop a modular system where each feature (habit creation, reminders, progress tracking, etc.) is designed independently to ensure flexibility and scalability.
    - User Interface (UI): Design the UI following Android's Material Design principles to ensure the app is visually appealing and easy to use. Use a tabbed layout or bottom navigation to organize features efficiently, offering smooth navigation between habit lists, calendar views, and reports.
3. Development:
    - Backend Development:
        - Use Android SDK with Java/Kotlin for app logic.
        - Implement habit creation, modification, and deletion functionalities using local storage mechanisms like Room Database to store user-defined habits.
        - Schedule habit reminders using WorkManager to notify users at appropriate times.
        - Track progress by calculating streaks, generating reports, and displaying completion rates using appropriate database queries.
        - Handle user permissions (e.g., notification permissions) and ensure that reminders are handled reliably in the background.
    - Frontend Development:
        - Develop XML layouts to create a clean, intuitive interface for managing habits and viewing progress.

        - Implement real-time updates in the UI, ensuring users can see progress instantly after completing a habit.
        - Incorporate calendar and chart views to visualize habit tracking and progress, allowing users to easily identify patterns in their behavior.

4. Testing:
    - Unit Testing: Perform unit testing on individual modules, such as habit creation, reminder scheduling, and progress tracking, to ensure they function correctly in isolation.
    - Integration Testing: Validate that all components work together seamlessly, including the interaction between the backend (database and notifications) and frontend (UI updates).
    - User Testing: Conduct user testing across various Android devices and versions to check for compatibility and ensure a consistent user experience (UX).
    - Simulation Testing: Simulate various usage patterns (e.g., multiple reminders, long habit streaks) to test the app's reliability in real-world scenarios.
5. Maintenance:
    - Regular Updates: Roll out updates based on user feedback, adding new features like support for additional habit types or advanced progress analytics.
    - Adaptation: Monitor changes in Android APIs and update the app to remain compatible with newer versions of Android and evolving technologies, ensuring a smooth user experience over time.

This methodology ensures a systematic and user-friendly approach to developing Track Master, providing an efficient habit-tracking app that helps users build and maintain positive habits with ease.

Screenshots And Code

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.HabitTracker"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/Theme.HabitTracker">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- SignupActivity: User registration screen -->
        <activity
            android:name=".SignupActivity"
            android:exported="true"
            android:theme="@style/Theme.HabitTracker">
        </activity>

        <!-- LoginActivity: User login screen -->
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:theme="@style/Theme.HabitTracker">
        </activity>

        <!-- DashboardActivity: After login success, welcome screen -->
        <activity
            android:name=".DashboardActivity"
            android:exported="true"
            android:theme="@style/Theme.HabitTracker">
        </activity>
        <activity
            android:name=".ProgressReportActivity"
            android:exported="true"
            android:theme="@style/Theme.HabitTracker">
        </activity>
    </application>

</manifest>
```

MainActivity.java

```java
package com.example.habittracker;

import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.content.Intent;
import android.widget.Button;
```

```
public class MainActivity extends AppCompatActivity {

    Button login, signup;

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

        login = findViewById(R.id.login);
        signup = findViewById(R.id.signup);

        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(), LoginActivity.class));
            }
        });

        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(), SignupActivity.class));
            }
        });
    }
}
```

SignupActivity.java

```
package com.example.habittracker;


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

public class SignupActivity extends AppCompatActivity {

    DatabaseHelper db;
    EditText username, password, confirmPassword;
    Button signup;

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

        db = new DatabaseHelper(this);

        username = findViewById(R.id.username);
        password = findViewById(R.id.password);
        confirmPassword = findViewById(R.id.confirmPassword);
        signup = findViewById(R.id.signup);

        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String user = username.getText().toString();
                String pass = password.getText().toString();
                String confirmPass = confirmPassword.getText().toString();


                if (user.equals("") || pass.equals("") || confirmPass.equals("")) {
                    Toast.makeText(SignupActivity.this, "Please fill out all fields", Toast.LENGTH_SHORT).show();
                } else {
                    if (pass.equals(confirmPass)) {
                        boolean checkUser = db.checkUsername(user);
                        if (!checkUser) {
                            boolean insert = db.insertUser(user, pass);
                            if (insert) {
                                // Fetch the userId for the newly registered user
                                int userId = db.getUserId(user);
                                if (userId != -1) {
                                    Toast.makeText(SignupActivity.this, "Registered successfully!", Toast.LENGTH_SHORT).show();

                                    // Pass the userId to the next activity
                                    Intent intent = new Intent(getApplicationContext(), DashboardActivity.class);
                                    intent.putExtra("USER_ID", userId);
                                    startActivity(intent);
                                    finish();
                                } else {
                                    Toast.makeText(SignupActivity.this, "Failed to retrieve user ID", Toast.LENGTH_SHORT).show();
                                }
                            } else {
                                Toast.makeText(SignupActivity.this, "Registration failed", Toast.LENGTH_SHORT).show();
                            }
                        } else {
                            Toast.makeText(SignupActivity.this, "Username already exists", Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        Toast.makeText(SignupActivity.this, "Passwords do not match", Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });
    }
}
```

LoginActivity.java

```
package com.example.habittracker;

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

public class LoginActivity extends AppCompatActivity {

    DatabaseHelper db;
    EditText username, password;
    Button login;

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

        db = new DatabaseHelper(this);

        username = findViewById(R.id.username);
        password = findViewById(R.id.password);
        login = findViewById(R.id.login);

        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String user = username.getText().toString();
                String pass = password.getText().toString();

                if (user.equals("") || pass.equals("")) {
                    Toast.makeText(LoginActivity.this, "Please enter all fields", Toast.LENGTH_SHORT).show();
                } else {
                    boolean checkUser = db.checkUserCredentials(user, pass);
                    if (checkUser) {
                        // Fetch user ID after successful login
                        int userId = db.getUserId(user);
                        if (userId != -1) {
                            // Redirect to DashboardActivity and pass the userId
                            Intent intent = new Intent(getApplicationContext(), DashboardActivity.class);
                            intent.putExtra("USER_ID", userId);
                            intent.putExtra("USER_NAME", user);
                            startActivity(intent);
                            finish();
                        } else {
                            Toast.makeText(LoginActivity.this, "Failed to retrieve user ID", Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        Toast.makeText(LoginActivity.this, "Invalid credentials", Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });
    }
}
```

DashboardActivity.java

```
package com.example.habittracker;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
```

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

public class DashboardActivity extends AppCompatActivity {

    private TextView greetingText;
    private EditText walkingStepsDisplay, caloriesDisplay, hydrationDisplay, yogaDisplay, sleepDisplay, readingDisplay;
    private Button walkingIncrementButton, caloriesIncrementButton, hydrationIncrementButton, yogaIncrementButton,
sleepIncrementButton, readingIncrementButton;
    private CheckBox walkingCheckbox, caloriesCheckbox, hydrationCheckbox, yogaCheckbox, sleepCheckbox, readingCheckbox;
    private Button doneButton, viewProgressButton;
    private int userId;
    private DatabaseHelper dbHelper;

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

        // Get user ID from login
        userId = getIntent().getIntExtra("USER_ID", -1);
        String username = getIntent().getStringExtra("USER_NAME");

        if (userId == -1) {
            Toast.makeText(this, "User ID not found", Toast.LENGTH_SHORT).show();
            finish();  // Exit the activity if no valid user ID is passed
        }

        // Initialize UI elements
        initializeUIElements();
        dbHelper = new DatabaseHelper(this);
        setGreetingText(username);

        // Load today's habit data if exists
        loadTodayHabitData();

        // Set up increment button listeners
        setUpIncrementButtonListeners();

        // Done Button Logic to Save Habits
        doneButton.setOnClickListener(v -> saveUserHabits());


        // Inside onCreate method
        viewProgressButton=findViewById(R.id.view_progress_button);
        viewProgressButton.setOnClickListener(v -> viewProgressReport());

    }

        private void viewProgressReport() {
            Intent intent = new Intent(DashboardActivity.this, ProgressReportActivity.class);

            // Pass the current habit data to the ProgressReportActivity
            intent.putExtra("WALKING_STEPS", Integer.parseInt(walkingStepsDisplay.getText().toString()));
            intent.putExtra("CALORIES", Integer.parseInt(caloriesDisplay.getText().toString()));
            intent.putExtra("HYDRATION", Double.parseDouble(hydrationDisplay.getText().toString()));
            intent.putExtra("YOGA_MINUTES", Integer.parseInt(yogaDisplay.getText().toString()));
            intent.putExtra("SLEEP_HOURS", Integer.parseInt(sleepDisplay.getText().toString()));
            intent.putExtra("READING_PAGES", Integer.parseInt(readingDisplay.getText().toString()));

            startActivity(intent);  // Launch the ProgressReportActivity
    }


    private void initializeUIElements() {
        greetingText = findViewById(R.id.greeting_text);
        walkingStepsDisplay = findViewById(R.id.walking_edit_text);
        walkingIncrementButton = findViewById(R.id.walking_increment);
        walkingCheckbox = findViewById(R.id.walking_checkbox);
        caloriesDisplay = findViewById(R.id.calories_edit_text);
        caloriesIncrementButton = findViewById(R.id.calories_increment);
        caloriesCheckbox = findViewById(R.id.calories_checkbox);
        hydrationDisplay = findViewById(R.id.hydration_edit_text);
        hydrationIncrementButton = findViewById(R.id.hydration_increment);
        hydrationCheckbox = findViewById(R.id.hydration_checkbox);
        yogaDisplay = findViewById(R.id.yoga_edit_text);
        yogaIncrementButton = findViewById(R.id.yoga_increment);
        yogaCheckbox = findViewById(R.id.yoga_checkbox);
        sleepDisplay = findViewById(R.id.sleep_edit_text);
        sleepIncrementButton = findViewById(R.id.sleep_increment);
        sleepCheckbox = findViewById(R.id.sleep_checkbox);
        readingDisplay = findViewById(R.id.reading_edit_text);
        readingIncrementButton = findViewById(R.id.reading_increment);
        readingCheckbox = findViewById(R.id.reading_checkbox);
        doneButton = findViewById(R.id.done_button);
    }


    private void setGreetingText(String userName) {
        Calendar calendar = Calendar.getInstance();
        int hour = calendar.get(Calendar.HOUR_OF_DAY);
        String greeting;

        // Determine the appropriate greeting based on the current hour
        if (hour < 12) {
            greeting = "Good Morning, " + userName; // Morning
        } else if (hour < 17) {
            greeting = "Good Afternoon, " + userName; // Afternoon
        } else {
            greeting = "Good Evening, " + userName; // Evening
        }

        greetingText.setText(greeting); // Set greeting text

        // Display the current date
        SimpleDateFormat sdf = new SimpleDateFormat("MMMM dd, yyyy", Locale.getDefault());
        String currentDate = sdf.format(calendar.getTime());
        // Assuming you have a TextView for the date
        TextView dateTextView = findViewById(R.id.current_date); // Make sure you have this TextView in your layout
        dateTextView.setText("Today's Date: " + currentDate);
    }
    //private void setGreetingText(String username) {
        //String greeting = "Good Morning";
        //greetingText.setText(greeting);
    //}

    private void setUpIncrementButtonListeners() {
        walkingIncrementButton.setOnClickListener(v -> incrementWalkingSteps());
        caloriesIncrementButton.setOnClickListener(v -> incrementCalories());
        hydrationIncrementButton.setOnClickListener(v -> incrementHydration());
        yogaIncrementButton.setOnClickListener(v -> incrementYoga());
        sleepIncrementButton.setOnClickListener(v -> incrementSleep());
        readingIncrementButton.setOnClickListener(v -> incrementReading());
    }
```

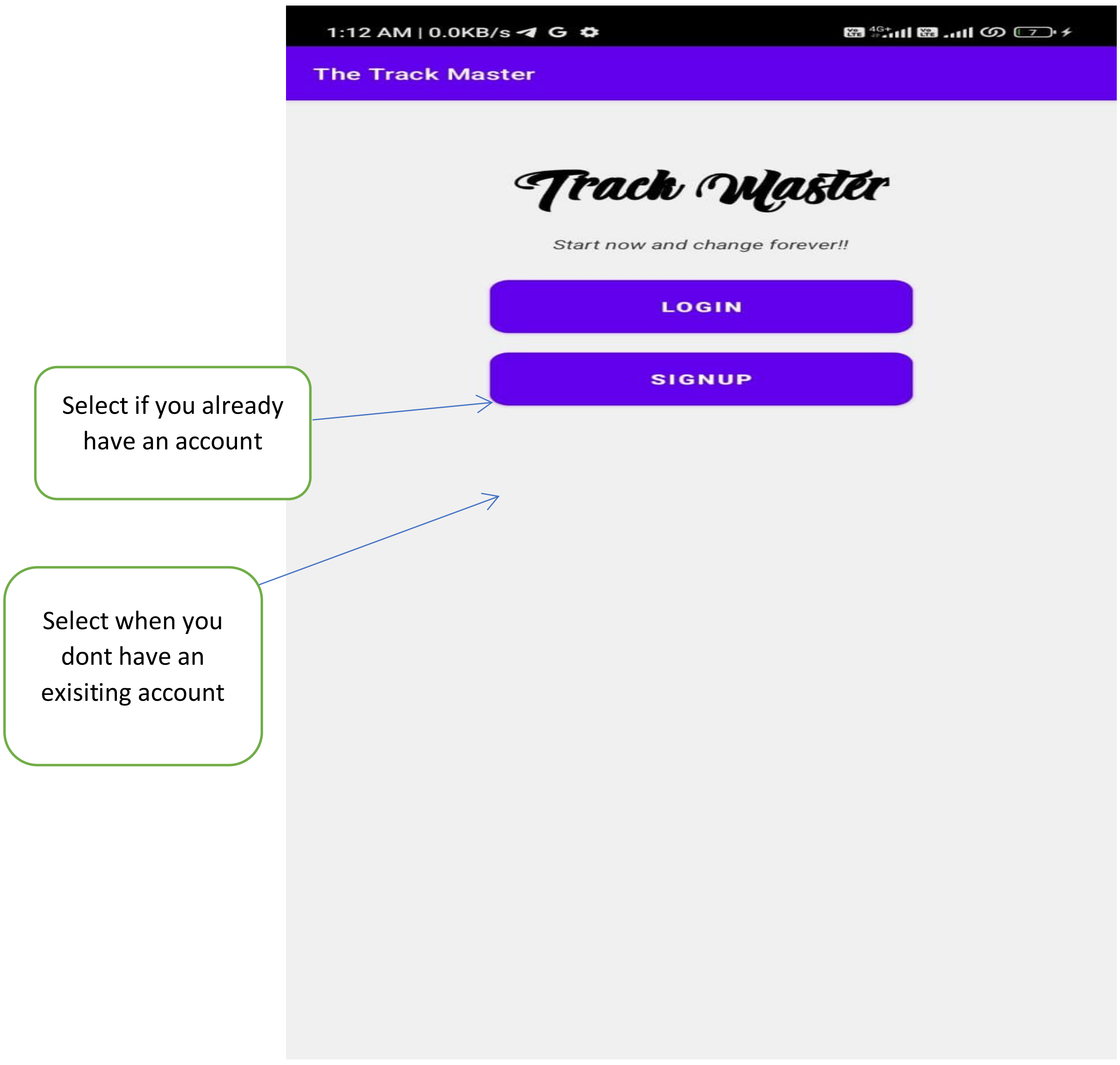
```
  private void loadTodayHabitData() {
    Cursor cursor = dbHelper.getTodayHabitData(userId);

    // Check if cursor is null or empty
    if (cursor != null && cursor.moveToFirst()) {
      // Use safe method to get column indices
      int walkingIndex = cursor.getColumnIndex(DatabaseHelper.COL_WALKING_STEPS);
      int caloriesIndex = cursor.getColumnIndex(DatabaseHelper.COL_CALORIES);
      int hydrationIndex = cursor.getColumnIndex(DatabaseHelper.COL_HYDRATION);
      int yogaIndex = cursor.getColumnIndex(DatabaseHelper.COL_YOGA_MINUTES);
      int sleepIndex = cursor.getColumnIndex(DatabaseHelper.COL_SLEEP_HOURS);
      int readingIndex = cursor.getColumnIndex(DatabaseHelper.COL_READING_PAGES);

      // Populate the fields with existing data if column indices are valid
      if (walkingIndex != -1) {
        String walkingSteps = cursor.getString(walkingIndex);
        walkingStepsDisplay.setText(walkingSteps);
        walkingCheckbox.setChecked(Integer.parseInt(walkingSteps) >= 10000);
      }

      if (caloriesIndex != -1) {
        String calories = cursor.getString(caloriesIndex);
        caloriesDisplay.setText(calories);
        caloriesCheckbox.setChecked(Integer.parseInt(calories) < 1500);
      }

      if (hydrationIndex != -1) {
        String hydration = cursor.getString(hydrationIndex);
        hydrationDisplay.setText(hydration);
        hydrationCheckbox.setChecked(Double.parseDouble(hydration) >= 3);
      }

      if (yogaIndex != -1) {
        String yoga = cursor.getString(yogaIndex);
        yogaDisplay.setText(yoga);
        yogaCheckbox.setChecked(Integer.parseInt(yoga) >= 15);
      }

      if (sleepIndex != -1) {
        String sleep = cursor.getString(sleepIndex);
        sleepDisplay.setText(sleep);
        sleepCheckbox.setChecked(Integer.parseInt(sleep) >= 7);
      }

      if (readingIndex != -1) {
        String readingPages = cursor.getString(readingIndex);
        readingDisplay.setText(readingPages);
        // Add any additional logic for reading if needed
      }
    } else {
      Log.e("DashboardActivity", "No habit data found for user ID: " + userId);
    }

    // Ensure cursor is closed to prevent memory leaks
    if (cursor != null) {
      cursor.close();
    }
  }


  private void incrementWalkingSteps() {
    try {
            int steps = Integer.parseInt(walkingStepsDisplay.getText().toString());
            steps += 50; // Increment steps by 50
            walkingStepsDisplay.setText(String.valueOf(steps));
            walkingCheckbox.setChecked(steps >= 10000); // Target: 10,000 steps
        } catch (NumberFormatException e) {
            Toast.makeText(DashboardActivity.this, "Invalid number", Toast.LENGTH_SHORT).show();
        }
    }

    private void incrementCalories() {
        try {
            int calories = Integer.parseInt(caloriesDisplay.getText().toString());
            calories += 50; // Increment calories by 50
            caloriesDisplay.setText(String.valueOf(calories));
            caloriesCheckbox.setChecked(calories < 1500); // Target: <1500 calories
        } catch (NumberFormatException e) {
            Toast.makeText(DashboardActivity.this, "Invalid number", Toast.LENGTH_SHORT).show();
        }
    }

    private void incrementHydration() {
        try {
            double liters = Double.parseDouble(hydrationDisplay.getText().toString());
            liters += 0.5; // Increment water intake by 0.5 liters
            hydrationDisplay.setText(String.valueOf(liters));
            hydrationCheckbox.setChecked(liters >= 3); // Target: 3 liters
        } catch (NumberFormatException e) {
            Toast.makeText(DashboardActivity.this, "Invalid number", Toast.LENGTH_SHORT).show();
        }
    }

    private void incrementYoga() {
        try {
            int minutes = Integer.parseInt(yogaDisplay.getText().toString());
            minutes += 5; // Increment yoga minutes by 5
            yogaDisplay.setText(String.valueOf(minutes));
            yogaCheckbox.setChecked(minutes >= 15); // Target: 15 minutes
        } catch (NumberFormatException e) {
            Toast.makeText(DashboardActivity.this, "Invalid number", Toast.LENGTH_SHORT).show();
        }
    }

    private void incrementSleep() {
        try {
            int hours = Integer.parseInt(sleepDisplay.getText().toString());
            hours += 1; // Increment sleep hours by 1
            sleepDisplay.setText(String.valueOf(hours));
            sleepCheckbox.setChecked(hours >= 7); // Target: 7 hours
        } catch (NumberFormatException e) {
            Toast.makeText(DashboardActivity.this, "Invalid number", Toast.LENGTH_SHORT).show();
        }
    }

    private void incrementReading() {
        try {
            int pages = Integer.parseInt(readingDisplay.getText().toString());
            pages += 1; // Increment reading pages by 1
            readingDisplay.setText(String.valueOf(pages));
            readingCheckbox.setChecked(pages >= 10); // Target: 10 pages
        } catch (NumberFormatException e) {
            Toast.makeText(DashboardActivity.this, "Invalid number", Toast.LENGTH_SHORT).show();
        }
    }
    private void saveUserHabits() {
        // Get current data from EditTexts
        int walkingSteps = Integer.parseInt(walkingStepsDisplay.getText().toString());
        int calories = Integer.parseInt(caloriesDisplay.getText().toString());
        double hydration = Double.parseDouble(hydrationDisplay.getText().toString());
        int yogaMinutes = Integer.parseInt(yogaDisplay.getText().toString());
        int sleepHours = Integer.parseInt(sleepDisplay.getText().toString());
        int readingPages = Integer.parseInt(readingDisplay.getText().toString());

        // Save today's habits to the database
        boolean isUpdated = dbHelper.saveUserHabits(userId, walkingSteps, calories, hydration, yogaMinutes, sleepHours,
readingPages,null);
        if (isUpdated) {
            Toast.makeText(DashboardActivity.this, "Habits updated successfully!", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(DashboardActivity.this, "Error updating habits.", Toast.LENGTH_SHORT).show();
        }
    }


}
```

DatabaseHelper.java

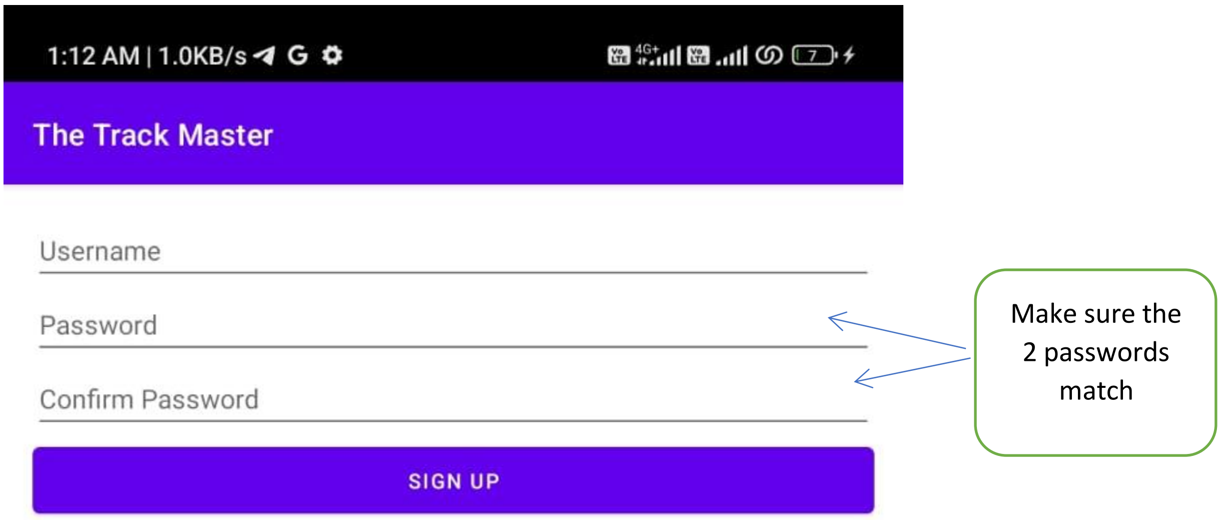
```
package com.example.habittracker;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class DatabaseHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "UserData.db";
    public static final String TABLE_USERS = "users";
    public static final String TABLE_HABITS = "habits";

    // User Table columns
    public static final String COL_1 = "ID";
    public static final String COL_2 = "USERNAME";
    public static final String COL_3 = "PASSWORD";

    // Habits Table columns
    public static final String COL_HABIT_ID = "HABIT_ID";
    public static final String COL_USER_ID = "USER_ID";
    public static final String COL_WALKING_STEPS = "WALKING_STEPS";
    public static final String COL_CALORIES = "CALORIES";
    public static final String COL_HYDRATION = "HYDRATION";
    public static final String COL_YOGA_MINUTES = "YOGA_MINUTES";
    public static final String COL_SLEEP_HOURS = "SLEEP_HOURS";
    public static final String COL_READING_PAGES = "READING_PAGES";
    public static final String COL_DATE = "DATE";  // Store a formatted date

    // Singleton instance
    private static DatabaseHelper instance;

    // Get Singleton instance of DatabaseHelper
    public static synchronized DatabaseHelper getInstance(Context context) {
        if (instance == null) {
            instance = new DatabaseHelper(context.getApplicationContext());
        }
        return instance;
    }

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        Log.d("DatabaseHelper", "Creating tables");
        db.execSQL("CREATE TABLE " + TABLE_USERS + " (ID INTEGER PRIMARY KEY AUTOINCREMENT, USERNAME
TEXT, PASSWORD TEXT)");
        db.execSQL("CREATE TABLE " + TABLE_HABITS + " (HABIT_ID INTEGER PRIMARY KEY AUTOINCREMENT,
USER_ID INTEGER, WALKING_STEPS INTEGER, CALORIES INTEGER, HYDRATION REAL, YOGA_MINUTES INTEGER,
SLEEP_HOURS INTEGER, READING_PAGES INTEGER, DATE TEXT, FOREIGN KEY(USER_ID) REFERENCES " +
TABLE_USERS + "(ID))");

        // Add index on USER_ID and DATE for performance optimization
        db.execSQL("CREATE INDEX idx_user_date ON " + TABLE_HABITS + " (USER_ID, DATE)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_USERS);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_HABITS);
        onCreate(db);
    }

    // Insert a new user into the database (Consider hashing passwords for security)
    public boolean insertUser(String username, String password) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COL_2, username);
        contentValues.put(COL_3, password); // Hash the password for security
        long result = db.insert(TABLE_USERS, null, contentValues);
        return result != -1;  // Return true if insertion was successful
    }

    // Check if the username already exists
    public boolean checkUsername(String username) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM " + TABLE_USERS + " WHERE USERNAME = ?", new String[]{username});
        boolean exists = cursor.getCount() > 0;
        cursor.close();
        return exists;
    }

    // Validate user credentials (Consider using hashed passwords)
    public boolean checkUserCredentials(String username, String password) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM " + TABLE_USERS + " WHERE USERNAME = ? AND PASSWORD = ?",
new String[]{username, password});
        boolean valid = cursor.getCount() > 0;
        cursor.close();
        return valid;
    }

    // Method to save or update user habits
    public boolean saveUserHabits(int userId, int walkingSteps, int calories, double hydration, int yogaMinutes, int sleepHours, int
readingPages, String date) {
        SQLiteDatabase db = this.getWritableDatabase();
        if (date == null) {
            date = getCurrentDate();
        }

        try {
            // Check if the user exists
            Cursor userCursor = db.rawQuery("SELECT ID FROM " + TABLE_USERS + " WHERE ID = ?", new
String[]{String.valueOf(userId)});
            boolean userExists = userCursor.getCount() > 0;
            userCursor.close();

            if (!userExists) {
                Log.e("DatabaseHelper", "User with ID " + userId + " does not exist.");
                return false;
            }

            // Check if data for the given date already exists
            Cursor dataCursor = db.rawQuery("SELECT * FROM " + TABLE_HABITS + " WHERE USER_ID = ? AND DATE = ?", new
String[]{String.valueOf(userId), date});
            boolean dataExists = dataCursor.getCount() > 0;
            dataCursor.close();

            // Prepare content values
            ContentValues contentValues = new ContentValues();
            contentValues.put(COL_USER_ID, userId);
            contentValues.put(COL_WALKING_STEPS, walkingSteps);
            contentValues.put(COL_CALORIES, calories);
            contentValues.put(COL_HYDRATION, hydration);
            contentValues.put(COL_YOGA_MINUTES, yogaMinutes);
            contentValues.put(COL_SLEEP_HOURS, sleepHours);
            contentValues.put(COL_READING_PAGES, readingPages);
            contentValues.put(COL_DATE, date);

            long result;
            if (dataExists) {
                // Update existing record
                result = db.update(TABLE_HABITS, contentValues, COL_USER_ID + " = ? AND " + COL_DATE + " = ?", new
String[]{String.valueOf(userId), date});
                Log.d("DatabaseHelper", "Data updated successfully for user ID: " + userId);
            } else {
                // Insert a new record
                result = db.insert(TABLE_HABITS, null, contentValues);
                Log.d("DatabaseHelper", "Data inserted successfully for user ID: " + userId);
            }

            return result != -1;
        } catch (SQLException e) {
            Log.e("DatabaseHelper", "SQL Exception: " + e.getMessage(), e);
            return false;
        } finally {
            // No need to manually close db, SQLiteOpenHelper manages it
        }
    }

    // Method to get user's ID by username
    public int getUserId(String username) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT ID FROM " + TABLE_USERS + " WHERE USERNAME = ?", new String[]{username});
        if (cursor.moveToFirst()) {
            int userId = cursor.getInt(0);
            cursor.close();
            return userId;
        } else {
            cursor.close();
            return -1;  // Return -1 if user not found
        }
    }

    // Method to retrieve habit data for a specific date
    public Cursor getHabitDataByDate(int userId, String date) {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_HABITS + " WHERE " + COL_USER_ID + " = ? AND " + COL_DATE + " =
?", new String[]{String.valueOf(userId), date});
    }

    // Method to retrieve all habit data for a user
    public Cursor getUserHabits(int userId) {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_HABITS + " WHERE " + COL_USER_ID + " = ?", new
String[]{String.valueOf(userId)});
    }

    // Method to get today's habit data for a user
    public Cursor getTodayHabitData(int userId) {
        return getHabitDataByDate(userId, getCurrentDate());
    }

    // Method to update today's habit data
    public boolean updateTodayHabitData(int userId, int walkingSteps, int calories, double hydration, int yogaMinutes, int sleepHours,
int readingPages) {
        return saveUserHabits(userId, walkingSteps, calories, hydration, yogaMinutes, sleepHours, readingPages, null);
    }

    // Method to get the current date in "yyyy-MM-dd" format
    private String getCurrentDate() {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
        return sdf.format(new Date());
    }
}
```

Screenshots

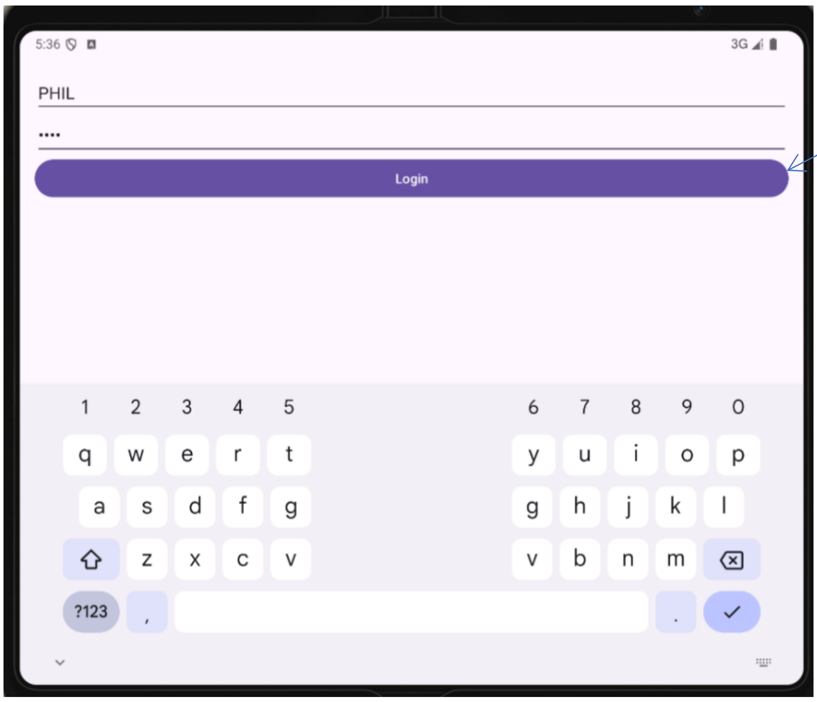
First time open app then first of all you need to signup . The user is provided with 2 options of whether the user wants to Signup or Login


1:12 AM | 0.0KB/s
The Track Master
Track Master
Start now and change forever!!
LOGIN
SIGNUP
Select if you already have an account
Select when you dont have an exisiting account

If the user is new and wants to create the account inorder to use the application once the SIGNUP button is clicked the user has to enter this information.

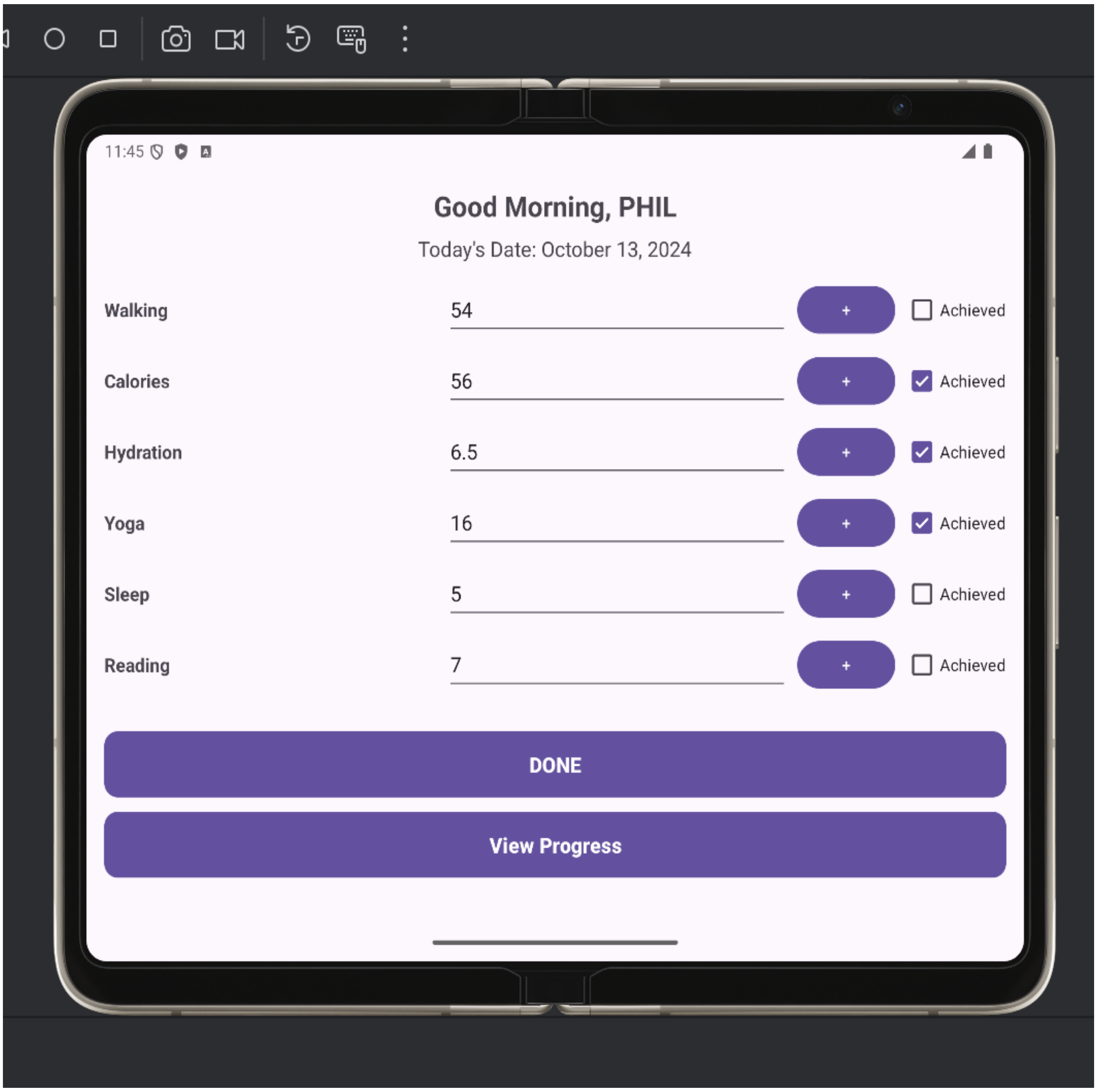


If the user already has an account then the user simply has to enter the username and the correct corresponding password



Login after entering valid username

Once the user is successfully logged in, the user is provided with a list of habits and the user can update them according to that specific day activities

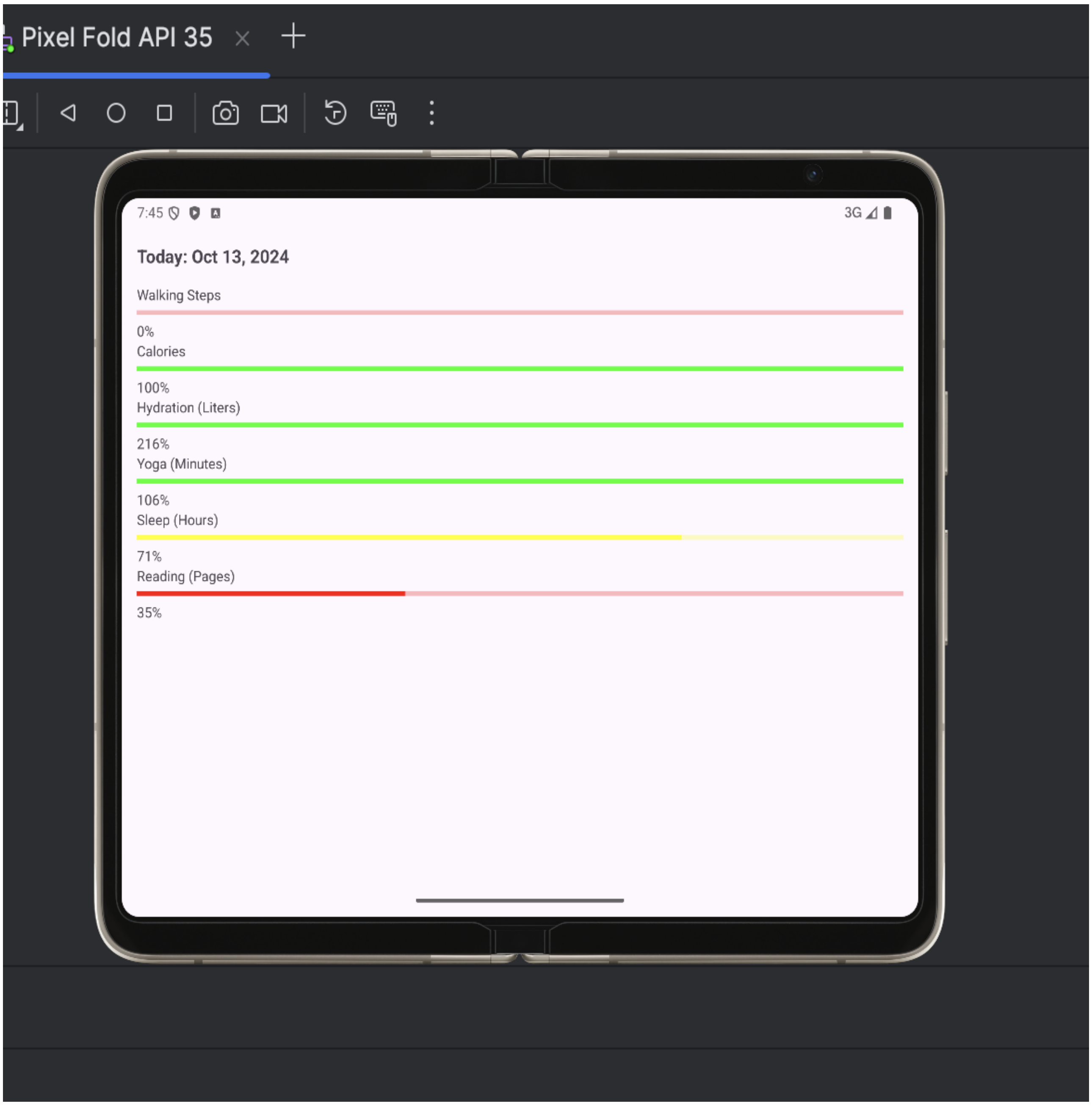

11:45

Good Morning, PHIL

Today's Date: October 13, 2024

Walking	54	+	☐ Achieved
Calories	56	+	☑ Achieved
Hydration	6.5	+	☑ Achieved
Yoga	16	+	☑ Achieved
Sleep	5	+	☐ Achieved
Reading	7	+	☐ Achieved

The user can view progress of each day by clicking the View Progress button



Today: Oct 13, 2024

Walking Steps

0%

Calories

100%

Hydration (Liters)

216%

Yoga (Minutes)

106%

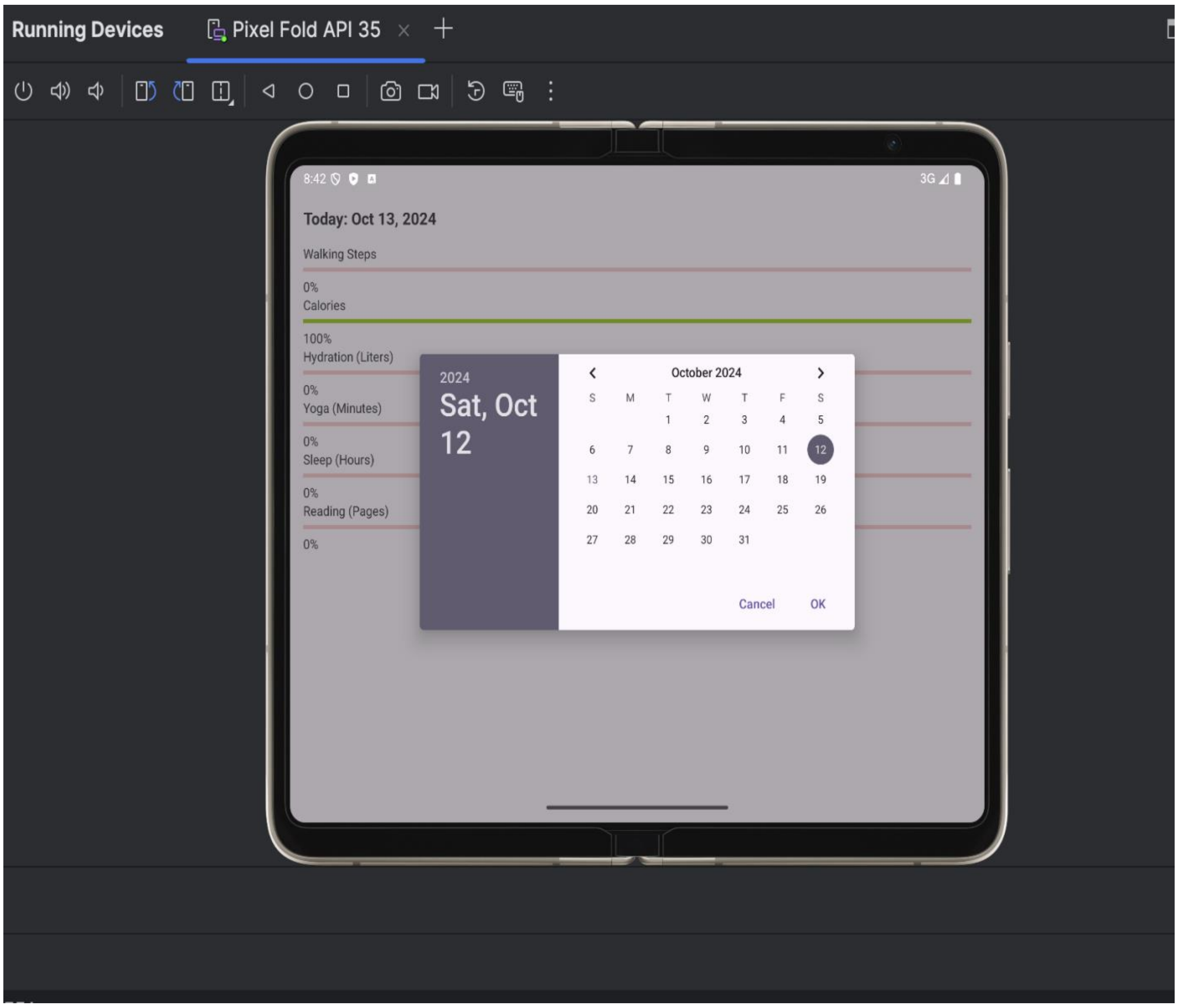
Sleep (Hours)

71%

Reading (Pages)

35%

The user can also view the progress data of the past dates



Today: Oct 13, 2024

Walking Steps

0%

Calories

100%

Hydration (Liters)

0%

Yoga (Minutes)

0%

Sleep (Hours)

0%

Reading (Pages)

0%

2024

Sat, Oct 12

October 2024

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Conclusion

The Habit Tracker application successfully demonstrates the potential of technology in facilitating personal development and promoting healthier lifestyle choices. By providing users with an intuitive interface and essential features such as habit logging, progress tracking, and personalized feedback, the application empowers individuals to take control of their daily routines and establish sustainable habits.

Throughout the development process, we have implemented key functionalities, including user authentication, data management, and a date picker for tracking habits over time. These features enhance user engagement and provide valuable insights into individual progress, ultimately contributing to improved accountability and motivation.

As we reflect on this project, we recognize the importance of continual improvement. Future enhancements could include the integration of reminders, social sharing capabilities, and advanced analytics to provide users with a deeper understanding of their habits and achievements. Additionally, we aim to improve the application by adding more features based on user feedback, which could include support for more diverse habits and personalized goal-setting. These improvements will significantly enrich the user experience and help users achieve their objectives more effectively.

In conclusion, the Habit Tracker serves not only as a practical tool for habit formation but also as a stepping stone toward a healthier lifestyle. By leveraging technology to support personal growth, we aim to inspire users to embrace positive change and cultivate habits that enhance their overall well-being.

Reference