Chapter 1

INTRODUCTION

Most of the Modern world runs on Electricity. Electric power and electronics are at the heart of many modern technologies. So, there is tremendous amount of work involved in generation to distribution of electricity. The maintenance of this system is a very risky procedure, which has to be looked after with much care and safety.

In this chapter, we introduce certain aspects related to it, such as electric power distribution, substation and line clearing system. We also look upon the android technology and python along with its few features which will be implemented in our project.

1.1 Electric Power Distribution

Electric power distribution is the final stage in the delivery of electric power, it carries electricity from the transmission system to individual consumers. Distribution substations connect to the transmission system and lower the transmission voltage to medium voltage ranging between 2 kV and 35 kV with the use of transformers.

Distribution of power is an important part of power system, as well as a giant system of complex uncertainty, which is a line of transmitting power from the distribution transformer to the power point, so as to supply power for each distribution substation of cities and various power loads.

➢ **Power Distribution System**

A distribution substation is located near or inside city/town/village/industrial area. It receives power from a transmission network. The high voltage from the transmission line is then stepped down by a step-down transformer to the primary distribution level voltage. Primary distribution voltage is usually 11 kV, but can range between 2.4 kV to 33 kV depending upon region or consumer.

➢ **A typical power distribution system consists of**

→Distribution substation
→Feeders
→Distribution Transformers
→Distributor conductors
→Service mains conductors

- **Distribution feeders:** The stepped-down voltage from the substation is carried to distribution transformers via feeder conductors.
- **Distribution transformer**: A distribution transformer, also called as service transformer, provides final transformation in the electric power distribution system.
- **Distributor conductor**: Output from a distribution transformer is carried by distributor conductor. Tapping's are taken from a distributor conductor for power supply to the end consumers.
- **Service Conductor**: These run from the service point to the service disconnecting means (the service equipment, not the meter). Service-entrance conductors can enter an installation from overhead (service drop) or underground (service lateral).

Fig 1.1 shows the block diagram of current distribution system.

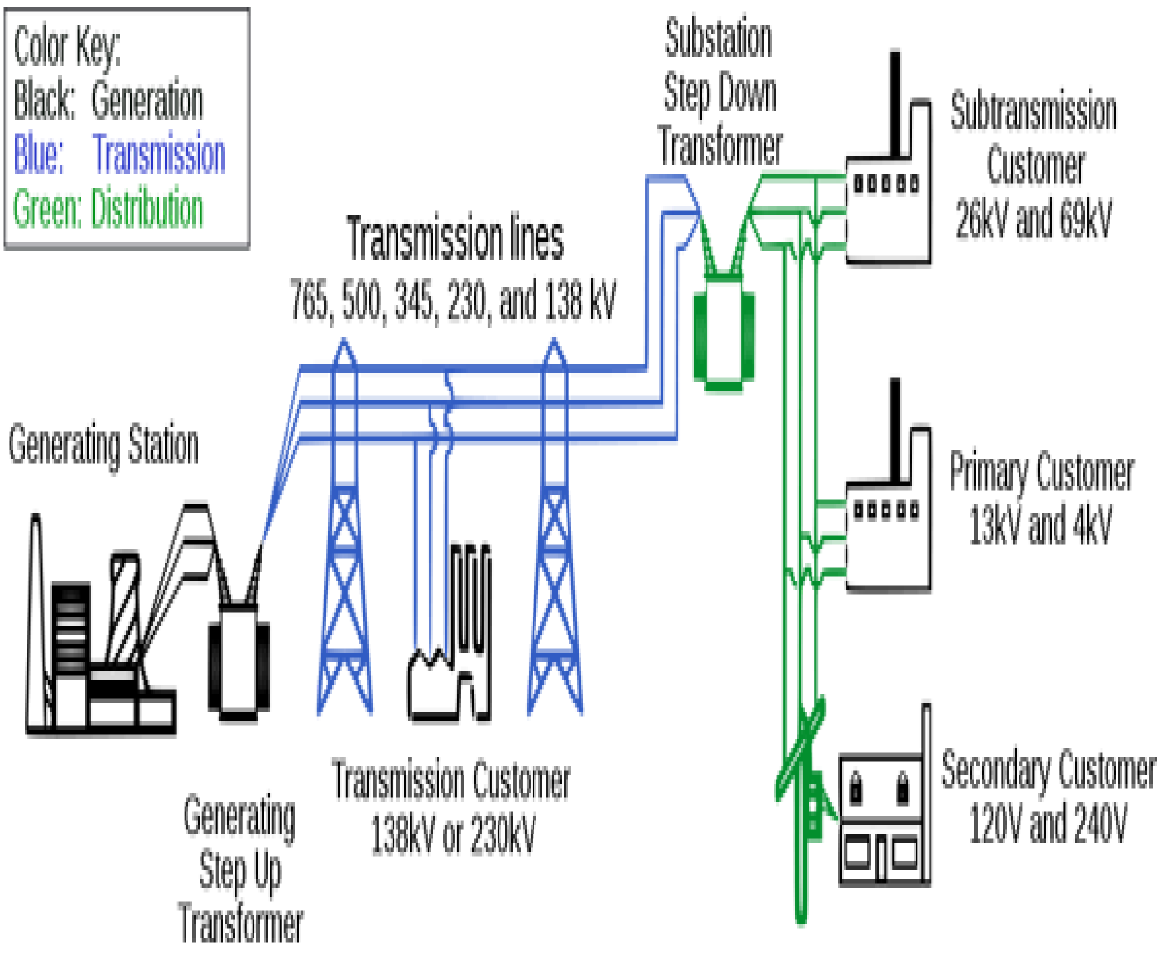


Fig 1.1: Current Distribution System

1.1.1 Substation

A substation is a part of an electrical generation, transmission, and distribution system. Substations transform voltage from high to low, or the reverse, or perform any of several other important functions. Between the generating station and consumer, electric power may flow through several substations at different voltage levels. A substation may include transformers to change voltage levels between high transmission voltages and lower distribution voltages, or at the interconnection of two different transmission voltages.

Substations may be owned and operated by an electrical utility, or may be owned by a large industrial or commercial customer. Generally, substations are unattended, relying on SCADA for remote supervision and control.

The word substation comes from the days before the distribution system became a grid. As central generation stations became larger, smaller generating plants were converted to distribution stations, receiving their energy supply from a larger plant instead of using their own generators. The first substations were connected to only one power station, where the generators were housed, and were subsidiaries of that power station.

Fig 1.2 shows a typical substation.

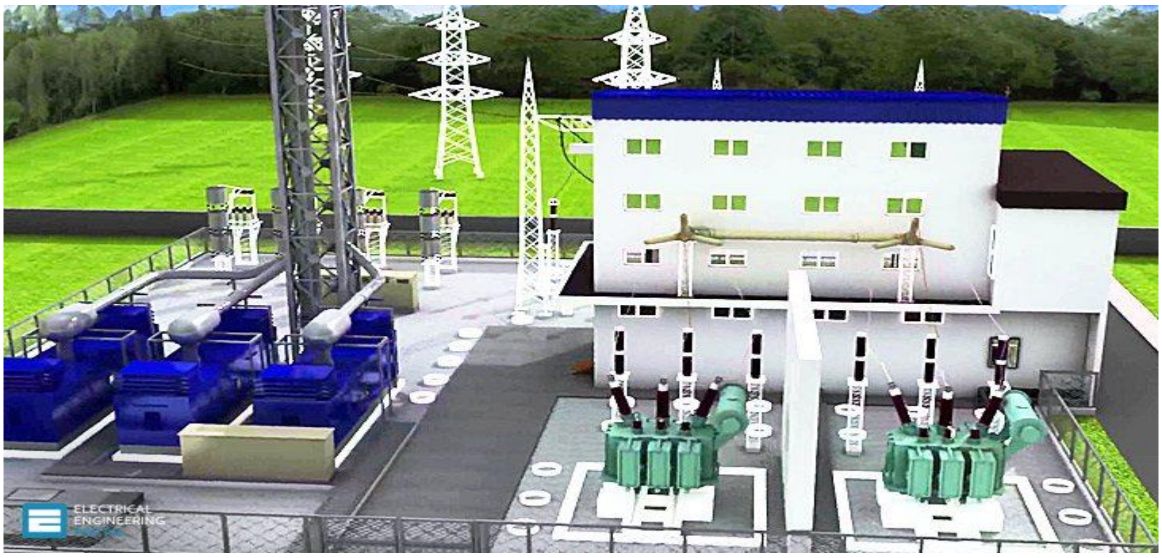


Fig 1.2: Image of a Substation

1.1.2 Elements of a Substation

Substations generally have switching, protection and control equipment and transformers. In a large substation, circuit breakers are used to interrupt any short circuits or overload currents that may occur on the network. Smaller distribution stations may use recloser circuit breakers or fuses for protection of distribution circuits. Substations themselves do not usually have generators, although a power plant may have a substation nearby. Other devices such as capacitors, voltage regulators, and reactors may also be located at a substation.

Substations may be on the surface in fenced enclosures, underground, or located in special-purpose buildings. High-rise buildings may have several indoor substations. Indoor substations are usually found in urban areas to reduce the noise from the transformers, for reasons of appearance, or to protect switchgear from extreme climate or pollution conditions.

A grounding (earthing) system must be designed. The total ground potential rise, and the gradients in potential during a fault (called touch and step potentials), must be calculated to protect passers-by during a short-circuit in the transmission system. Earth faults at a substation can cause a ground potential rise. Currents flowing in the Earth's surface during a fault can cause metal objects to have a significantly different voltage than the ground under a person's feet; this touch potential presents a hazard of electrocution. Where a substation has a metallic fence, it must be properly grounded to protect people from this hazard.

The main issues facing a power engineer are reliability and cost. A good design attempts to strike a balance between these two, to achieve reliability without excessive cost. The design should also allow expansion of the station, when required.

1.2 Line Clearing System

Line Clearing System is the module which the substation handle under them with the help of section officers and many linemen to handle efficient power distribution.

The various components involved in line clearing system are explained below as shown in fig 1.3.

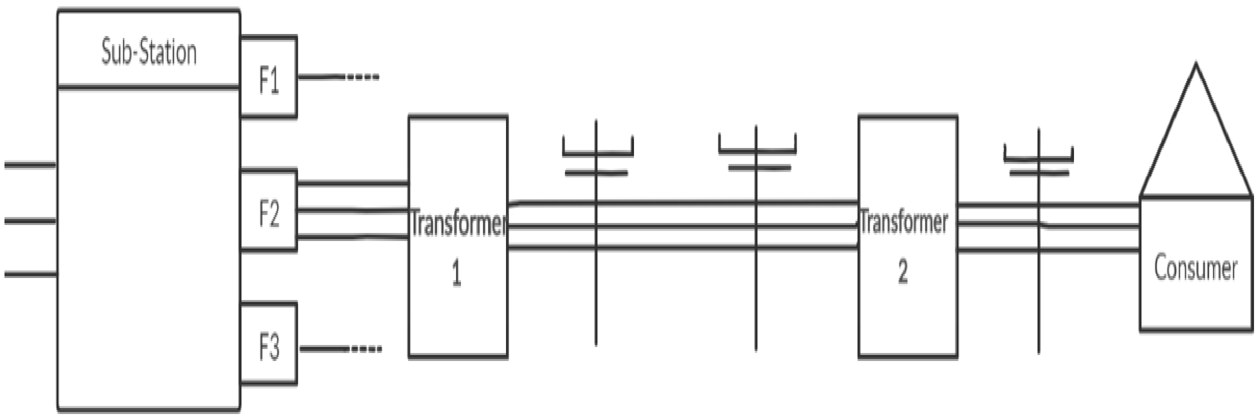


Fig 1.3: Pictorial representation of Line Clearing System

- **Sub-Station:** A substation is a part of an electrical generation, transmission, and distribution system. Substations transform voltage from high to low, or the reverse, or perform any of several other important functions. There are one or many Section Officers present under one Sub-Station.
- **Feeder:** F1, F2 and F3 are Feeders. In electric power distribution, it is a voltage power line, transferring power from a distribution substation to the distribution transformers.
- **Section Officer:** A Section Officer has one or more Feeders under his control and can communicate with Sub-Station operator to switch it off.
- **Transformer:** A Transformer is a passive electrical device that transfers electrical energy between two or more circuits.
- **Transmission Pole:** A Transmission Pole is a column or post used to support overhead power lines (electrical cable) i.e. equipment related to transformers and street lights.
- **Consumer:** One who consumes the electricity, can be primary or secondary consumer.

The consumer on complaining of the power supply being cut, the linemen verify the flow of electricity in the nth transformer if absent, the n-1th transformer is checked and so on. Where there is no flow detected, he asks the section officer to inform the substation to cut down the power supply in that particular feeder to carry out the required work.

1.3 Developments in Android Technology

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google.

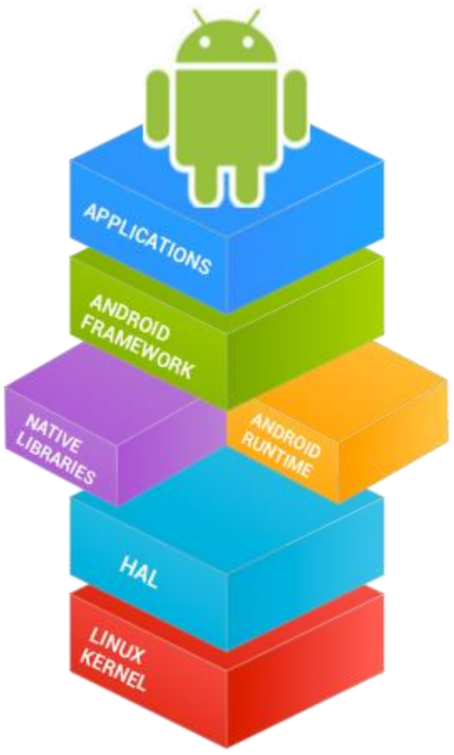


Fig 1.4: Android Application Stack

As shown in fig 1.4, Android delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications. Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It was built to be truly open. Android is built on the open Linux Kernel.

On a basic level, Android is a distribution of Linux that includes a Java Virtual Machine (JVM), with Java being the preferred programming language for most Android applications. The Android Software Development Kit (SDK) includes a debugger, libraries, a handset emulator, documentation, sample code and tutorials.

Some of the developments in Android technology are listed below:

- **Interface**

Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard Game controllers and full-size physical keyboards are supported via Bluetooth or USB. The response to user input is designed to be immediate and provides a fluid touch interface, often using the vibration capabilities of the device to provide haptic feedback to the user. Internal hardware, such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented or allowing the user to steer a vehicle in a racing game by rotating the device, simulating control of a steering wheel.

- **Memory Management**

Since Android devices are usually battery-powered, Android is designed to manage processes to keep power consumption at a minimum. When an application is not in use the system suspends its operation so that, while available for immediate use rather than closed, it does not use battery power or CPU resources Android manages the applications stored in memory automatically: when memory is low, the system will begin invisibly and automatically closing inactive processes, starting with those that have been inactive for the longest amount of time. Life hacker reported in 2011 that third-party task killer applications were doing more harm than good.

- **Firebase**

Firebase is a Backend-as-a-Service — BaaS — that started as a YC11 start-up and grew up into a next-generation app-development platform on Google Cloud Platform. Firebase frees developers to focus crafting fantastic user experiences. You don’t need to manage servers. You don’t need to write APIs. Firebase is your server, your API and your data store, all written so generically that you can modify it to suit most needs. Yeah, you’ll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can’t be everything to everybody. But it gets pretty close.

- **Method**

The Software is implemented using Android where we focus on two modules, Station officer and Substation operator, both having different interfaces, interacting with each other with communication message to act on. The app is managed to reduce battery power consumption and the space of app is optimized, we will to store the data in the platform provided by Google Firebase for analysis of the data and performance of the app.

1.4 Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as described below and shown next in the architecture diagram.

1. Apps: Your apps live at this level, along with core system apps for email, SMS messaging, calendars, Internet browsing, or contacts.

2. Application Framework: All features of Android are available to developers through application programming interfaces (APIs) written in the Java language. You don't need to know the details of all of the APIs to learn how to develop Android apps, but you can learn more about the following APIs, which are useful for creating apps:

- View System used to build an app's UI, including lists, buttons, and menus.
- Resource Manager used to access to non-code resources such as localized strings, graphics, and layout files.
- Notification Manager used to display custom alerts in the status bar.
- Activity Manager that manages the lifecycle of apps.
- Content Providers that enable apps to access data from other apps.
- All framework APIs that Android system apps use.

3. Libraries and Android Runtime: Each app runs in its own process and with its own instance of the Android Runtime, which enables multiple virtual machines on low-memory devices. Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features that the Java API framework uses. Many core Android system components and services are built from native code that requires native libraries written in C and C++. These native libraries are available to apps through the Java API framework.

4. Hardware Abstraction Layer (HAL): The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module.

5. Linux Kernel: The foundation of the Android platform is the Linux kernel. The above layers rely on the Linux kernel for underlying functionalities such as threading and low-level memory management. Using a Linux kernel enables Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

In the figure 1.5 we discuss architecture of android systems. It contains five layers as given in paragraphs above.

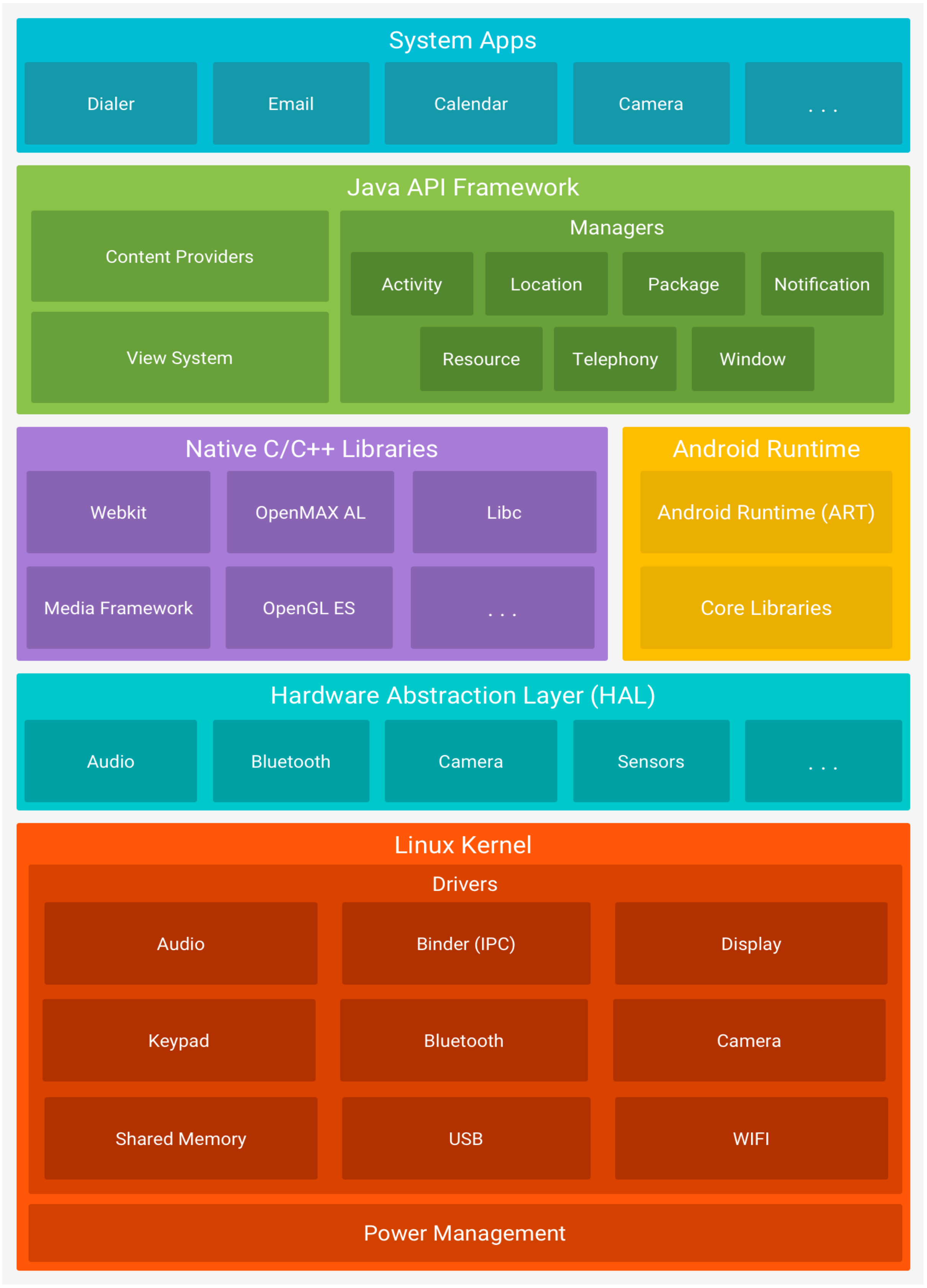


Fig 1.5: Android Architecture

1.4.1 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Google’s Android Operating System, built on JetBrains' IntelliJ IDEA software and designed Specially for Android Development. It is available for download on Windows, MacOS and Linux based Operating Systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version. External projects backport some Java 9 features.

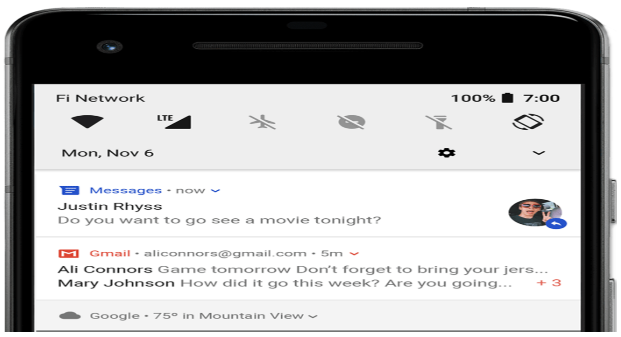
While IntelliJ states that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support).

1.4.2 Android User Interface and Navigation

Android provides a variety of pre-built UI components such as structured layout objects and UI controls that allow you to build the graphical user interface for your app. Android also provides other UI modules for special interfaces such as dialogs,

➢ **Notifications Overview**

A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open your app or take an action directly from the notification.



Fi Network 100% 7:00

Mon, Nov 6

Messages • now

Justin Rhyss

Do you want to go see a movie tonight?

Gmail • aliconnors@gmail.com • 5m

Ali Connors Game tomorrow Don't forget to bring your jers...

Mary Johnson How did it go this week? Are you going... + 3

Google • 75° in Mountain View

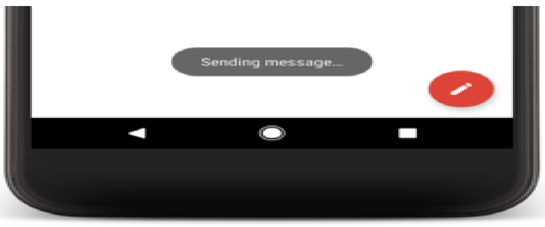
Fig 1.6: Android Notification

Users can swipe down on the status bar to open the notification drawer, where they can view more details and take actions with the notification.

➢ **Toasts overview**

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

For example, clicking Send on an email triggers a "Sending message..." toast, as shown in the following screen capture:



Sending message...

Fig 1.7: Android Toast

1.5 Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum, first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

1.5.1 Flask

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

- **Components**

The micro framework Flask is based on the Pocoo projects Werkzeug and Jinja2.

- **Werkzeug**

Werkzeug is a utility library for the Python programming language, in other words a toolkit for Web Server Gateway Interface (WSGI) applications, and is licensed under a BSD License. Werkzeug can realize software objects for request, response, and utility functions.

It can be used to build a custom software framework on top of it and supports Python 2.7 and 3.5 and later.

- **Jinja**

Jinja, also by Ronacher, is a template engine for the Python programming language and is licensed under a BSD License. Similar to the Django web framework, it handles templates in a sandbox.

➢ **Features**

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

➢ **Example**

The following code shows a simple web application that displays "Hello World!" when visited:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Hello World!"
if __name__ == "__main__":
    app.run()
```

➢ **Deployment**

Setting up Flask applications on Python Anywhere

There are two main ways to set up a Flask application on Python Anywhere:

- First option:

Starting from scratch using our default versions of Flask. Importing a pre-existing app using Manual configuration, and using a virtualenv. The first option works well if you're just playing around and want to throw something together from scratch. Go to the Web Tab and hit add a new Web App, and choose Flask and the Python version you want.

- The Second option is described in more detail below:
  - Getting your code onto Python Anywhere

This guide assumes you've already managed to get your code onto Python Anywhere. Check out the uploading and downloading files guide if you need to. For the purpose of these examples, we ll assume your code lives at /home/yourusername/mysite

  - Check your config

If you're importing existing code, review all of your Flask configuration settings to ensure that they match their new home. For instance, if you've specified a SERVER_NAME in your config, make sure that it matches the web app name.

  - Setting up your virtualenv

Open up a new Bash console from your Dashboard and run

mkvirtualenv --python=/usr/bin/python3.6 my-virtualenv

# use whichever python version you prefer

pip install flask

You'll see the prompt changes from a $ to saying (my-virtualenv)$ -- that's how you can tell your virtualenv is active. Whenever you want to work on your project in the console, you need to make sure the virtualenv is active. You can reactivate it at a later date with

$ workon my-virtualenv

(my-virtualenv)$

You can also install any other dependencies you may have at this point, like Sqlalchemy, using **pip install flask-sqlalchemy**, or **pip install -r requirements.txt**, if you have a requirements.txt

  - Setting up the Web app using Manual configuration

Go to the Web Tab and hit add a new web app. Choose Manual Configuration, and then choose the Python version -- make sure it's the same version as the one you used in your virtualenv

Now go to the Virtualenv section, and enter your virtualenv name: my-virtualenv. When you hit enter, you'll see it updates to the full path to your virtuaelenv (/home/yourusername/.virtualenvs/my-virtualenv).

Finally, go edit the wsgi configuration file. You'll find a link to it near the top of the Web tab.

- Configuring the WSGI file

To configure this file, you need to know which file your flask app lives in. The flask app usually looks something like this:

```
app = Flask(__name__)
```

Make a note of the path to that file, and the name of the app variable (is it "app"? Or "application"?) -- in this example, let's say it's /home/yourusername/mysite/flask_app.py, and the variable is "app".

In your WSGI file, skip down to the flask section, uncomment it, and make it looks something like this:

```
import sys
path = '/home/yourusername/mysite'
if path not in sys.path:
  sys.path.insert(0, path)
from flask_app import app as application
```

Do not use app.run()

When you're using Flask on your own PC, you'll often "run" flask using a line that looks something like this:

```
app.run(host='127.0.0.1',port=8000,debug=True)
```

That won't work on PythonAnywhere -- the only way your app will appear on the public internet is if it's configured via the web tab, with a wsgi file.

More importantly, 'if app.run() gets called when we import your code, it will crash your app', and you'll see a 504 error on your site, as detailed in Flask504Error

Thankfully, most Flask tutorials out there suggest you put the app.run() inside an if __name__ = '__main__': clause, which will be OK, because that won't get run when we import it.

This is ok

```
app = Flask(__name__)
@app.route('/')
def home():
    # etc etc, flask app code
if __name__ == '__main__':
    app.run()
```

This is not ok:

```
app = Flask(__name__)
@app.route('/')
def home():
    # etc etc, flask app code
app.run()
```

- What about my database config?

Many guides on the Internet also suggest you put your database setup inside the __main__ clause, like this:

```
if __name__ == '__main__':
    db.create_all()
    app.run()
```

That will work fine on your machine, but, again, we don't want to use app.run() on Python Anywere. But you'll still want to be able to run db.create_all() every so often on Python Anywhere, to update your database tables or whatever it may be.

Two solutions -- either just run it from a Bash console (remembering to activate your virtualenv first) and then Ctrl+C the flask server when it runs

```
$ workon my-virtualenv
(my-virtualenv)$ python flask_app.py
 * Running on http://127.0.0.1:5000/^C
(my-virtualenv)$
```

Or make a clever little if in your main that checks if it's running on PythonAnywhere, eg: from socket import gethostname

```
[...] if __name__ == '__main__':
    db.create_all()
  if 'liveconsole' not in gethostname():
        app.run()
```

1.5.2 Python Anywhere

PythonAnywhere is an online integrated development environment (IDE) and web hosting service (Platform as a service) based on the Python programming language. Founded by Giles Thomas and Robert Smithson in 2012, it provides in-browser access to server-based Python and Bash command-line interfaces, along with a code editor with syntax highlighting. Program files can be transferred to and from the service using the user's browser. Web applications hosted by the service can be written using any WSGI-based application framework.

PythonAnywhere was created by Resolver Systems, who also produced Resolver One, a Python-based Spread sheet program. On 16 October 2012 the product was acquired by a new company, PythonAnywhere LLP, who will develop and maintain the product in the future, and have taken on the existing development team.

➢ **Features**

- In-browser interactive consoles with code running on hosted servers, shareable between multiple users.
- WSGI-based web hosting, e.g. Django, Flask, web2py
- Support for coding from iPad and other mobile devices.
- Syntax-highlighting in-browser editor.
- Many popular Python modules pre-installed.
- Cron-like scheduled tasks to run scripts at a given time of day.

➢ **Uses**

PythonAnywhere is described as "the simplest way to deploy web2py applications" in the official book on the web framework, is suggested when learning numpy, is deployment platform of choice in Django Girls tutorial, and is recommended as a way of hosting machine learning-based web applications.

1.6 Literature Survey

In order to know the state of the art in electric power distribution, substation and line clearing system, we carried out literature survey. In the following paragraphs we discuss the gist of the same.

- **BESCOM**: Bangalore Electric Supply Company Limited.

By the study of BESCOM website we find that the communication and safety procedures being followed in the line clearing system are still manually carried out. This possesses a possible threat to the personnel involved in the job. There are no procedures implemented with the help of software-based solution.

Generally by conducting a research on Government websites and electric supply companies, we find that there is no software based solution to solve problems in a line clearing system, the companies have not yet opted any IT technology to address their issues, and thus there is a need to implement a solution which solves the problem in a better way.

HESCOM (Hubli Electric Supply Company Limited) has a MOU with our college where in the R&D department conducts projects with us. Mr. Raghavendra Vishwakarma, AEE, R&D Cell, HESCOM has given out this problem as a research project for us. He explained the problem being faced in the communication procedure and gave a brief overview to provide a solution to this through software technology.

On the request made by the HESCOM officials, we surveyed regarding this and did not find much information about it, we met Mr. Arul, AEE, CSD3 Substation Hubli and the respective Section Officer there, discussed about the problems being faced and collected the basic information from there, this motivated us to take up the problem and project on line clearing system which provides an app based communication for different sections of our domain and which also focuses mainly on the standards and safety measures involved during different perspectives.

The following are the IEEE reference papers used in our survey:

1. Lance Allison and Mohammad Muztaba Fuad, “Inter-App Communication between android apps developed in App-Inventor and Android Studio.” Dept of Computer Science, Winston-Salem State University, Winston-Salem, NC, USA, 16 May 2016.
2. Arun Kumar and Supriya Panda, “A Survey: How Python Pitches in IT-World,” Dept of CSE, MRIIRS, Faridabad, 10 October 2019.

1.7 Motivation and Problem Definition

From the literature survey we specify the motivation behind taking up this project and the problem statement associated with it.

- **MOTIVATION**

Human life is very precious. In electrical system maintenance, there can be many casualties involved such as shocks, minor accidents or even loss of life due to improper communication or negligence in action in a line clearing system. It is very much necessary to ensure such misunderstandings will not happen which may lead to critical situations. To ensure this we intend to provide a software solution with an android application.

We can provide an app-based communication tool for different personnel involved in electric supply company to improvise their operation methods and inculcate accurate and quick functionalities.

- **PROBLEM STATEMENT**

FOOL PROOF LINE CLEARING SYSTEM

1.8 Objectives Fulfilled

The objectives of the project which are fulfilled are

- To collect the data from the concerned resources/authority.
- To slice the data, which involves segmentation of data.
- Implement the required functionalities in an android app using Android Studio with the necessary user interface and database.
- Host the app in PythonAnywhere Server.
- Validation, by a Field Expert or the concerned authority of the results obtained.

1.9 SCOPE AND LIMITATIONS

From the objectives mentioned, we specify the scope and its uses/advantages in the present-day industry and possibility of its improvement in the future. We also discuss the limitations the current model has in it.

➢ **SCOPE**

- Automation of the manual procedure of communication between different personnel involved with many substations and the section officers.
- To prevent any type of casualties with respect to line clearing system due to negligence in action or improper action of the concerned authority.
- One can scale the project to many users and the systems concerned with it to function in better way and efficient manner.
- There will be transparency in maintaining all the records associated with the particular system and the data cannot be tampered or modified which helps in correct validation of the data and its results by the field officers.

➢ **LIMITATIONS**

- The system is being implemented currently for single Substation operator, which limits its possibility of scaling up the project.
- The app should be continuously connected to internet.
- The server used is free domain and slower in loading the components in the app.

1.10 Program and Course Outcomes in the project

Now a days the focus is on outcome-based education, the project that are carried out also need to satisfy the program outcome attainment requirements. There are twelve PO’s that need to be mapped. Based on the percentage of objective achieved we get the attainment.

The project titled "Fool Proof Line Clearing System" is a product that helps the personnel involved in line clearing system. The level of mapping is based on the quantum of work as mentioned below.

1. **Engineering knowledge:**

   Java, Android Studio operation, Python, Flask, Database Management System and Software Engineering knowledge.

2. **Problem analysis:**

   Possibility of miscommunication through standard communication procedure and as a added safety measure which ensures double verification for the users in the line clearing system.

3. **Design/development of solutions:**

   Usage of SQL function for data collection and storing, so that standard information is maintained, with some users having privilege to access the data through PythonAnywhere.

4. **Conduct investigations of complex problems.**

   Research on Bloom's levels and efficient security measure.

5. **Modern tool usage:**

   Android Studio, MySQL, PythonAnywhere.

6. **The engineer and society:**

   Educational Institutions, Government and Nongovernment organizations.

7. **Environment and sustainability:**

   Reduced manual work: time saving and easy to handle.

   "Go green" by reducing paper work and maintaining the communication and transaction details digitally.

8. **Ethics:**

   Authentication and authorization of users for using the app. Overcoming the problems of miscommunication among the users.

9. **Individual and teamwork:**

   i. Database design

   ii. UI design

   iii. Android App development

   iv. Hosting in PythonAnywhere

   v. Report and documentation

   Each of the above work is assigned to individuals and each work is completed by taking suggestions from team members.

10. **Communication:**

    Discussion with faculties and resource persons from the company. Project Report writing, presented a Paper in order to demonstrate the proper implementation of the ideas to the outside world.

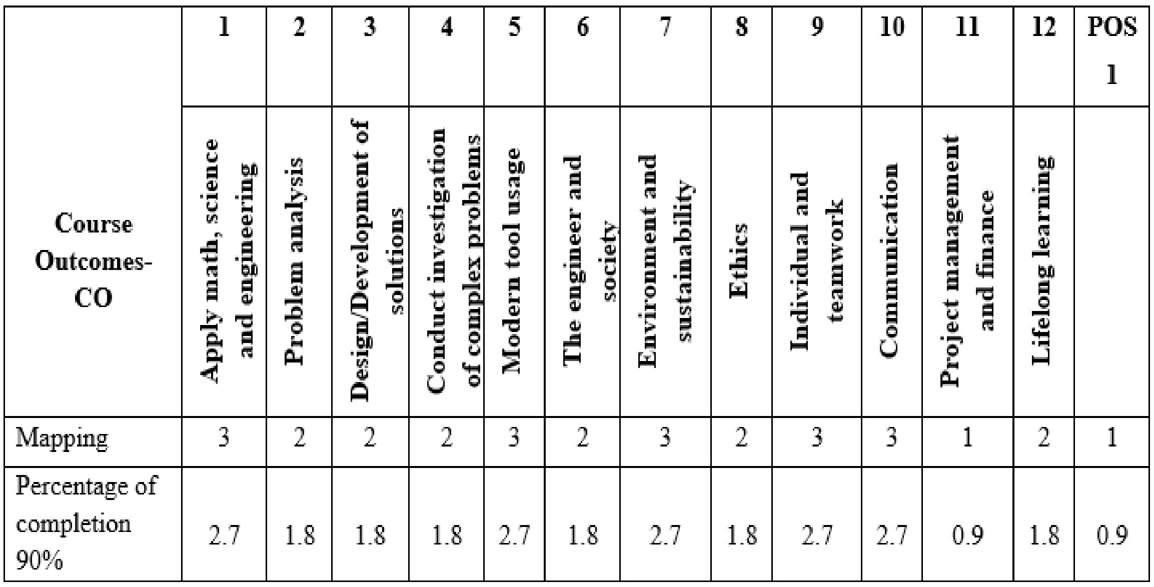
11. **Project management and finance.**

    The project is built using open source tools. So, no financial involvement took place. But if the project is to be sold or commercialized then the financial involvement would certainly take place like. Also, project management activities are taken care.

12. **Life-long learning:**

    Adaptive in nature.

Table 1.1: The project addresses the following PSOs



Course Outcomes-CO	1	2	3	4	5	6	7	8	9	10	11	12	POS 1
	Apply math, science and engineering	Problem analysis	Design/Development of solutions	Conduct investigation of complex problems	Modern tool usage	The engineer and society	Environment and sustainability	Ethics	Individual and teamwork	Communication	Project management and finance	Lifelong learning	
Mapping	3	2	2	2	3	2	3	2	3	3	1	2	1
Percentage of completion 90%	2.7	1.8	1.8	1.8	2.7	1.8	2.7	1.8	2.7	2.7	0.9	1.8	0.9

1.11 Organization of the Report

The project report is organized as mentioned below-

In this chapter 01, we explained the content related to Developments in Android Technology, Android Architecture, Python and its features implemented in this project, Electric Power Distribution, Substation and Line Clearing System form the backbone of this project. Literature Survey is discussed. Motivation and Problem Definition are specified. Objectives fulfilled, scope and limitations are defined.

Chapter 2: This chapter presents the working methodology of our application. In brief we have shown data collected and system requirements and explained in detail how the app works with a diagram.
Chapter 3: In this chapter everything related to system design is specified. Different modules, interfaces are given with the help of use case and sequence diagrams. Code snippets and database details are also specified.
Chapter 4: This chapter covers entirely regarding the testing. Testing objectives and strategies are mentioned. Different testing methods are explained and the test cases are given in the tabular format.
Chapter 5: This chapter is entirely for the snapshots i.e. the screenshots of the app. All the different user interface which the users operate are given in this section of the chapter.
Chapter 6: This chapter concludes the project. The conclusion and future scope of the project is specified here.

Chapter 2

METHODOLOGY

The methodology for the project work “Fool Proof Line Clearing System” is explained here. As we have seen in section 1.2 the abstract working, the components and the terminologies involved in the line clearing system, we now discuss the specific working of this application model in our project.

2.1 Data Collection

Data collection involves collecting the required data from various sources. In our project we collected the data physically, by visiting different places to collect the data regarding section officer details, feeder details, operator details. This helped us in understanding the process also.
Below few photos depict the process of the data collection.

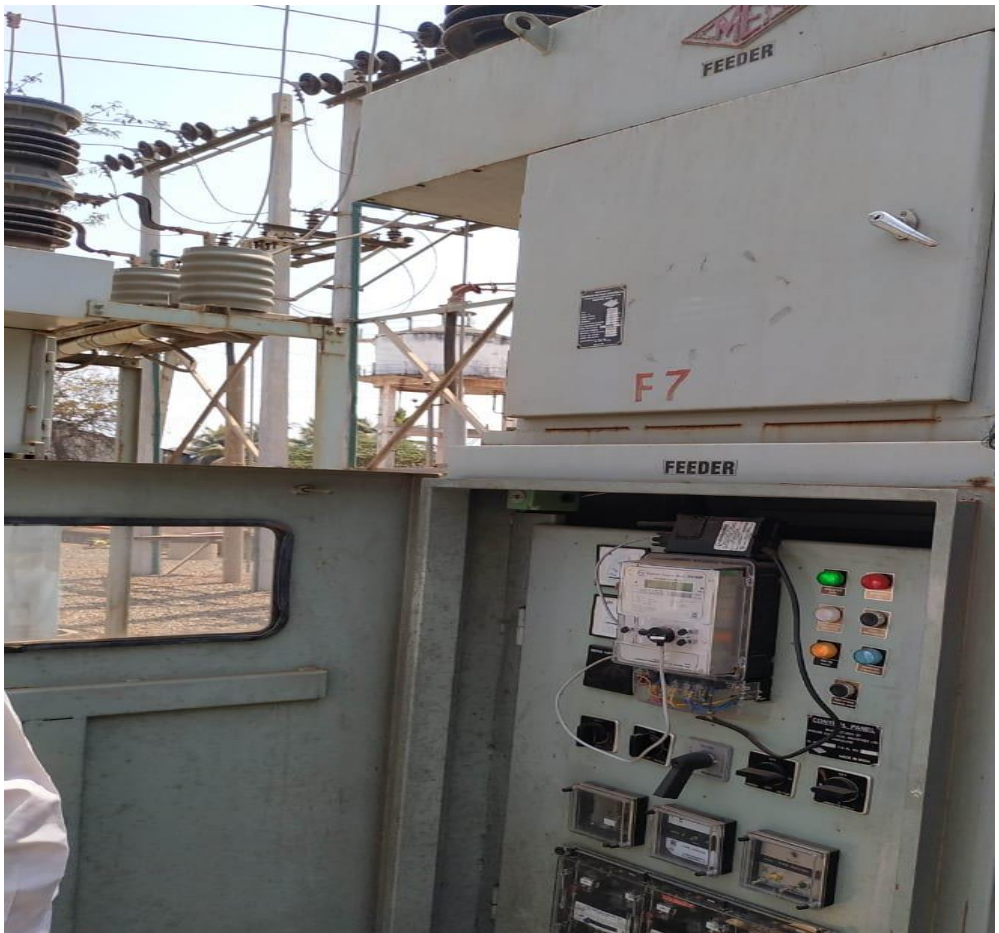


Fig 2.1: Feeder

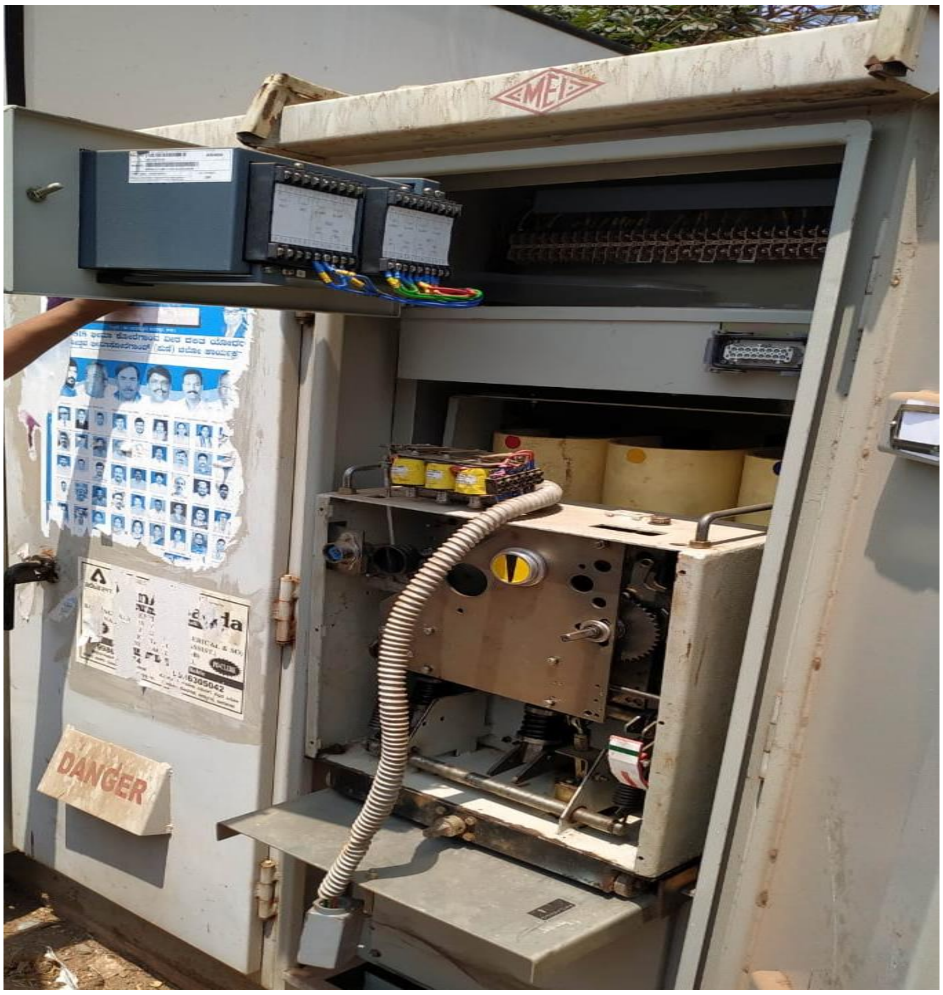


Fig 2.2: Electrical component

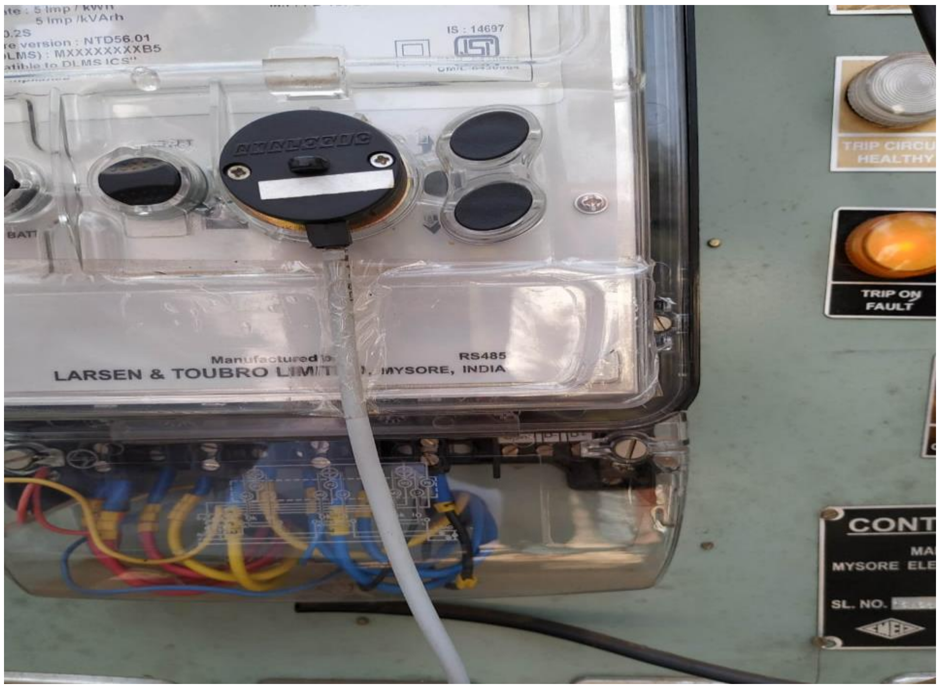


Fig 2.3: Feeder Component

2.2 Working Method

The application consists of components mentioned in line clearing system with different functional modules and interfaces associated with them. We explain below its working. Fig 2.4 shows the Abstract Working of Line Clearing System

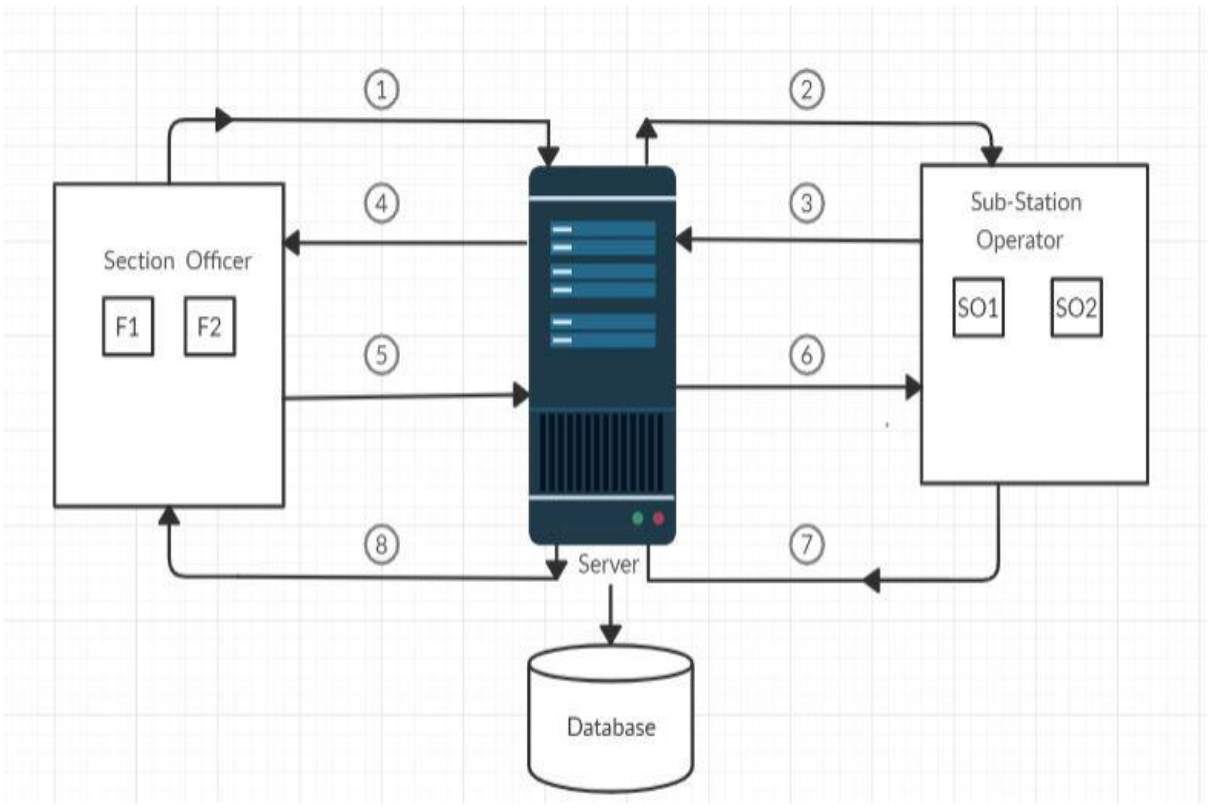


Fig 2.4: Abstract Working of Line Clearing System

There are basically 4 users in this application. Admin, Assistant Executive Engineer (AEE), Section Officer (SO) and Sub-Station Operator (SSO).

- Admin: Admin has the authority to add new users (AEE, SO's and SSO) to the application. He assigns them login credentials for the app. He can view the list of employees and feeders present. He can add new Feeders and assign it to the respective SO. He can also view all the requests made by SO's to the SSO and can keep track of them.
- AEE: He is the person who is the head for SO's and SSO. He can view all the details regarding the SO's and SSO like their names and phone number details and feeders present under each SO. He also has the provision to view the requests made between SO's and SSO and can monitor them easily.

- Section Officer (SO): Each SO present will have one or more feeders under his control which he can request the SSO to switch it off during any work to be carried out. After the work is completed, he will request back the SSO to switch on that particular feeder.
- Sub-Station Operator (SSO): There is one operator per Sub-Station who monitors the requests made by the SO's present under that Sub-Station. The SSO also has to store the details of the date and time duration for which the feeder was turned off.

Below we briefly explain the 8 steps as per the fig 2.4 which show how a Section Officer and Sub-Station operator communicate with each other in the app to turn off and on a particular feeder which is the main functionality of this application.

- Step 1: The Section Officer requests that a particular feeder under him to be turned off to the Sub-Station operator by mentioning the reason for that feeder to be turned off and also specifies the time period in which he expects it to be turned off. This generates a token from his side.
- Step 2: The message from the officer gets delivered to the operator as a request.
- Step 3: The operator acknowledges and completes the request made by the Section officer by turning off that feeder and this starts the time of the generated token which says when a feeder was turned off by the operator.
- Step 4: The officer gets to know about the request acknowledgement and starts his work.
- Step 5: The Section officer after the completion of the work requests back the operator that feeder to be turned on by specifying a secret code which he enters as a method of verification for the operator.
- Step 6: The operator gets the message about the new request from the officer and has to enter the same secret code given by officer for authentication purpose to complete the request.
- Step 7: The operator enters the secret code and turns on the feeder thereby completing the request of the Section officer. This completes the generated token's time which says when that feeder was turned back on. The time duration and date on which the feeder was turned off will be recorded in the database.
- Step 8: The request of the officer is completed and the feeder is turned on.

2.3 System Requirements and Software Details

Software details are mentioned below.

- Front end:
  - Integrated Development Environment: Android Studio IDE 3.6
  - Android SDK
  - Programming Language: JAVA
- Back end:
  - Python 3.7
  - Framework: Flask
  - Database: MySQL
- Hosting: PythonAnywhere

System Requirements are,

- Smartphone with Android OS 6.0 (Marshmallow) and above.
- Laptop with Android Studio installed in it.

Table 2.1 describes the basic system requirements for Laptop and Android Studio.

Table 2.1: Basic system requirements for Android Studio

	Windows	**Mac**	**Linux**
Operating System Version	Microsoft® Windows® 8/10 (32- or 64-bit) *The Android Emulator only supports 64-bit Windows.*	Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)	GNOME or KDE desktop *Tested on gLinux based on Debian (4.19.67-2rodete2).*
Random Access Memory (RAM)	4 GB RAM minimum; 8 GB RAM recommended.		
Free disk space	2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image).		
Minimum required JDK version	Java Development Kit 8		

Summary:

During data collection we understood how the different system exist in the system and their working. Data plays a major role for the design of the database. We then proposed the main working method of the application by giving the explanation through a block diagram. Then listed out all the details of software's to be used and the necessary system requirements to build and run it.

Chapter 3

SYSTEM DESIGN

System Design is the process of defining the modules, interfaces and data for a system to satisfy specified requirements. We put forth the use case diagram, sequence diagrams, code snippets and database details.

3.1 Use Case Diagram

A use case diagram is a simple representation of a user's interaction with the system that shows the relationship between the user's and the different use cases in which they are involved. The use case diagram of our system is shown below.

Admin gets the highest authority and can control majority of the functions involved. AEE, SO and SSO are the other users with their respective functions as mentioned in methodology.

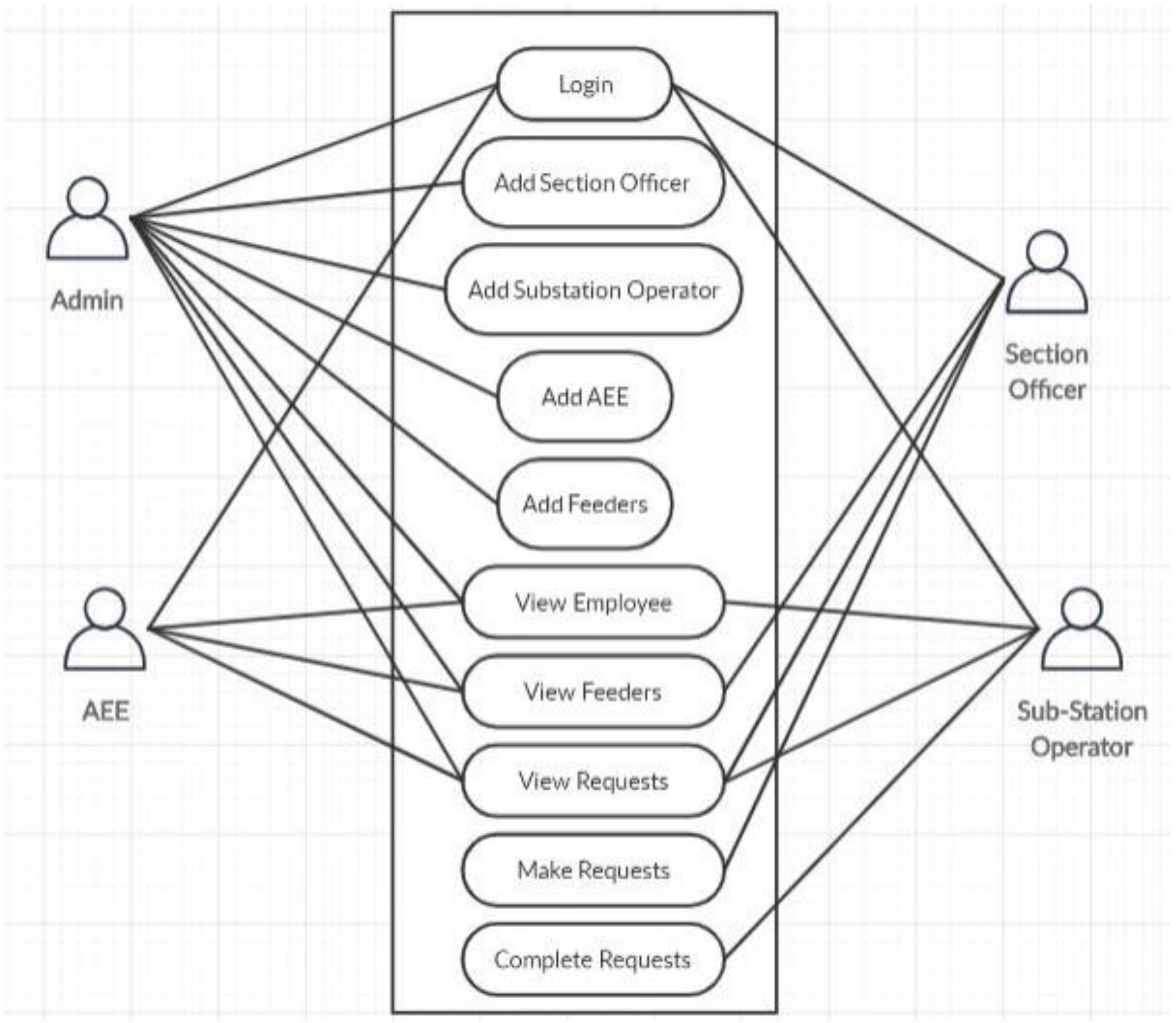


Fig 3.1: Use case diagram

3.2 Sequence Diagrams

A Sequence Diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

3.2.1 Admin Login and Adding Employee

Admin will first login to the system by using his login username and password. Then he will click on employee option where he can add new employees by specifying their details. The details are stored in the database. The new employee added is displayed in the employee list.

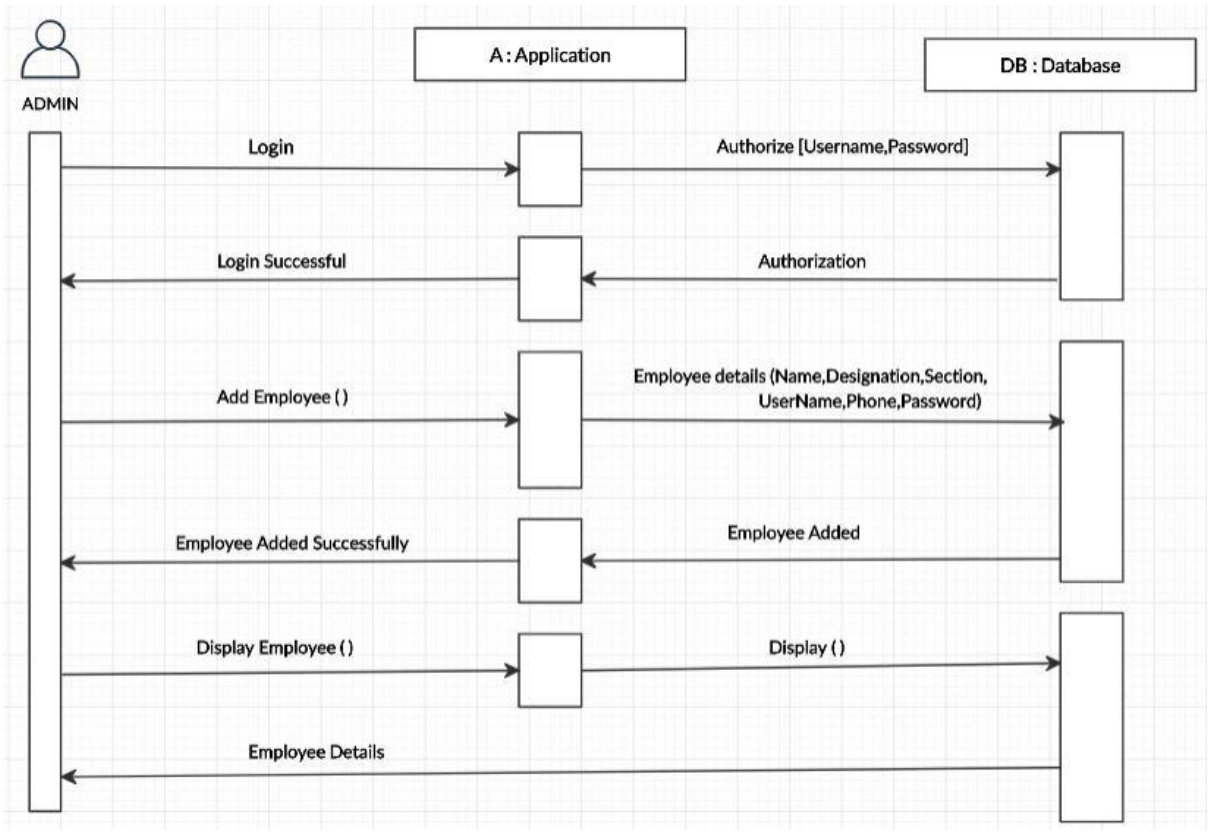


Fig 3.2: Sequence Diagram for Admin Login and Adding Employee

3.2.2 Admin Adding Feeder

Admin will first login to the system by using his login username and password. Then he will click on feeders option where he can add new feeders by specifying their details. He assigns a feeder to the respective section officer. Then the newly added feeder is displayed in the feeders list.

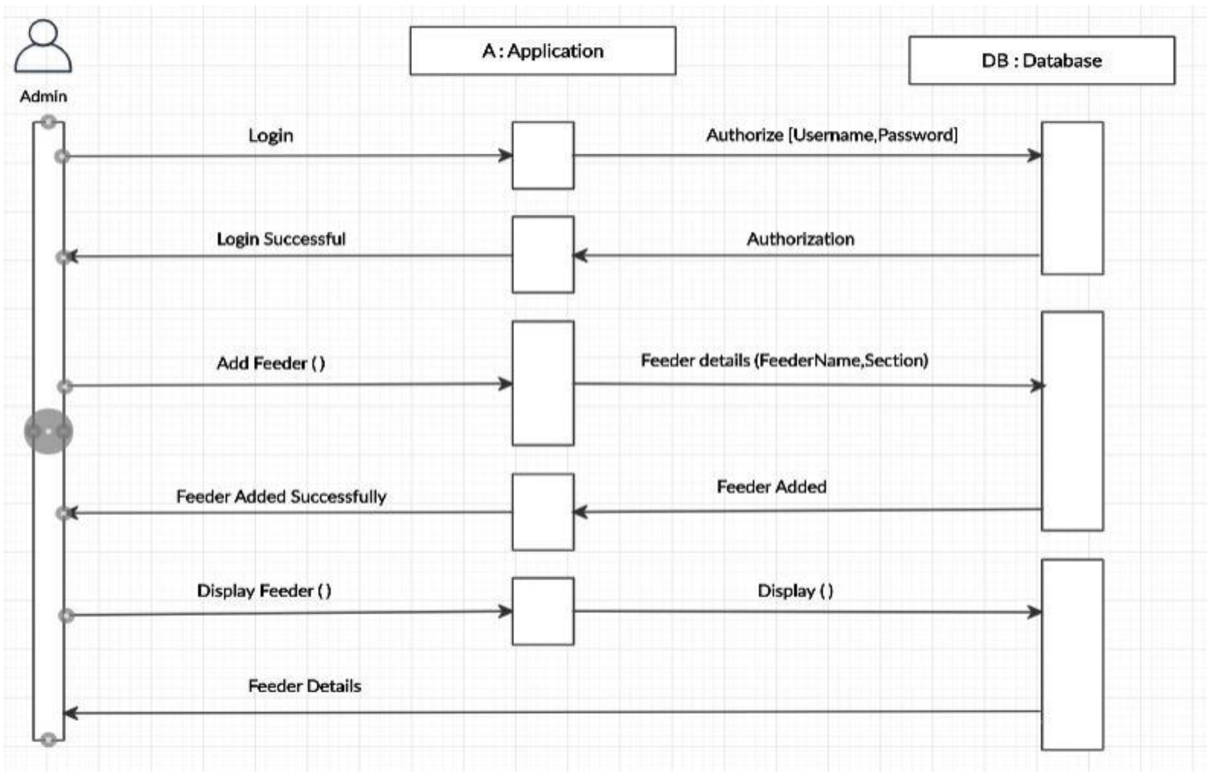


Fig 3.3: Sequence Diagram for Admin Adding Feeder

3.2.3 Section Officer Request to turn off Feeder

Section Officer will request Operator to turn off a feeder by specifying the feeder details such as the reason to turn off and time period. The request is notified to the operator. He turns off that feeder and the details are stored in the database.

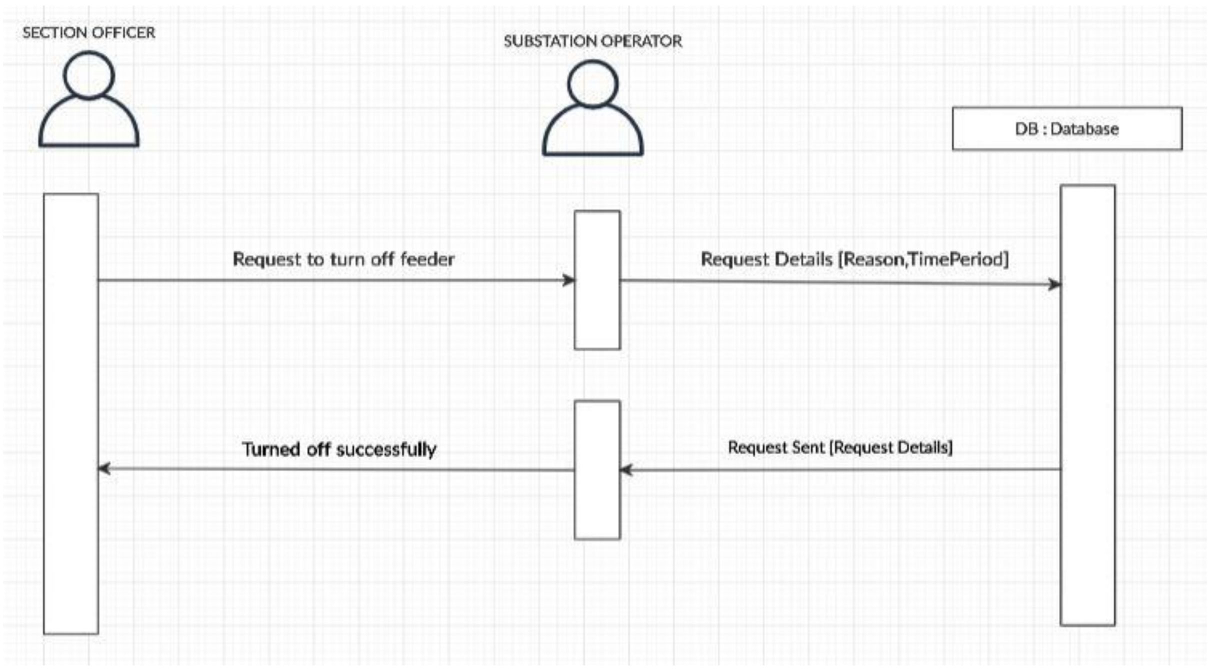

SECTION OFFICER
SUBSTATION OPERATOR
DB : Database
Request to turn off feeder
Request Details [Reason,TimePeriod]
Turned off successfully
Request Sent [Request Details]

Fig 3.4: Sequence Diagram for SO request to turn off Feeder

3.2.4 Section Officer Request to turn on Feeder

Section officer after the work is carried out, requests back the operator to turn on a feeder by specifying a secret key which the operator has to enter on his side for the verification purpose. The operator is notified back the request and he will enter the secret key sent by section officer for authentication and the feeder is turned on back. The details such as the time for which the feeder was turned off, the date and its other details are all stored in the database.

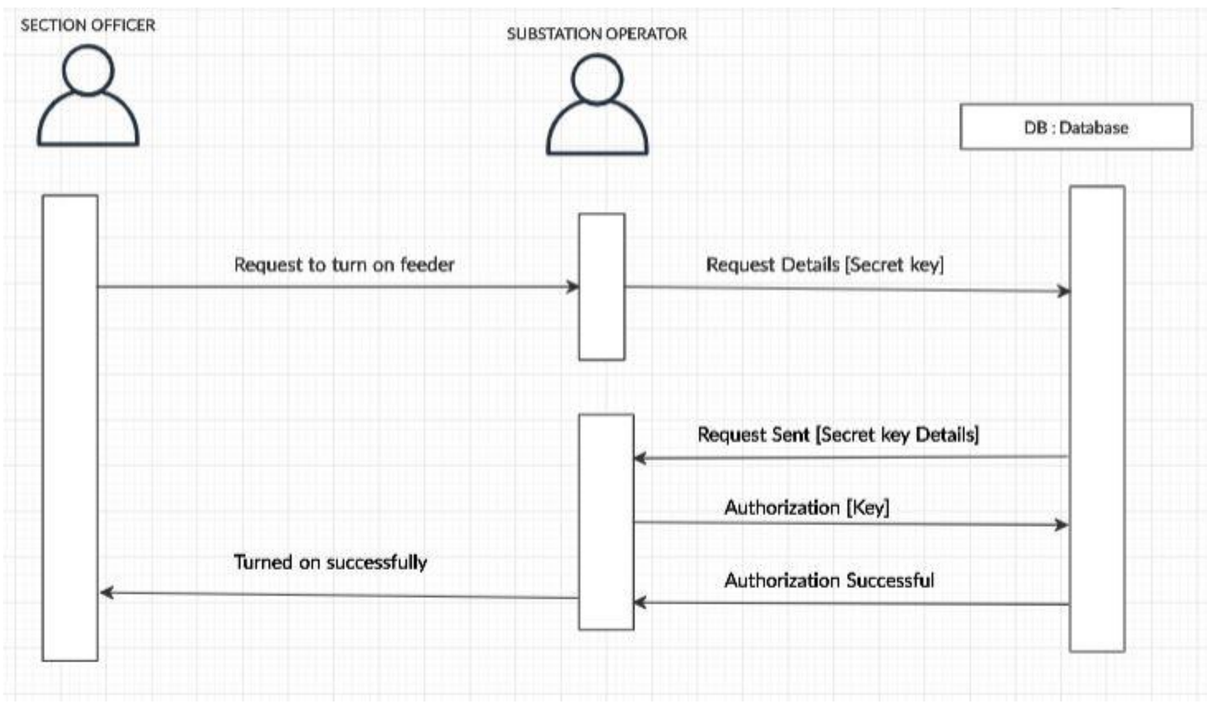


Fig 3.5: Sequence Diagram for SO request to turn on Feeder

3.3 Code Snippets

Code Snippets are the brief examples of the coding done in the project.

3.3.1 Activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:weightSum="10"
    tools:context=".activities.LoginActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/login_layout"
        android:padding="@dimen/app_padding"
        android:elevation="10dp"
        >
        <TextView
            android:id="@+id/tagLineTv"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:fontFamily="@font/heebo_bold_"
            android:gravity="center"
            android:paddingStart="5dp"
            android:paddingEnd="5dp"
            android:text="Welcome"
            android:textAppearance="@style/TextAppearance.AppCompat.Title"
            android:textColor="@color/app_text_color"
            android:transitionName="app_name_tag" />

        <TextView
            android:id="@+id/tagLineTv2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/tagLineTv"
            android:layout_marginEnd="30dp"
            android:fontFamily="@font/heebo_regular"
            android:gravity="start"
            android:paddingStart="5dp"
            android:paddingEnd="5dp"
            android:text="Login into your account"
            android:textAppearance="@style/TextAppearance.AppCompat.Body1"
            android:textColor="@color/app_text_color" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
```

```
android:background="@android:color/transparent"
android:layout_below="@id/tagLineTv2"
android:layout_marginTop="30dp"
>
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:hintTextAppearance="@style/inputTextMedium"

    android:paddingEnd="5dp"
    android:paddingStart="5dp"
    >
    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Username"
        android:id="@+id/username"
        android:textAppearance="@style/inputTextMedium"
        android:inputType="textEmailAddress"
        android:textSize="@dimen/text_medium"

        />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:hintTextAppearance="@style/inputTextMedium"
    android:paddingEnd="5dp"
    android:paddingStart="5dp"
    app:passwordToggleEnabled="true"
    >
    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:id="@+id/password"
        android:textAppearance="@style/inputTextMedium"
        android:textSize="@dimen/text_medium"
        android:inputType="textPassword"
        />
</com.google.android.material.textfield.TextInputLayout>
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="10dp"
    android:textColor="@color/app_text_color"
    android:background="@drawable/btn_background"
    android:text="Log in"
    android:fontFamily="@font/heebo_medium"
    android:textAllCaps="false"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    android:elevation="10dp"
    android:onClick="callHome" />
```

```
        </LinearLayout>
    </RelativeLayout>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/login_layout"
        android:layout_marginTop="50dp"
        android:layout_marginBottom="50dp"
        android:layout_marginLeft="50dp"
        android:layout_marginRight="50dp"
        android:src="@drawable/hescom"
        android:scaleType="fitXY"
        />
</RelativeLayout>
```

3.3.2 MainActivity.java

```
package com.project.powerhouse;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

import androidx.appcompat.app.AppCompatActivity;

import com.project.powerhouse.activities.LoginActivity;
import com.project.powerhouse.utility.AppUtils;

public class MainActivity extends AppCompatActivity {

    private static final long SPLASH_DISPLAY_LENGTH = 2000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        AppUtils.hideToolbar(getSupportActionBar());
        waitAndGo();
    }

    private void waitAndGo() {
        /* New Handler to start the Menu-Activity
         * and close this Splash-Screen after some seconds.*/
        new Handler().postDelayed(new Runnable(){
            @Override
            public void run() {
                /* Create an Intent that will start the Menu-Activity. */
                Intent mainIntent = new Intent(MainActivity.this, LoginActivity.class);
                startActivity(mainIntent);
                finish();
            }
        }, SPLASH_DISPLAY_LENGTH);
    }
}
```

3.4 Database Tables

The Database of the app is mentioned below. There are totally 3 tables used namely Employee, Feeders and TastTickets.

Database Tables:

Table 3.1: Database Tables

```
---------------------------
| Tables_in_Kleit$default |
---------------------------
| Employee                |
| Feeders                 |
| TaskTickets             |
---------------------------
```

Employee Table:

Table 3.2: Employee Table

```
---------------------------------------------------------------------------
| id | empName | section     | designation            | email | phone      |
password |
---------------------------------------------------------------------------
| 15 | so1     | Section 1   | Section officer        | so1   | 9412345684 |
123      |
| 16 | so2     | Section 2   | Section officer        | so2   | 9638527411 |
123      |
| 17 | aee     | Section 1   | AEE                    | aee   | 9876541231 |
123      |
| 18 | sso     | All Section | Sub-station operator | sso   | 9874561233 |
123      |
---------------------------------------------------------------------------
```

Feeders Tables:

Table 3.3: Feeders Table

```
-----------------------------------------------
| id | feederName | section   | switch_status |
-----------------------------------------------
| 16 | X          | Section 1 | ON            |
| 17 | Y          | Section 1 | ON            |
| 18 | Z          | Section 1 | ON            |
| 19 | C5         | Section 1 | ON            |
| 20 | ESI-4      | Section 2 | ON            |
-----------------------------------------------
```

TaskTickets Tables:

Table 3.4: TaskTickets Table

```
--------------------------------------------------------------------------
| id | sectionName | requestBy | handledBy | startTime  | endTime |
requestedStartTime | requestedEndTime | reason | secrateCode | taskCode |
created  | taskStatus | requestedEmpId | requestedFeederId |
--------------------------------------------------------------------------
| 22 | Section 1   | so1       |           | June 21 2020 - 15:15:19 |
June 21 2020 - 15:16:15 | 15:15      | 15:18           | BD Maintenance |
123         | 93610     | 2020-06-21 09:42:33 | COMPLETED |  15 |      16 |
--------------------------------------------------------------------------
| 23 | Section 1   | so1       |           | June 21 2020 - 17:21:44 |
June 21 2020 - 17:22:42 | 17:25      | 17:35           | Jump at HT     |
111         | 27587     | 2020-06-21 11:51:22 | COMPLETED |  15 |      17 |
--------------------------------------------------------------------------
| 24 | Section 2   | so2       |           | June 21 2020 - 17:24:01 |
June 21 2020 - 17:24:37 | 17:30      | 17:40           | TC Failure     |
1           | 66739     | 2020-06-21 11:53:43 | COMPLETED |  16 |      20 |
--------------------------------------------------------------------------
| 25 | Section 2   | so2       |           | June 23 2020 - 12:01:24 |
June 23 2020 - 12:02:00 | 12:5       | 12:10           | Dual blown off |
1           | 61508     | 2020-06-23 06:31:03 | COMPLETED |  16 |      20 |
--------------------------------------------------------------------------
```

Summary:

The first phase during system design was to establish all the use cases for all the users in the application. We then proposed sequence diagrams of how different objects interact in the time period with different functions. The database tables are also specified in this chapter along with some real time data during the testing. Brief code snippets are also mentioned.

Chapter 4

TESTING

The development of software system involves a series of production activities where opportunities for injection of human facilities are enormous. Error may begin to occur at very inception of the process where objectives, may be erroneously or imperfectly specified, as well as in later design and development stages. Because of human inability to perform and communicate in software development is accompanied by quality assurance activity.

Software testing is a critical element of software assurance and represents the ultimate review of specification, design and coding. Testing presents an interesting anomaly for software engineer. The engineer creates a series of tests that are intended to demolish the software that has been built. In fact, testing is one step in software testing process that can be viewed as destructive rather than constructive.

4.1 Testing Objective

Testing is the process of executing a program with the intent of finding an error. A good test case is one that has high probability of finding yet undiscovered error. A successful test is one that uncovers yet undiscovered error.

The above objectives imply a dramatic change in viewpoint. They more counter to the commonly held view that a successful test is one in which no errors are found. Testing cannot show the absence of defect, it can only show that software errors are present.

4.2 Testing Strategies

There are two general strategies for testing software. These are as follows.

- Code Testing: This examines the logic of the program. To follow this test, cases are developed such that every path of program is tested.
- Specification Testing: This examines the specification stating what the program should do and how it should perform under various conditions. Then test cases are developed for each condition and combinations of conditions to be submitted for processing.

4.3 Testing Method Used

Here Black Box testing and Statistical testing are used. In Black box testing, all possible types of inputs are seen for corresponding outputs and if not giving correctly, code is corrected. In Statistical testing, checking for all variables whether they are assigned values before using it, whether array bound is correctly defined, whether looping statement is terminating without going to infinite loop, whether function parameters are passed in order and about numbering parameters etc, are checked successfully and found whether is working correctly or not.

The stages in testing process are:

4.3.1 Unit Testing

The individual components are tested to ensure that they operate correctly. Each component is tested independently without interference of other components.
Ex: Tested for Username and Password during login

4.3.2 Module Testing

Module is a collection of dependent components such as an object classes and an abstract data type are some loose collection of procedures and functions. A module encapsulates related components so it can be tested without other system modules.
Ex: Phone Number should be 10 numerical digits only. This module is tested independently with dummy values for proper working.

4.3.3 Subsystem Testing

This phase involves testing collection of modules, which have been integrated into subsystem. Subsystem may be independently designed and implemented. The most common problems, which arise in the large software systems are subsystems interface mismatches. The subsystem test process should therefore concentrate on the detection of interface errors by rigorously exercising these interfaces.
Ex: Testing various functions present in admin, section officer and substation modules and checking whether updating information in one module gets updated in other modules also and whether the interfaces interact and communicate efficiently.

4.3.4 System Testing

The subsystems are integrated to make up the entire system. The testing process is concerned with finding errors, which result from unanticipated interactions between subsystem and system component. It is also concerned with validating the system's functional and non-functional requirements.

Ex: while adding any data the system must be updating it during the operation to the database and should also be visible in other subsystems.

4.3.5 Acceptance Testing

This is the final stage in testing process before the system is tested for operational use. Acceptance testing may reveal errors and omissions in the system requirements definition because the real data exercises the system in different phase from the test data. Acceptance testing may also reveal the requirements problem where the system facilities do not meet the user's needs or system performance is unacceptable.

Ex: Testing is done for all objectives which were stated in the project statement whether they meet the requirements or not.

4.4 Test Cases

Table 4.1: login Testing

Module	**Test Case**	**Test Data**	**Description**	**Expected Output**	**Obtained Output**	**Pass / Fail**
Admin Login	Username and password validation	Enter Valid username and password	Verifies with database for correct credentials	Login Successful	Login Successful	Pass
Admin Login	Username and password validation	Enter Invalid username and password	Verifies with database for correct credentials	Username and password mismatch	Username and password mismatch	Pass
AEE Login	Username and password validation	Enter Valid username and password	Verifies with database for correct credentials	Login Successful	Login Successful	Pass

AEE Login	Username and password validation	Enter Invalid username and password	Verifies with database for correct credentials	Username and password mismatch	Username and password mismatch	Pass
SO Login	Username and password validation	Enter Valid username and password	Verifies with database for correct credentials	Login Successful	Login Successful	Pass
SO Login	Username and password validation	Enter Invalid username and password	Verifies with database for correct credentials	Username and password mismatch	Username and password mismatch	Pass
SSO Login	Username and password validation	Enter Valid username and password	Verifies with database for correct credentials	Login Successful	Login Successful	Pass
SSO Login	Username and password validation	Enter Invalid username and password	Verifies with database for correct credentials	Username and password mismatch	Username and password mismatch	Pass

Table 4.2: Adding Employee

Module	**Test Case**	**Test Data**	**Description**	**Expected Output**	**Obtained Output**	**Pass / Fail**
Adding Employee	Adding Employee with same Username Validation	Employee Username	Admin adds new employee with existing username	Error. Should not be added	Integrity error. Username already exists	Pass

Table 4.3: Phone Number Validation

Module	**Test Case**	**Test Data**	**Description**	**Expected Output**	**Obtained Output**	**Pass / Fail**
Adding Employee	10 Digits Phone Number Validation	Employee Phone Number 10 digits	Should accept only if its 10 digits	Added successfully	Added successfully	Pass
Adding Employee	10 Digits Phone Number Validation	Employee Phone Number not with 10 digits	Should not accept if its other than 10 digits	Phone no must be 10 digits	Phone no must be 10 digits	Pass

Table 4.4: Secret Key Validation

Module	**Test Case**	**Test Data**	**Description**	**Expected Output**	**Obtained Output**	**Pass / Fail**
Substation operator	Secret Key validation	Same Secret key entered by section officer	Should turn on feeder only if key is same	Password Verified	Password Verified	Pass
Substation operator	Secret Key validation	Different Secret key	Should not turn on the feeder if key is different	Please Verify entered Password	Please Verify entered Password	Pass

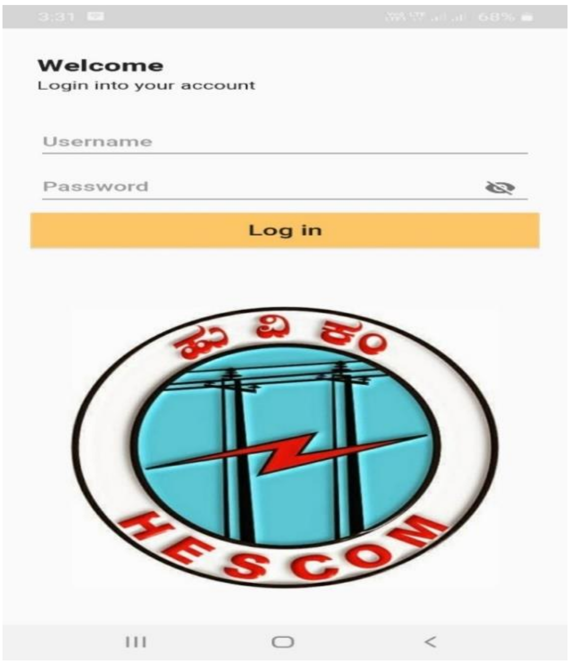
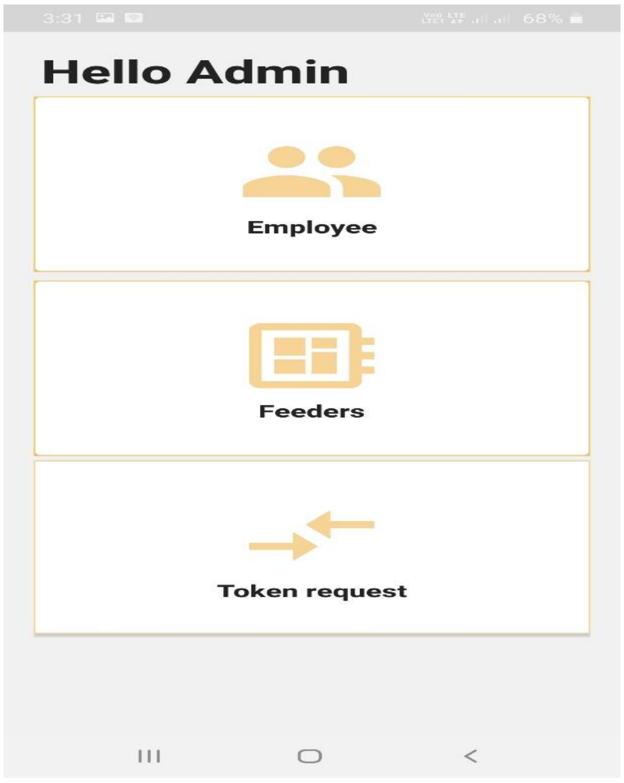
Summary:

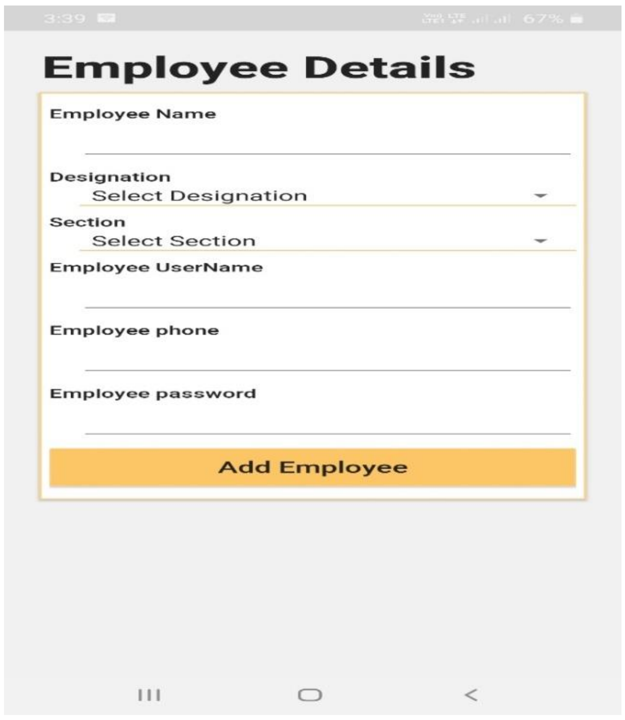
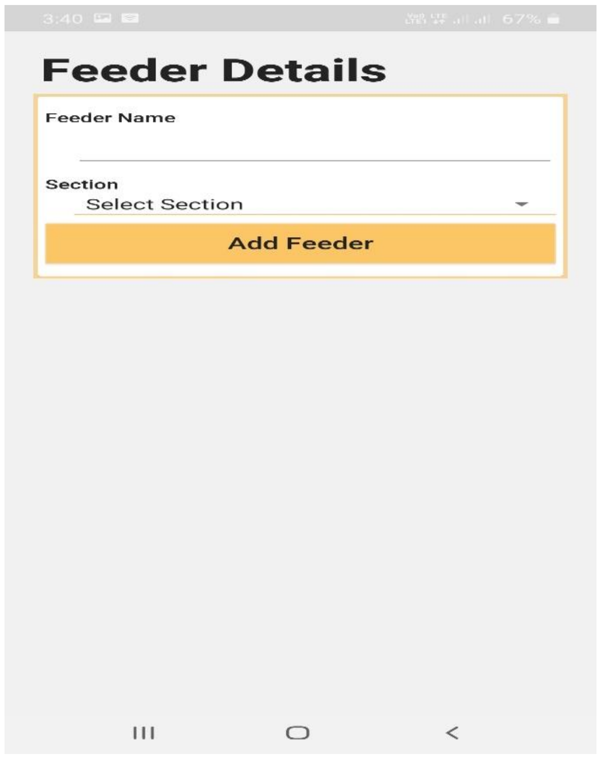
Testing is one of the important phases during development to understand how the system behaves while used differently. Code testing and specification testing are the two test strategies carried out. We carried out different types of testing methods as mentioned in the chapter and found out few errors which needed correction such as phone no validation, request toast message necessity, etc. We made these changes along the system to function efficiently. Different test cases carried out are specified in the tables.

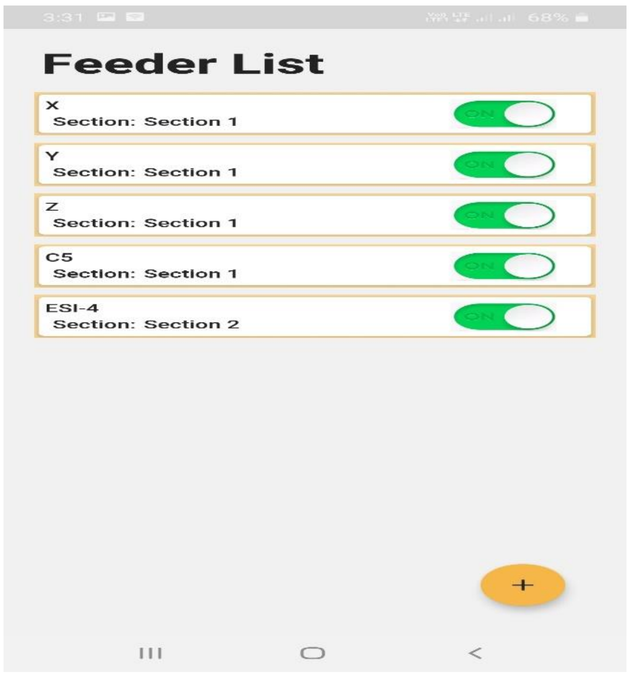
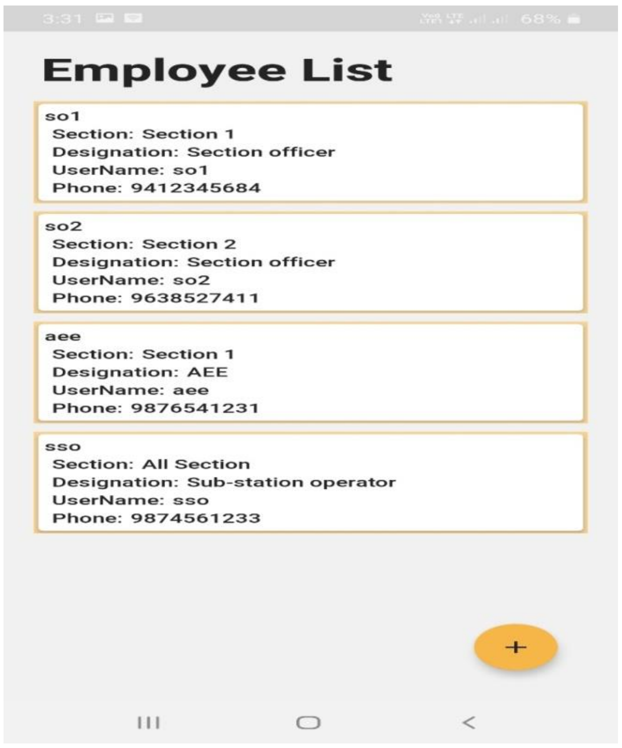
Chapter 5

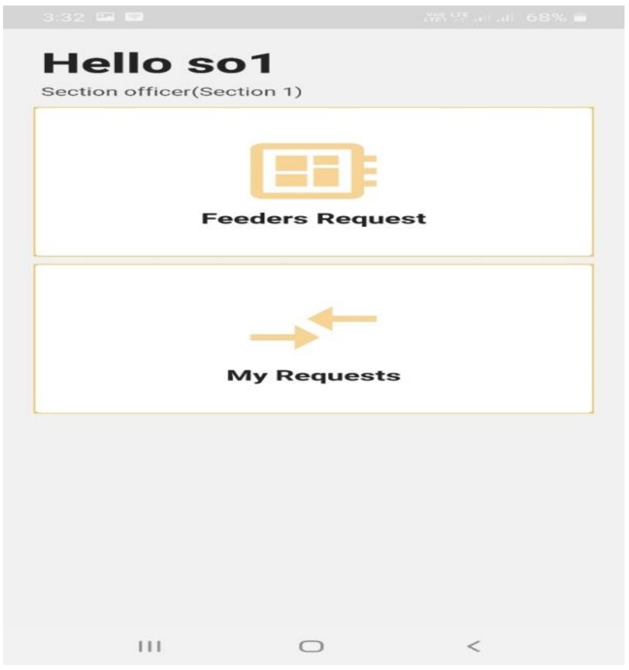
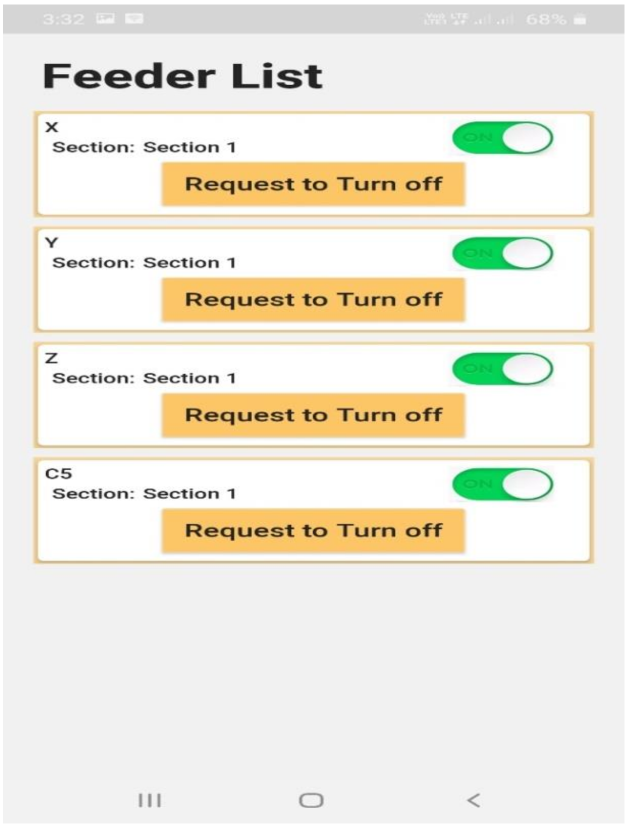
SNAPSHOTS

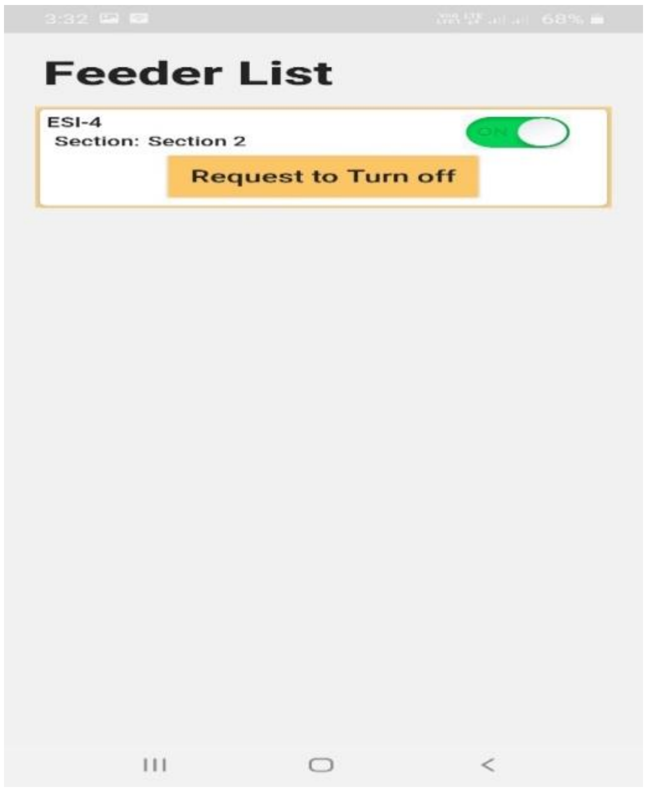
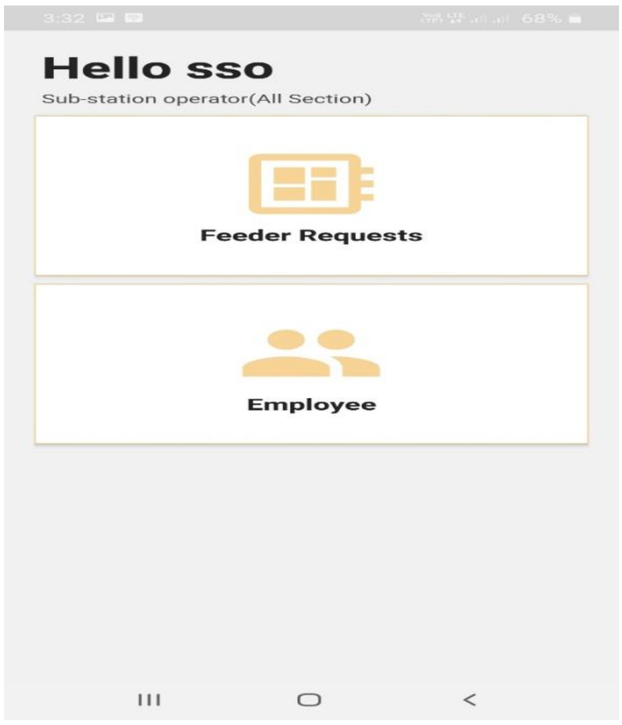
Screenshots of the user interface of the application are given below.

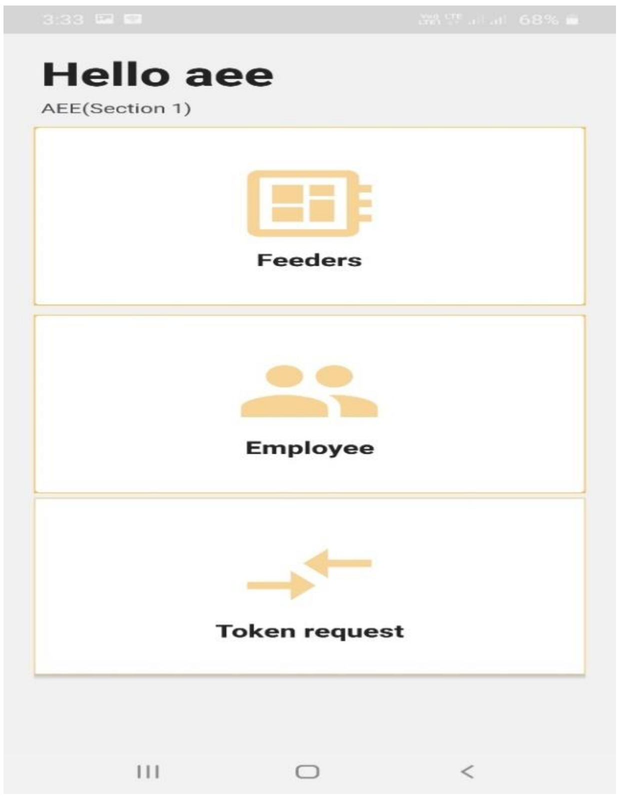
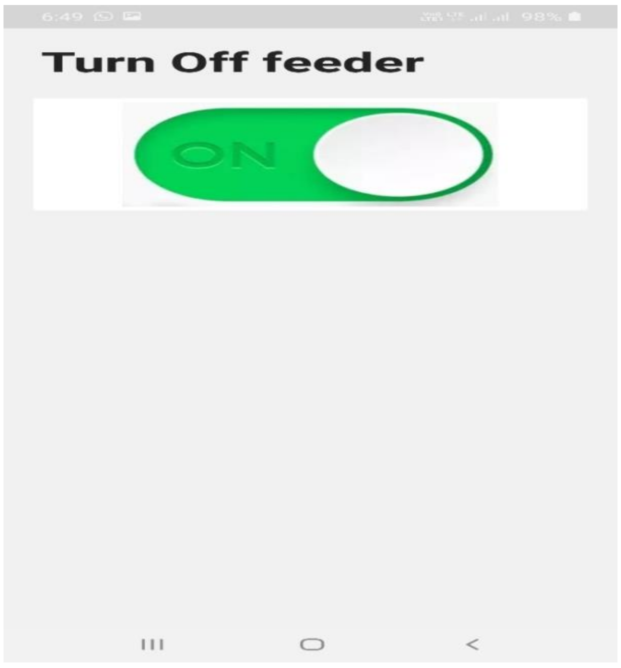
 3:31 68% Welcome Login into your account Username Password Log in Fig 5.1: Login Screen	➢ This screen is the first screen of the application. ➢ All the users' login through their login credentials.
 3:31 68% Hello Admin Employee Feeders Token request Fig 5.2: Admin Screen	➢ This is the admin's home screen. ➢ This shows all the info regarding employees, feeders and token requests.

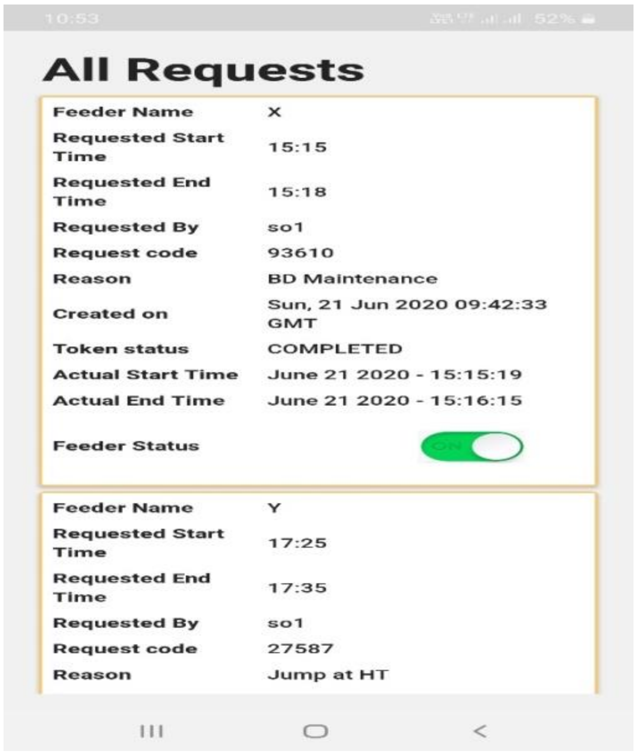
 3:39 Employee Details Employee Name Designation Select Designation Section Select Section Employee UserName Employee phone Employee password Add Employee Fig 5.3: Add Employee	➢ This is the Add Employee screen for the admin. ➢ Admin will add all the employees here by specifying their details.
 3:40 Feeder Details Feeder Name Section Select Section Add Feeder Fig 5.4: Add Feeder	➢ This is the Add Feeder screen for the admin. ➢ Admin will add all feeders here by specifying their details. ➢ He will also assign the feeder to respective section officer.

 **Feeder List** X Section: Section 1 ON; Y Section: Section 1 ON; Z Section: Section 1 ON; C5 Section: Section 1 ON; ESI-4 Section: Section 2 ON; + Fig 5.5: Feeders list	➢ This screen is Feeders list for the admin. ➢ All the feeders added will be displayed here to him.
 **Employee List** so1 Section: Section 1 Designation: Section officer UserName: so1 Phone: 9412345684; so2 Section: Section 2 Designation: Section officer UserName: so2 Phone: 9638527411; aee Section: Section 1 Designation: AEE UserName: aee Phone: 9876541231; sso Section: All Section Designation: Sub-station operator UserName: sso Phone: 9874561233; + Fig 5.6: Employee’s list	➢ This screen is Employees list for the admin. ➢ All the employees added belonging to different will be displayed here to him.

 Hello so1 Section officer(Section 1) Feeders Request My Requests Fig 5.7: SO1 Screen	➢ This is the Section Officer 1's home screen. ➢ He can make new requests and view his previous requests.
 Feeder List X Section: Section 1 ON Request to Turn off Y Section: Section 1 ON Request to Turn off Z Section: Section 1 ON Request to Turn off C5 Section: Section 1 ON Request to Turn off Fig 5.8: SO1 Feeder Screen	➢ This is SO1's feeder request screen. ➢ He can request to turn on/off any feeder under his control.

 3:32 Feeder List ESI-4 Section: Section 2 ON Request to Turn off Fig 5.9: SO2 Feeder Screen	➢ This is SO2's feeder request screen. ➢ He can request to turn/off any feeder under his control.
 3:32 Hello sso Sub-station operator(All Section) Feeder Requests Employee Fig 5.10: Operator Screen	➢ This is Sub-Station Operator home screen. ➢ He can view and respond to all the requests made by the SO's. ➢ He can also view employee details.

 3:33 68% Hello aee AEE(Section 1) Feeders Employee Token request Fig 5.11: AEE Screen	➢ This Assistant Executive Engineer’s home screen. ➢ He can view all the employees and feeders present in the app. ➢ He can also view all the requests made by the So’s.
 Fig 5.12: Operator Feeder turn off screen	➢ This is the toggle switch for operator. ➢ He can turn off the feeder by turning off the switch.



10:53 52%

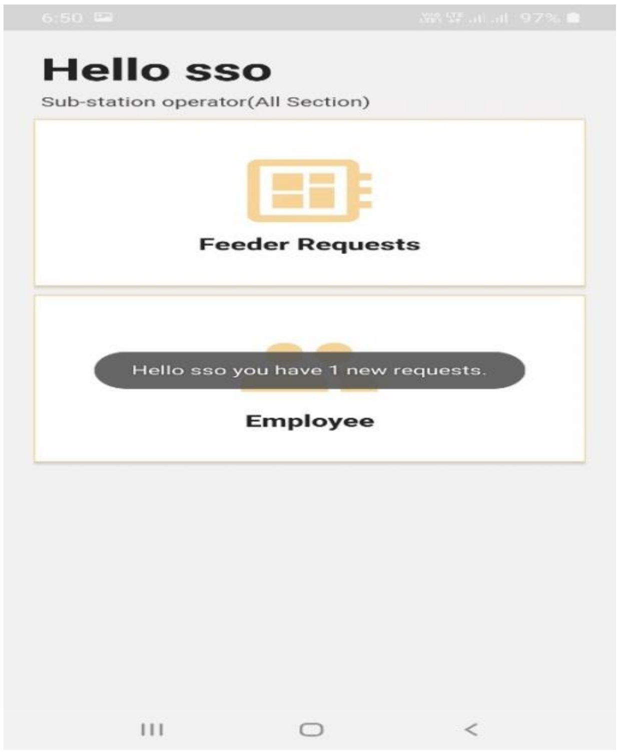
All Requests

Feeder Name	X
Requested Start Time	15:15
Requested End Time	15:18
Requested By	so1
Request code	93610
Reason	BD Maintenance
Created on	Sun, 21 Jun 2020 09:42:33 GMT
Token status	COMPLETED
Actual Start Time	June 21 2020 - 15:15:19
Actual End Time	June 21 2020 - 15:16:15
Feeder Status	ON

Feeder Name	Y
Requested Start Time	17:25
Requested End Time	17:35
Requested By	so1
Request code	27587
Reason	Jump at HT

Fig 5.13: SO request list

- This is the view requests screen for the section officer.
- He can view and keep track of all the requests made by him to the operator.



6:50 97%

Hello sso

Sub-station operator(All Section)

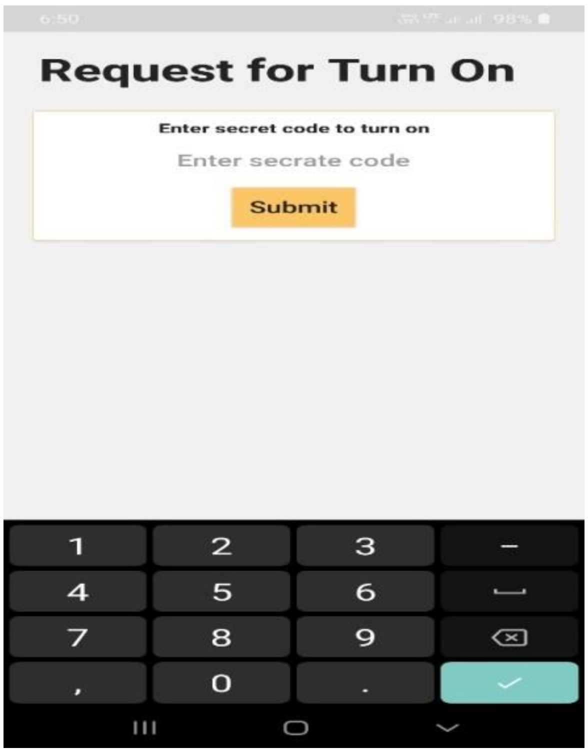
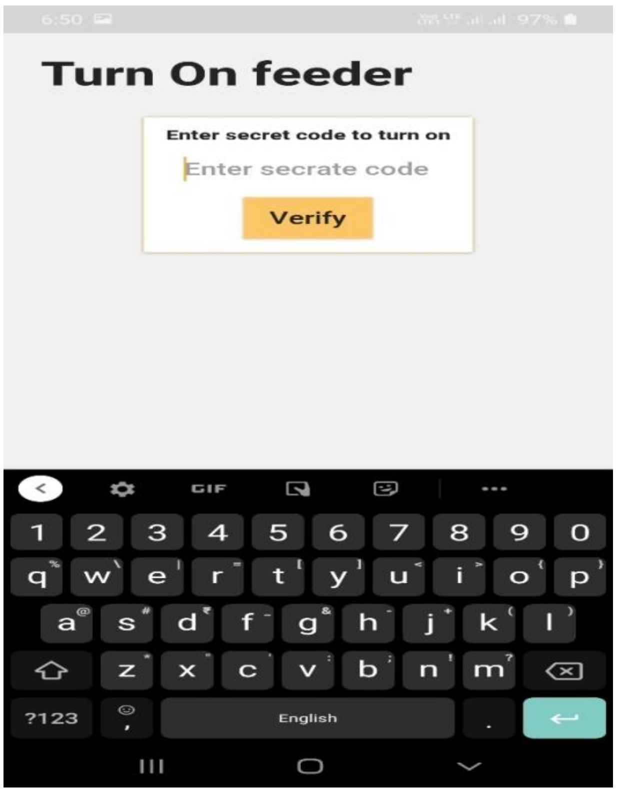
Feeder Requests

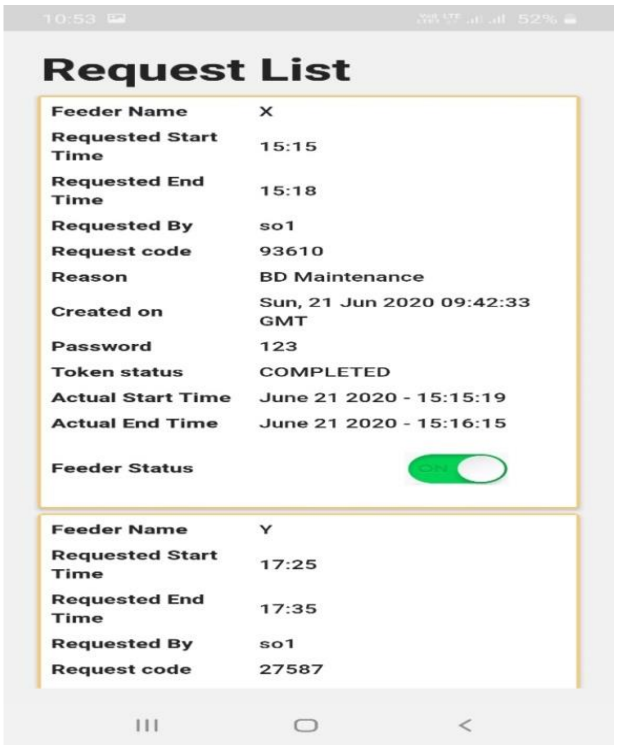
Hello sso you have 1 new requests.

Employee

Fig 5.14: SSO request Toast message

- This is the toast message for the sub station operator whenever SO makes a request to him.
- This message will be delivered to him whenever he logs in to the app and there are new requests made.

 Request for Turn On Enter secret code to turn on Enter secrate code Submit Fig 5.15: SO Secret key Screen	➢ Whenever the SO wants to make a request to turn on the feeder back, he has to provide a secret key for the SSO to use. ➢ This ensures double verification for the SSO side.
 Turn On feeder Enter secret code to turn on Enter secrate code Verify Fig 5.16: SSO secret key screen	➢ The SSO has to enter the same secret key used by the SO. ➢ Only then the feeder will be turned back on.



Request List

Feeder Name	X
Requested Start Time	15:15
Requested End Time	15:18
Requested By	so1
Request code	93610
Reason	BD Maintenance
Created on	Sun, 21 Jun 2020 09:42:33 GMT
Password	123
Token status	COMPLETED
Actual Start Time	June 21 2020 - 15:15:19
Actual End Time	June 21 2020 - 15:16:15
Feeder Status	ON

Feeder Name	Y
Requested Start Time	17:25
Requested End Time	17:35
Requested By	so1
Request code	27587

Fig 5.17: SSO request list with key

- This is the request screen for the SSO.
- There is also secret key detail mentioned as password which he enters in the following screen to turn back the feeder on.

Summary:

This chapter includes all the snapshots i.e. screenshots of the user interface of the application. All these forms the UI of the app which the users will navigate through the app. The brief explanation of each screenshot is also given beside to understand their functionality and working.

Chapter 6

CONCLUSION AND FUTURE SCOPE

The conclusion of this project includes the summary in a brief manner. How we finally conclude this project, our thoughts regarding the work carried out so far and etc. Future scope means the better possibilities of the project. Its further expansion which may be done in the near future. Additional functionalities or new features which can be added. The project can also be further expanded in terms of no of users.

6.1 Conclusion

The project discussed in this report is "Fool Proof Line Clearing System". The objective of the project was to implement the required functionalities in an android application. The concerned objectives mentioned earlier are met in this project. The app is developed in a very simple approach and minimal functionality which is required by the end users. The entire app is made solely to improve the communication between the two users involved in the line clearing system of the current distribution. All the users and the people associated with this system understand the advantage of it. The system solves the problem mentioned foremost.

Our experience to develop this project into a product has been very fruitful. We learnt many things associated with how to develop a useful product which is used by the users. We understood how development of a product takes place. We gained good knowledge and practical experience by working on this project. We got to meet different people associated with different domains, understood their problems and requirements, collected the real time data needed. Our guide also helped us in understanding the core aspects of the project correctly. Overall, it was a very good experience for us working on this project.

6.2 Future Scope

- ➢ The app at present is uploaded in free server which limits its time of use, it can be uploaded in fully functional server which has no time limit and by doing so the server response time increases drastically ensuring smooth operations in the app and the communication.
- ➢ The app can be included with other needed or necessary functions as per the requirements of the users to improve its functionality.

- One can scale the project to many users and the systems concerned with it to function in better way and efficient manner.
- It can be implemented in other branches of the current distribution system where communication plays a crucial role for the system as well as for the safety of all the personnel involved in the work.
- By the study of this application, other systems can be automated which still function in a manual way of working and communication.
- This project can be integrated with the hardware related components as a project related to Internet of things which is more beneficial for the system and its working.

Summary:

This chapter itself is like the summary of the project carried out. We put out the conclusion by saying all the work carried out in accordance with the requirements specified. Our experience and leaning from this project so far. Future scope is where we specify all the possible features or improvements that can be done further to this application.

BIBLIOGRAPHY

1. Organization Reference:

   HESCOM (Hubli Electric Supply Company Ltd): DSM (Demand Side Management), R&D Cell, Navanagar Hubli.
   - Executive Engineer: Mr. Rangaraju
   - Assistant Executive Engineer: Mr. Raghavendra Vishwakarma
   - CSD3 Substation Gokul Road Hubballi.
2. BESCOM (Bangalore Electric Supply Company Ltd):

   https://bescom.karnataka.gov.in/english

   https://bescom.karnataka.gov.in/storage/pdf-files/IE-rules.pdf

   https://bescom.karnataka.gov.in/storage/pdf-files/safety-manual.pdf
3. Introduction to Android: http://developer.android.com/guide/index.html
4. Android operating system: https://en.wikipedia.org/wiki/Android_(operating_system)
5. Android Studio: https://en.wikipedia.org/wiki/Android_Studio
6. Android API: http://developer.android.com/guide/topics/ui/index.html
7. Android UI: http://developer.android.com/guide/topics/ui/index.html
8. Android Architecture: https://developer.android.com/guide/platform
9. Python: https://en.wikipedia.org/wiki/Python_(programming_language)
10. PyhtonAnywhere: https://www.pythonanywhere.com/

    https://en.wikipedia.org/wiki/PythonAnywhere
11. Flask: https://en.wikipedia.org/wiki/Flask_(web_framework)
12. Electrical Substation: https://en.wikipedia.org/wiki/Electrical_substation
13. Current Distribution System: https://en.wikipedia.org/wiki/Electric_power_distribution