# EEC 693 Special Topics in Elect Engr Section 51
# Sensor Recorder Application

Name: Sindooja Bhagwan Gajam
CSU ID: 2888228

## Table of Contents

- **Introduction**
- **Purpose and Scope**
- **Architecture**
- **Design**
- **Core Features**
- **Technology, Software, and Hardware Used**
- **Activity Layouts**
- **Activities**
- **Configuration and Build Gradle**
- **Application Screenshots**
- **Challenges Faced**
- **Conclusion**

# 1. Introduction

The Sensor Recorder Application is an Android-based mobile application designed to utilize a smartphone's built-in sensors. It employs the Android SensorManager API to access and manage sensor data, including accelerometer and gyroscope readings. The application listens to real-time sensor events and processes the data to provide immediate visual feedback in the user interface. This approach ensures accurate and efficient utilization of the device's embedded sensors for data recording and analysis. The application facilitates user registration, login, and password recovery using Firebase authentication. It records accelerometer and gyroscope data, displays them in real time on the UI, and logs the data into a CSV file for later visualization in graph form.

This report aims to provide a detailed evaluation of the Sensor Recorder app, including its architecture, core functionalities, and potential areas for enhancement. The application's design considerations, development methodologies, and performance characteristics will also be explored. To support this analysis, screenshots of the app's interface and outputs will be included where appropriate, offering a visual perspective on its capabilities.

# 2. Purpose and Scope

Purpose:
The primary purpose of this application is to provide an easy-to-use tool for recording sensor data for analytical and educational purposes.

Scope:

- Enable users to interact with their mobile sensors.
- Log and visualize sensor data.
- Offer secure user authentication.

# 3. Architecture

The application follows a robust layered architecture, ensuring modularity, scalability, and maintainability:

1. **Presentation Layer**:

    - This layer includes UI components responsible for capturing user inputs and presenting data visually. Activities like MainActivity and LoginActivity reside here, ensuring a user-friendly experience.
    - Example: When a user selects a CSV file, the MainActivity interacts with the business logic to process and display sensor data visually.
2. **Business Logic Layer**:
    - Handles core functionalities such as Firebase authentication, sensor data processing, and file operations like generating CSV files.
    - Example: Upon receiving login credentials from the Presentation Layer, this layer communicates with the Data Layer to validate the credentials against Firebase.
3. **Data Layer**:
    - Manages data storage and retrieval. This includes Firebase Authentication and Realtime Database integration, as well as local file operations for storing sensor logs.
    - Example: Sensor data is processed and logged into CSV files by this layer for persistence and future analysis.

---

**Case Study: Login Flow**
To understand the interaction between layers, consider the user login process:

- **Step 1**: The user enters their email and password in the LoginActivity (Presentation Layer).
- **Step 2**: These inputs are forwarded to the authentication function in the Business Logic Layer.
- **Step 3**: The Business Logic Layer interacts with Firebase (Data Layer) to verify credentials.
- **Step 4**: The response is sent back to the Business Logic Layer to determine if the user should be granted access.
- **Step 5**: Finally, the Presentation Layer updates the UI based on the result (success or error).

---

## 4. Design

The Sensor Recorder application is built using Android's robust development framework and adheres to modular design principles to ensure scalability and maintainability.

**Project Structure**:

- **Java Files**: Core logic is distributed across files such as MainActivity.java, SensorService.java, and DataStorageHandler.java. Each file encapsulates specific functionality, ensuring a clear separation of concerns.
- **XML Files**: These files define the user interface and app configurations. For instance, activity_main.xml governs the layout, while AndroidManifest.xml specifies permissions and application components.

The app leverages Android's SensorManager API to interface with device sensors, adopting a service-oriented architecture to handle background processes efficiently. This design ensures smooth interaction between the user interface, sensor data collection, and storage mechanisms.

---

## 5. Core Features

- **User Registration and Login:** Firebase-backed authentication with forgot password functionality.
- **Sensor Data Display:** Real-time accelerometer and gyroscope data shown in a table.
- **Data Logging:** Sensor readings logged into a CSV file.
- **Graph Visualization:** Users can select a CSV file and view the logged data in graphical form.

---

## 6. Technology, Software, and Hardware Used

Technology:

- Programming Language: Java
- IDE: Android Studio
- Backend: Firebase Authentication and Realtime Database

Software:

- Android SDK

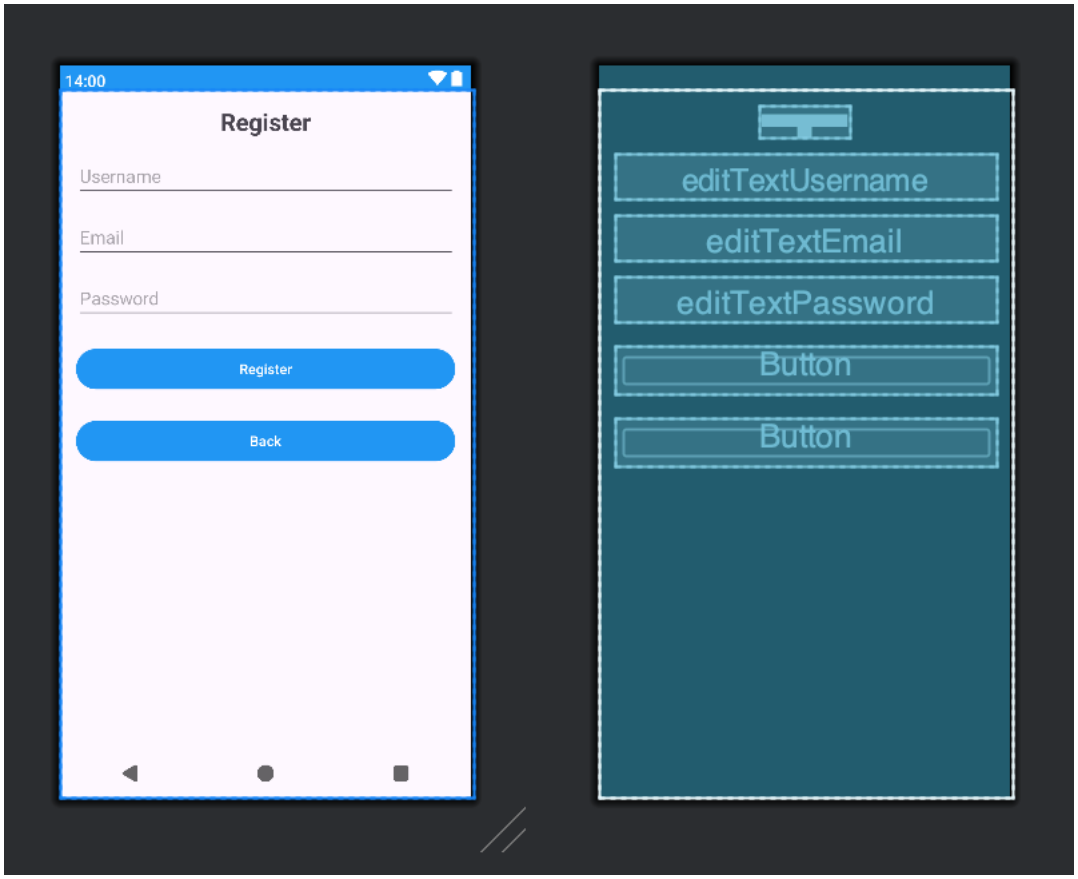
Hardware:

- Android smartphone with accelerometer and gyroscope sensors

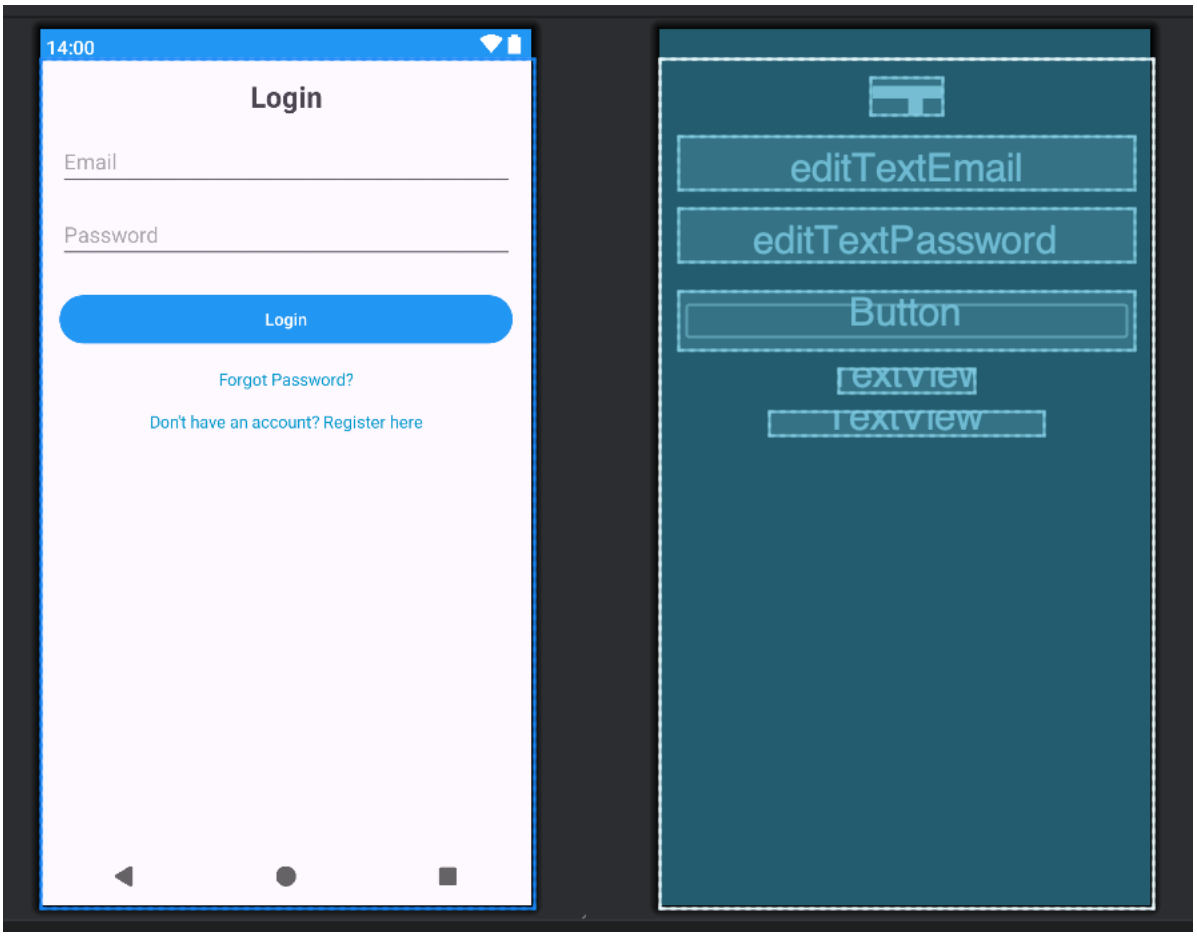
---

## 7. Activity Layouts

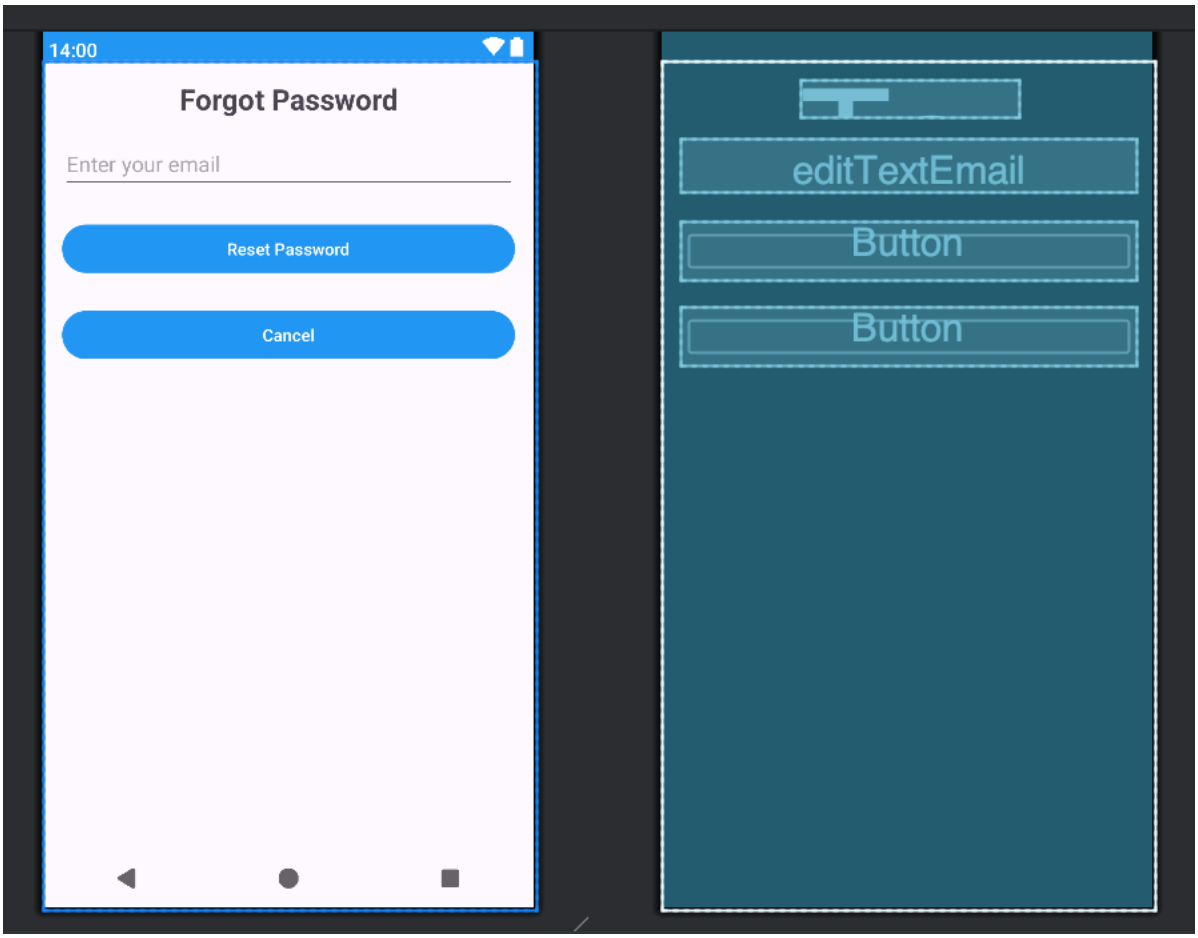
1. Registration Activity Layout



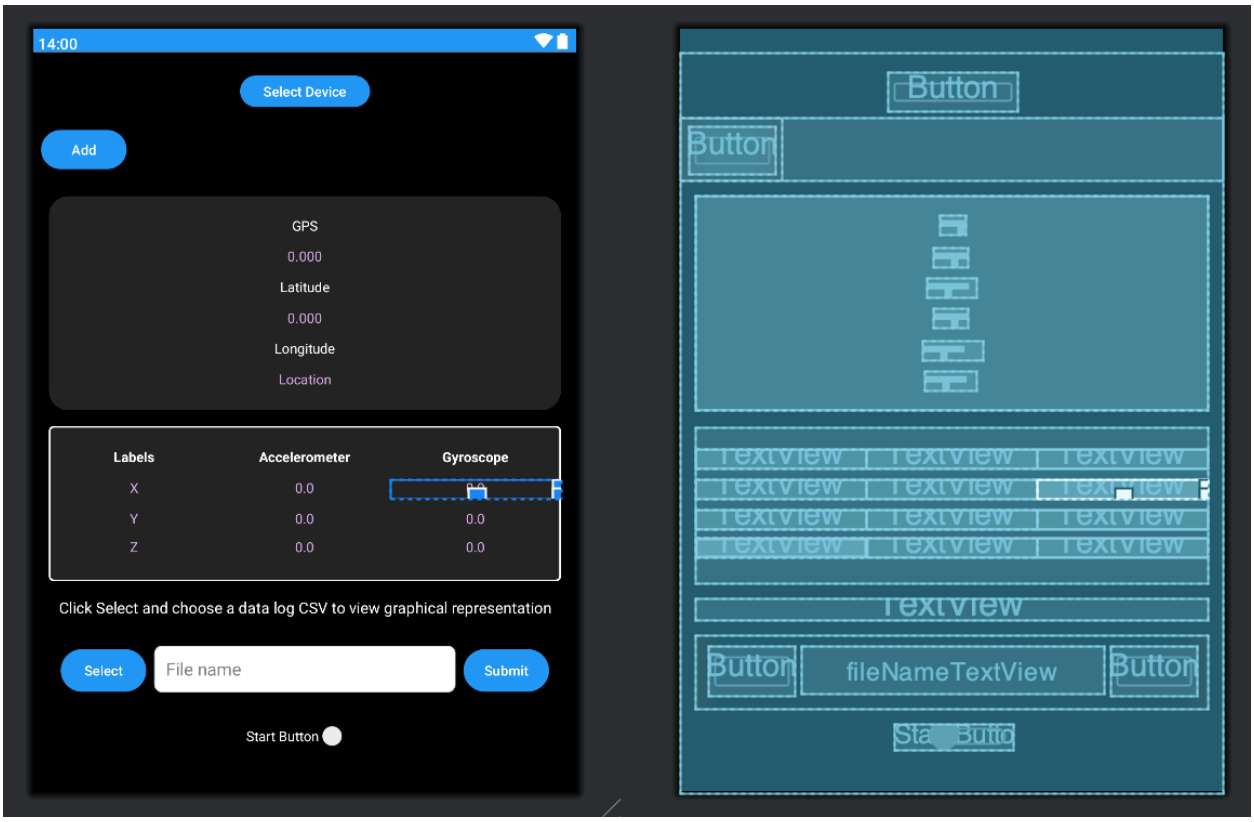
2. Login Activity Layout



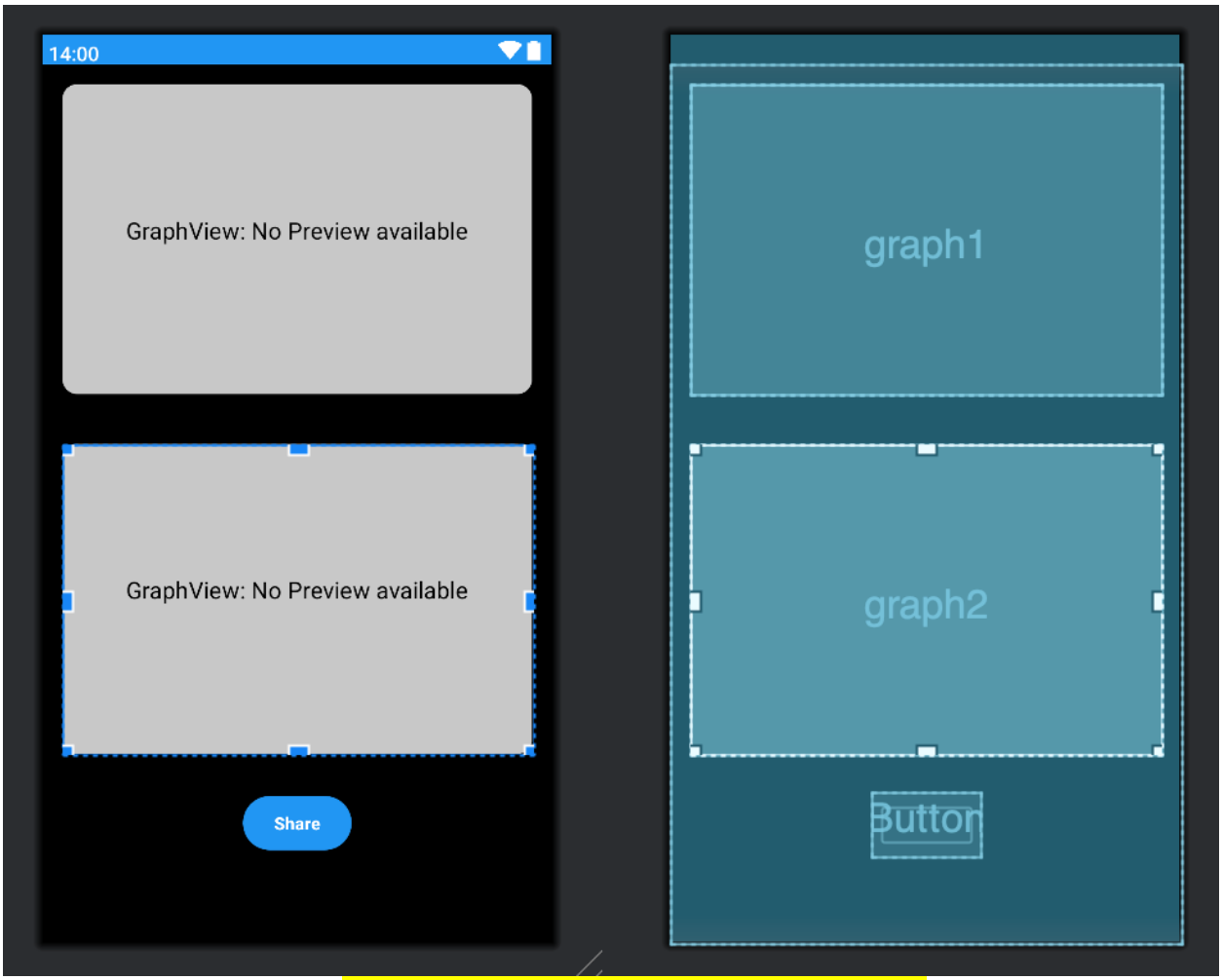
3. Forgot Password Activity Layout



4. Main Activity Layout



5. Graph Activity Layout




Please note the GraphView is Scrollable

---

## 8. Activities

### 1. RegisterActivity:

- **Purpose**: Allows users to register with an email and password, storing their credentials securely in Firebase.
- **Layer Interaction**:
  - **Presentation Layer**: Captures user input for email and password.
  - **Business Logic Layer**: Validates input (e.g., proper email format) and forwards the data for account creation.

- **Data Layer**: Interfaces with Firebase Authentication to create a new user.

- **Implementation**:

  - **UI Components**:
    - EditTexts for username, email, and password.
    - Buttons for registering and navigating back.
  - **Firebase Integration**:
    - Use FirebaseAuth.createUserWithEmailAndPassword(email, password) to create a new account.
    - Attach success and failure listeners to handle the result.
  - **Validation**:
    - Validate email format using android.util.Patterns.EMAIL_ADDRESS.
    - Ensure password strength by checking its length and complexity.
  - **Error Handling**:
    - Display Toast messages for errors like invalid email or password too short.
    - Navigate back to LoginActivity on successful registration.

2. **LoginActivity**:

   - **Purpose**: Authenticates users and redirects them to the main application interface.
   - **Layer Interaction**:

     - **Presentation Layer**: Collects login credentials and displays success or failure messages.
     - **Business Logic Layer**: Passes the credentials to Firebase for verification and handles the response.
     - **Data Layer**: Communicates with Firebase to authenticate the user.

   - **Implementation**:

     - **UI Components**:
       - EditTexts for email and password.
       - Buttons for login and navigation to registration or password reset.
     - **Firebase Authentication**:

    - Use FirebaseAuth.signInWithEmailAndPassword(email, password) for authentication.
    - Handle success by redirecting to MainActivity.
- **Error Handling**:
    - Provide specific error messages for cases like incorrect credentials or network failure.
- **Navigation**:
    - Use explicit intents to navigate to ForgotPasswordActivity or RegisterActivity.

3. **ForgotPasswordActivity:**

- **Purpose**: Sends a password reset email to the user.
- **Layer Interaction**:
    - **Presentation Layer**: Captures the user's email address.
    - **Business Logic Layer**: Validates the email and sends a password reset request.
    - **Data Layer**: Utilizes Firebase's password reset functionality.
- **Implementation**:

    1. **UI Components**:
        - EditText for email.
        - Buttons for sending reset instructions and canceling.
    2. **Firebase Integration**:
        - Use FirebaseAuth.sendPasswordResetEmail(email) to initiate the reset process.
        - Handle success by showing a confirmation Toast.
    3. **Validation**:
        - Check email format and show errors for invalid or empty inputs.
    4. **Navigation**:
        - Redirect back to LoginActivity after successful email dispatch.

4. **MainActivity:**

    - **Purpose**: Captures and logs real-time sensor data into CSV files.
    - **Layer Interaction**:
        - **Presentation Layer**: Displays real-time sensor data in the UI and allows user interaction for toggling sensors.
        - **Business Logic Layer**: Processes sensor events and forwards them for logging.

- **Data Layer**: Writes sensor data to CSV files and handles file storage.
- **Implementation**:
  1. **Sensor Management**:
     - Register accelerometer and gyroscope listeners using SensorManager.
     - Use SensorEventListener to capture sensor events.
  2. **Data Logging**:
     - Write sensor data to a CSV file using FileOutputStream.
     - Format the data for readability, including timestamps and activity labels.
  3. **Permissions**:
     - Request runtime permissions for location and storage.
  4. **UI Components**:
     - Display real-time sensor readings using TextView.
     - Provide toggles to start/stop recording.

5. **GraphActivity:** (Please note graph view is scrollable)

- **Purpose**: Parses CSV files and visualizes the data in a graphical format.
- **Layer Interaction**:
  - **Presentation Layer**: Provides a graphical representation of the data for user analysis.
  - **Business Logic Layer**: Processes CSV data and prepares it for visualization.
  - **Data Layer**: Reads CSV files stored locally.
- **Implementation**:

  1. **CSV Parsing**:
     - Use libraries like OpenCSV to read sensor data from the file.
  2. **Graph Rendering**:
     - Use GraphView to plot the accelerometer and gyroscope data.
     - Customize graph axes and legends for clarity.
  3. **UI Components**:
     - Include zoom and pan functionalities for detailed data analysis.
     - Add buttons for sharing or exporting graphs.
  4. **Error Handling**:
     - Check for empty or corrupt files and display appropriate messages.

---

## 9. Configuration and Build Gradle

Dependencies:

- Firebase Authentication and Realtime Database.
- jjoe64:graphview for data visualization.

# build.gradle (Module: app)

```gradle
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.google.services)
}

android {
    namespace = "com.example.sensorrecorderapp"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.sensorrecorderapp"
        minSdk = 26
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
}

configurations.all {
    exclude(group = "com.android.support", module = "support-compat")
    exclude(group = "com.android.support", module = "support-v4")
}

dependencies {

    implementation(libs.appcompat) {
        exclude(group = "com.android.support")
    }
```

```
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
    implementation(libs.core) {
        exclude(group = "com.android.support", module = "support-compat")
    }
    implementation("com.google.android.gms:play-services-location:21.0.1")
    implementation("androidx.recyclerview:recyclerview:1.3.1")
    implementation("androidx.wear.tiles:tiles-material:1.2.0")
    implementation("com.google.firebase:firebase-auth:23.1.0")
    implementation("com.google.android.gms:play-services-maps:18.1.0")
    implementation("androidx.cardview:cardview:1.0.0")
    implementation("androidx.annotation:annotation:1.6.0")
    implementation("com.google.code.gson:gson:2.10.1")
    implementation("androidx.documentfile:documentfile:1.0.1")
    implementation("com.opencsv:opencsv:5.7.1")
    implementation("com.google.firebase:firebase-firestore:23.0.3")
    implementation(platform("com.google.firebase:firebase-bom:33.7.0"))
    implementation("com.jjoe64:graphview:4.2.2")
}
```

## build.gradle (Project level)

```gradle
buildscript {
    repositories {
        google()
        mavenCentral()
    }

}

plugins {
    alias(libs.plugins.android.application) apply false
    alias(libs.plugins.google.services) apply false
    alias(libs.plugins.jetbrains.kotlin) apply false
}
```

## Setting.gradle

```gradle
pluginManagement {
    repositories {
        google {
            content {
                includeGroupByRegex("com\\.android.*")
                includeGroupByRegex("com\\.google.*")
```

```
                includeGroupByRegex("androidx.*")
            }
        }
        mavenCentral()
        gradlePluginPortal()
    }
}
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
    }
}

rootProject.name = "SensorRecorderApp"
include(":app")
```
```

**Permissions:**

- Add permissions for sensor access and file storage in the `AndroidManifest.xml`.

**AndroidManifest**

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

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:largeHeap="true"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.SensorRecorderApp">

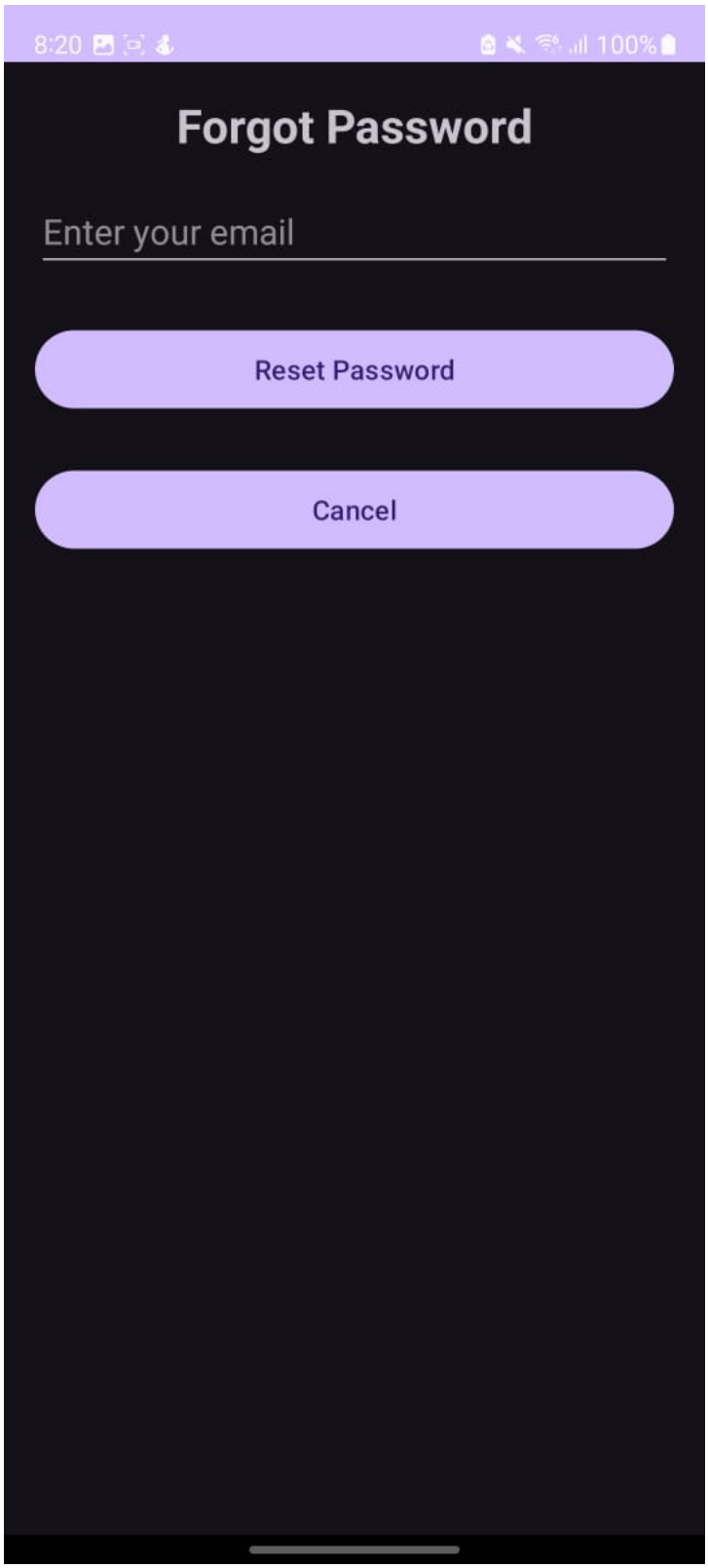
    <activity
        android:name=".ForgotPasswordActivity"
        android:exported="false"
        android:parentActivityName=".LoginActivity">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.sensorrecorderapp.LoginActivity" />
    </activity>
```
```

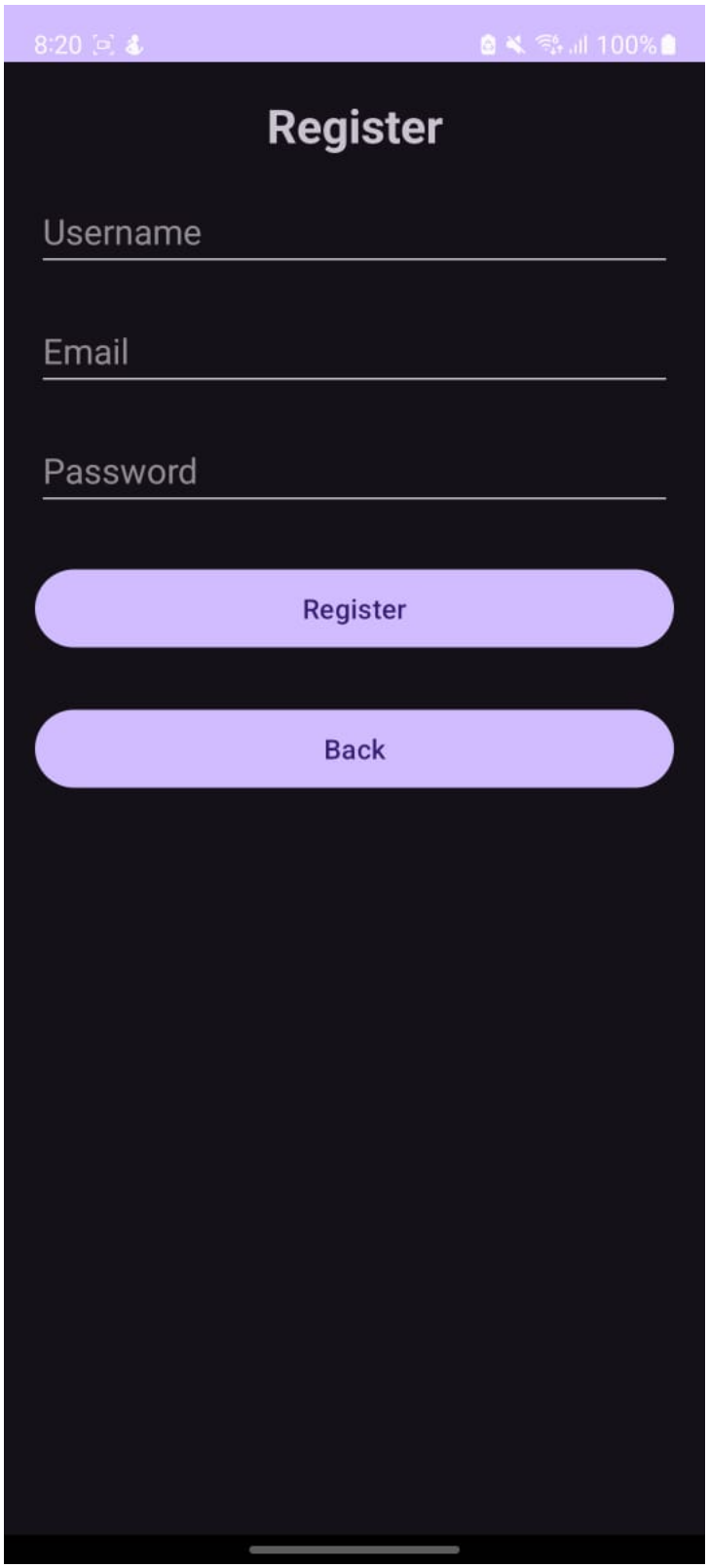
```
    <activity
        android:name=".LoginActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".RegisterActivity"
        android:exported="false"
        android:parentActivityName=".LoginActivity">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.sensorrecorderapp.LoginActivity" />
    </activity>
    <activity
        android:name=".MainActivity"
        android:exported="true">
    </activity>
    <activity
        android:name=".GraphActivity"
        android:exported="true"
        android:parentActivityName=".MainActivity">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.sensorrecorderapp.MainActivity" />
    </activity>

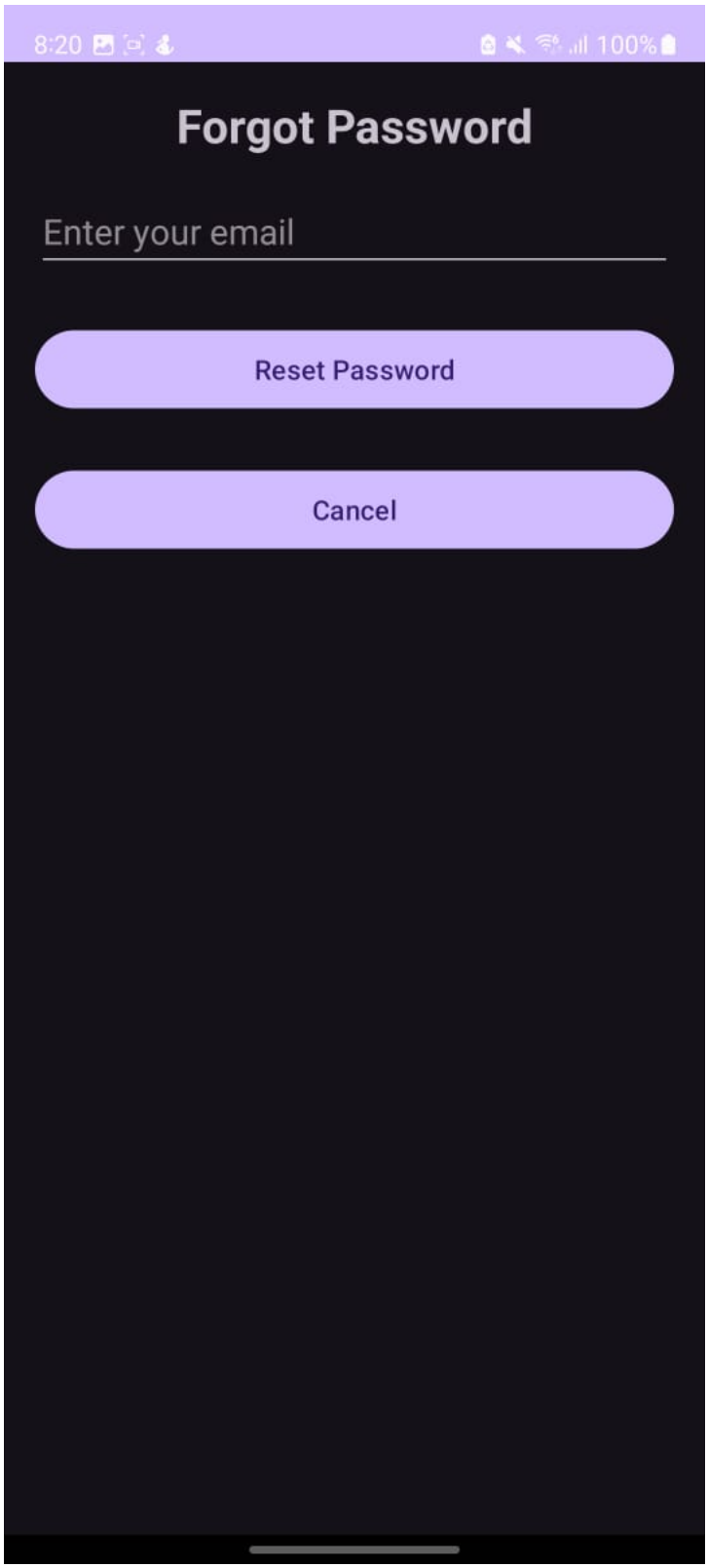
    <meta-data
        android:name="preloaded_fonts"
        android:resource="@array/preloaded_fonts" />
</application>
</manifest>
```

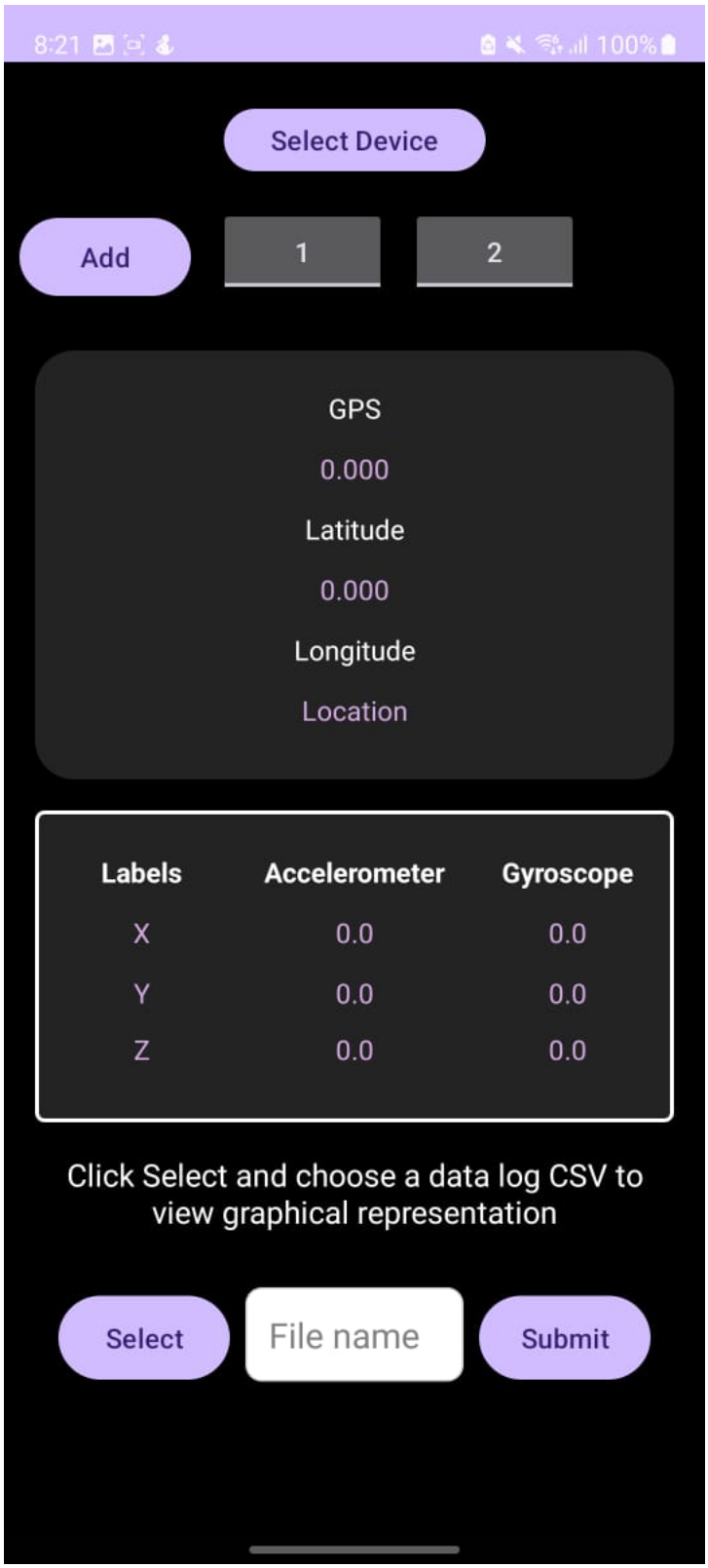
---

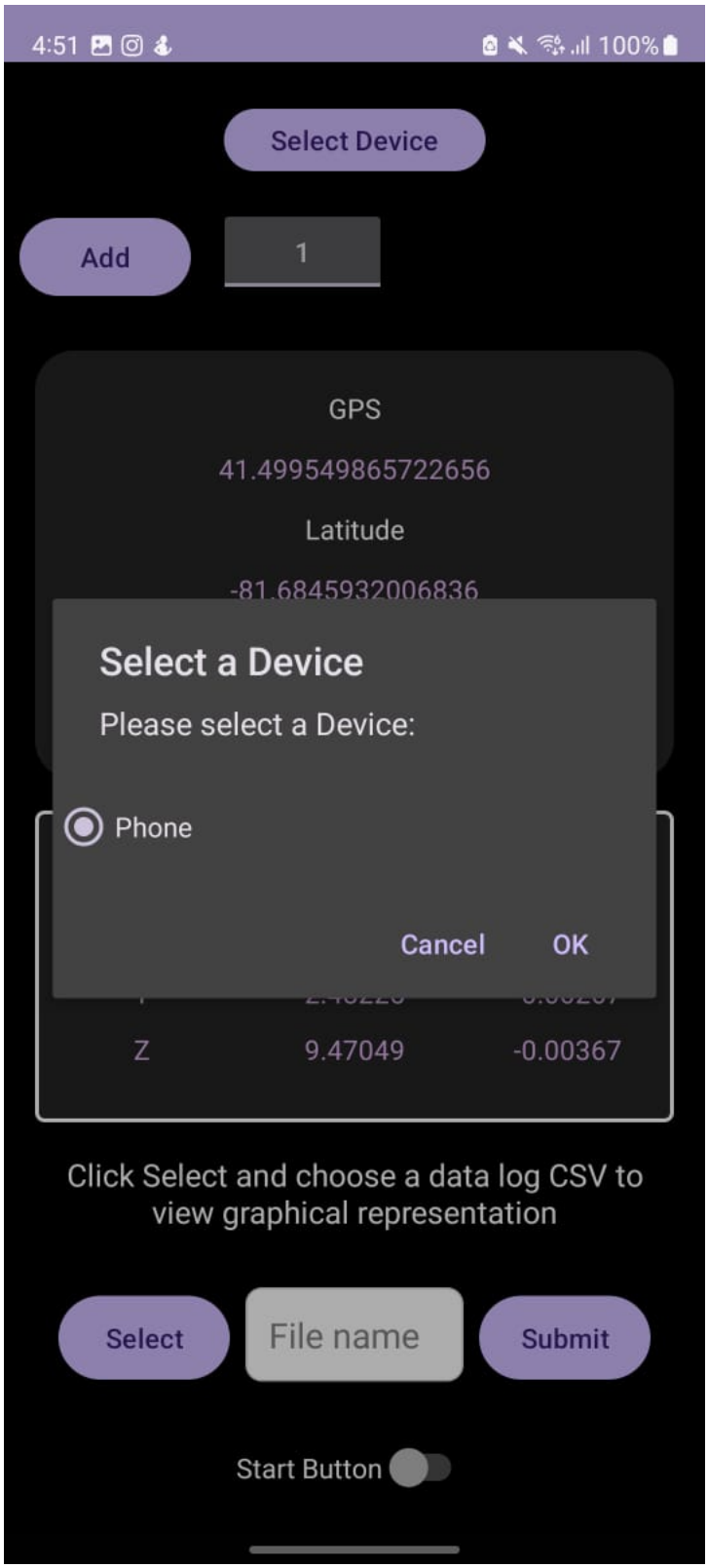
## 10. Application Screenshots

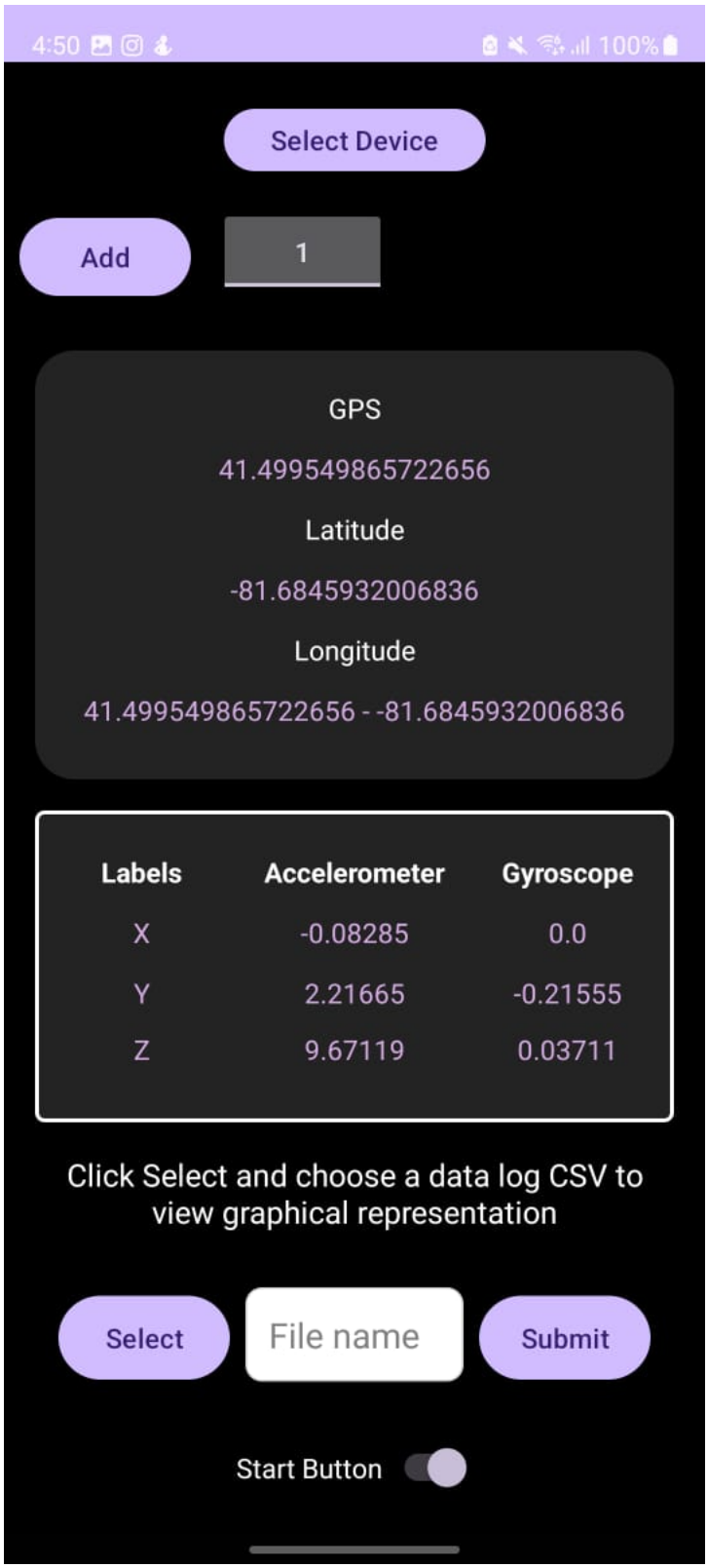

8:20
100%
Forgot Password
Enter your email
Reset Password
Cancel

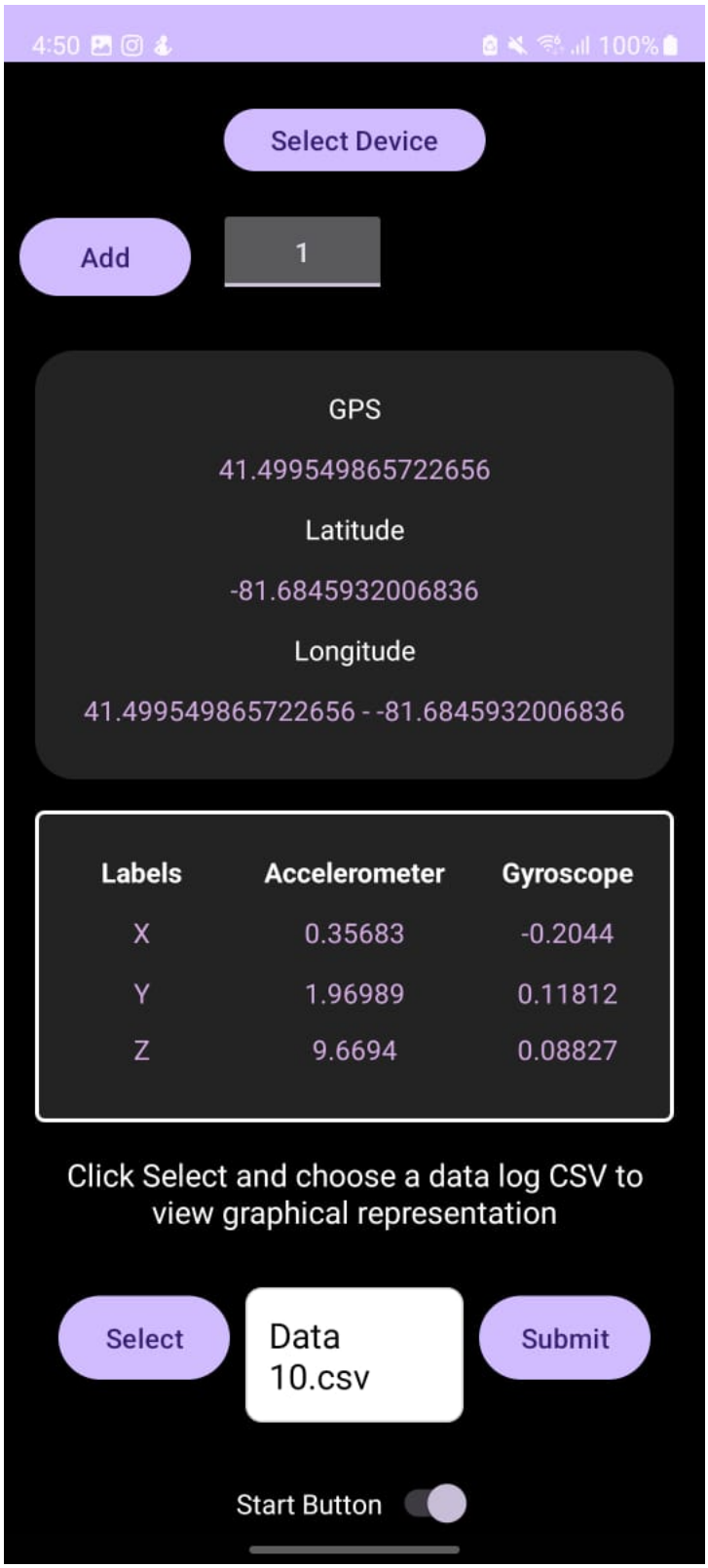

8:20
100%
Register
Username
Email
Password
Register
Back

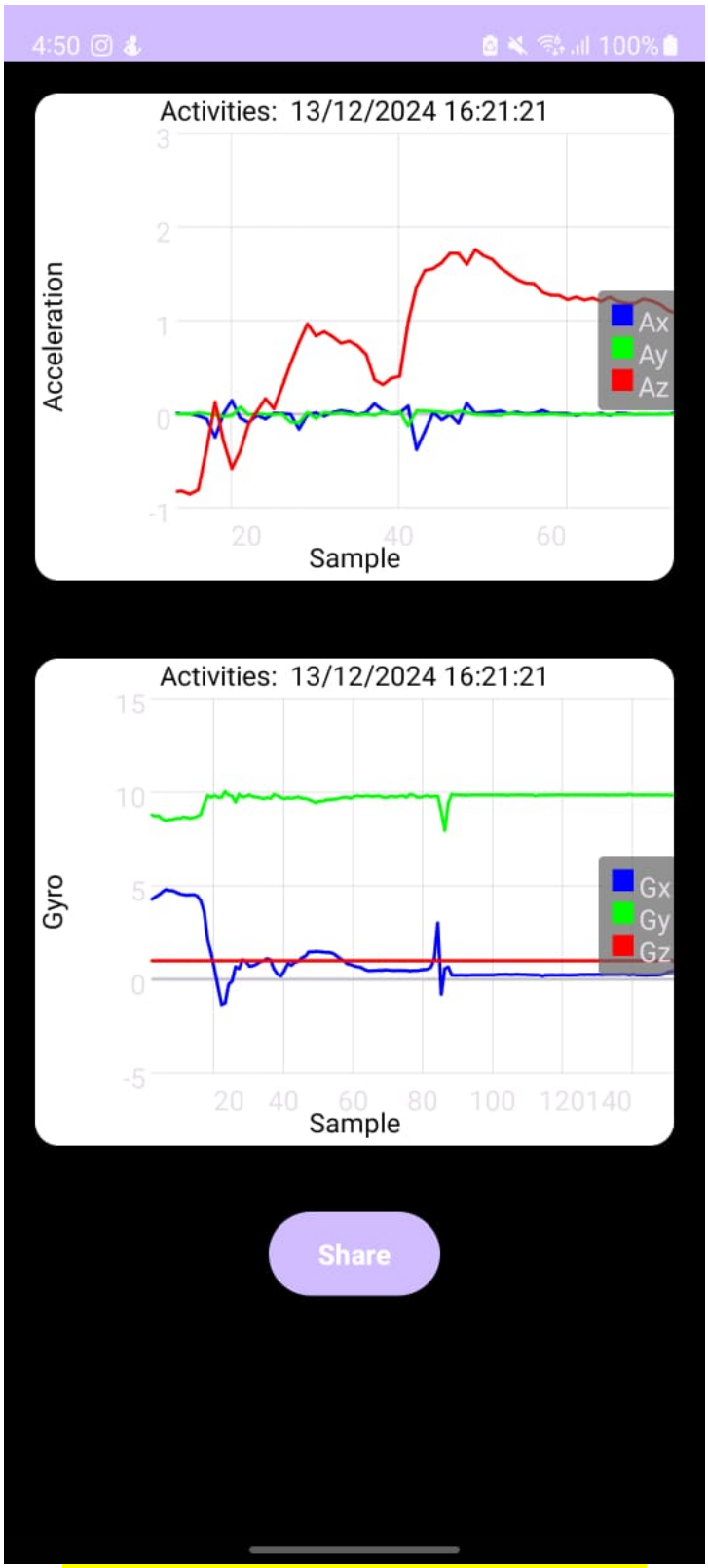

8:20
100%
Forgot Password
Enter your email
Reset Password
Cancel


8:21
100%
Select Device
Add
1
2
GPS
0.000
Latitude
0.000
Longitude
Location
Labels
Accelerometer
Gyroscope
X
0.0
0.0
Y
0.0
0.0
Z
0.0
0.0
Click Select and choose a data log CSV to view graphical representation
Select
File name
Submit


4:51
100%
Select Device
Add
1
GPS
41.499549865722656
Latitude
-81.6845932006836
Select a Device
Please select a Device:
Phone
Cancel
OK
Z
9.47049
-0.00367
Click Select and choose a data log CSV to view graphical representation
Select
File name
Submit
Start Button


4:50
100%
Select Device
Add
1
GPS
41.499549865722656
Latitude
-81.6845932006836
Longitude
41.499549865722656 - -81.6845932006836
Labels
Accelerometer
Gyroscope
X
-0.08285
0.0
Y
2.21665
-0.21555
Z
9.67119
0.03711
Click Select and choose a data log CSV to view graphical representation
Select
File name
Submit
Start Button


4:50
100%
Select Device
Add
1
GPS
41.499549865722656
Latitude
-81.6845932006836
Longitude
41.499549865722656 - -81.6845932006836
Labels
Accelerometer
Gyroscope
X
0.35683
-0.2044
Y
1.96989
0.11812
Z
9.6694
0.08827
Click Select and choose a data log CSV to view graphical representation
Select
Data 10.csv
Submit
Start Button




Please note the GraphView is Scrollable

## 11. Challenges Faced

- Firebase Configuration: Issues with setting up and synchronizing Firebase with the app.
- CSV Logging: Managing efficient data logging without performance lags.
- Graph Rendering: Handling large datasets in graph visualization.
- UI Responsiveness: Adapting the UI for various screen sizes.

---

## 12. Conclusion

The Sensor Recorder Application effectively demonstrates the integration of mobile sensors with Firebase backend services. It provides a practical tool for recording and visualizing sensor data, suitable for both educational and analytical use cases. Despite challenges, the application achieves its goals, offering a user-friendly and functional experience.