

APPLIANCE CONTROLLING

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Describe how you will implement Google Play Services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Implement UI for Each Activity and Fragment](#)

[Task 3: Implement Functionalities along with importing libraries](#)

[Task 4: Observe the Data Flow and Logic Checking](#)

[Task 5: Check on Simulator and build SIGNED APK](#)

GitHub Username: MrinaliniPal

Appliance Controlling

Description

Nowadays saving energy is an important factor. To conserve energy it is very important for us to take some necessary steps. In room, people forget to switch off the ac, fan or any other electrical appliances which are not required for that time being. This, though seems to be very casual, leads to a high misuse of electrical energy.

To eradicate this problem, one home automation system is decided to be developed which will help people to directly control any electrical appliance via mobile. Since mobile is portable, even if people forget to switch off electrical appliances at home or office, they can turn them off using their mobile from anywhere through valid internet connection. If there is no internet connectivity, the app can use Bluetooth and Wifi connection for local interaction.

Intended User

- Anyone who has an Android-phone.

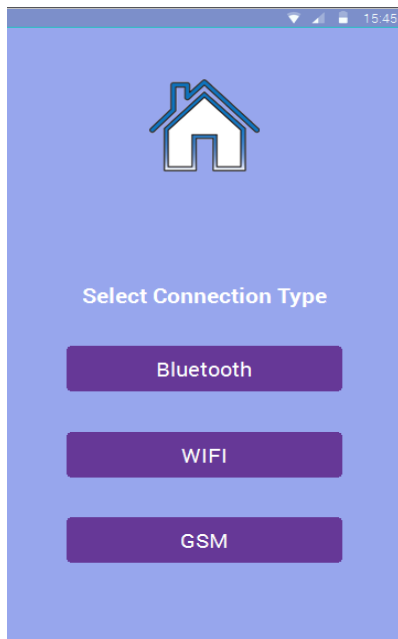
Features

The main features of this app are:

- Bluetooth and Wifi can be used at home or office for local interaction, GSM can be used anywhere where tower of the telecom operator is available.

User Interface Mocks

Screen 1



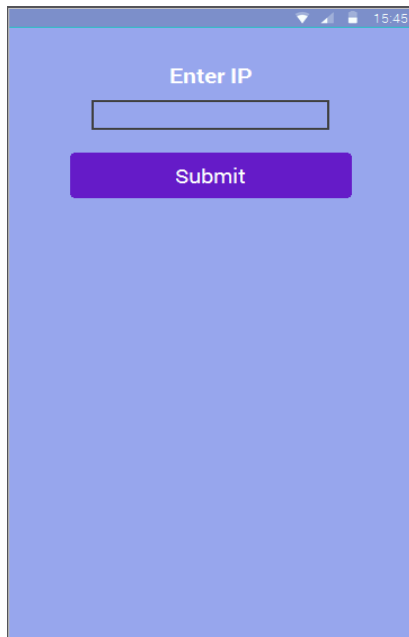
This is the first Page which opens after the user opens the app.

Screen 2



This activity opens when user wants to connect his device using Bluetooth.

Screen 3

A screenshot of a mobile application interface with a light blue background. At the top, there is a status bar with icons for signal, Wi-Fi, and battery, and the time 15:45. Below the status bar, the text "Enter IP" is centered. Underneath, there is a white rectangular input field. Below the input field, there is a purple rectangular button with the text "Submit" in white.

This activity opens when user wants to connect his device using WiFi.

Screen 4



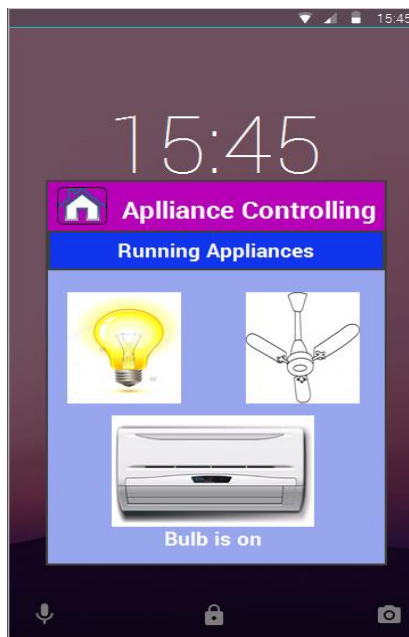
This activity opens automatically after choosing GSM option of Connectivity and also after pairing with Bluetooth or WiFi.

Screen 5



This activity shows the appliances that are in use.

Screen 6



This is the widget which shows the user the appliances That are in use.

Key Considerations

How will your app handle data persistence?

App will store data in

- Firebase for online and offline data persistence as Firebase has this inbuilt feature.

Describe any corner cases in the UX.

- When no Bluetooth device is paired: Show Message
- Wifi or Bluetooth connection is not setup: Show Message

Describe any libraries you'll be using and share your reasoning for including them.

The libraries which will be used are

- **Firebase** for online and offline data persistence.
- **Fabric** for crash analytics.
- **Android Design Support** to include material design.

Describe how you will implement Google Play Services.

The app will contain google ads and Firebase thereby including Google Play Service.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

Setup the project and configure

- Create skeleton app and link to Github repo.
- Import all the libraries needed.
- Create Database in the Firebase server.
- Create Mockup for the basic idea.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for selecting the mode of connection.
- Build UI for all the other activities required.

Task 3: Implement Functionalities along with importing libraries

- Impose the Functions on the built UI.

- Use the functions from imported libraries.
- Control the errors and exceptions.

Task 4: Observe the Data Flow and Logic Checking

- Observe how the specific data gets parsed from one function to other function and from one activity to another activity.
- The data should follow the logic that get implemented as decided.

Task 5: Check on Simulator and build SIGNED APK

- Check the full app and build the SIGNED APK of it.
- Control the errors and exceptions.