CHAPTER 1

**INTRODUCTION**

One of the main contributions of the 21st century’s Computer. Computers have become a way of life for today’s high-tech society. Many concepts of modern society that we have come to accept as common place would not be possible if there are no computers. Today computers have used extensively in many areas of business, industry, banking, education and other day-to-day activities. It optimizes feature stock and it has personality called and impersonal, sinister and Scheming.

The web application entitled ‘**POWER PLANT MANAGEMENT**’, is a package that deals with power plant activities on the web. This web application is developed in programming platform of ASP.Net , Android and MySQL for managing databases. It provides an new vision on the field of industry development and technology, we introduce this user-friendly software for every ends users, can view details and make transactions online etc.

The program is an online service management administrator program which is supported with IOT devices. Here consumers are first able to sign up to the site after the registration is done at the plantation and the sign is done using password that is send via SMS, then they have access to their profile as well as other functionalities. The program allows setting alerts to the limit to the use of current and make complaints to respective staffs that have duty in their locality and allows them to pay online.

The advantages of the proposed system are that the online system simply automates Power plant system. It is faster and reliable, provides better services as compared to manual system. This system may increase the profitable for both consumers and power plant . The user friendly and interactive interface makes using the application easy for everyone. Since this concept is new, there is no much advancement needed, until the software is expanded to IOS platform.

The project title “**POWER PLANT MANAGEMENT**” is interactive and highly user friendly, making the menu driven system help the end user to work on it without any operational difficulty. The system covers almost all the functional areas of an online farmers portal.

CHAPTER 2

**SYSTEM STUDY**

Analysis is a detailed study of various operations performed by a system and data are collected on the available files, decision points and transaction handled by present system. In the system analysis phase, first we had to analyze the system thoroughly and understand the process and come to an outline of new system. This is the proposed system and again we analyze the feasibility of proposed system that is selecting the best combination of hardware cost and the cooperating of staff of the organization is also considered. The various techniques used in this are:

* Observation

The observation of the functioning of the system that exists gives the idea for the design of the new system. It is helpful to understand and study entire current system.

* Interviews

The main objective is to obtain information regarding system from the concerned exclusive to understand the system requirements and they are to improve the existing system.

* Discussion

Discussion goes a long way in satisfying the needs of the management the objective of the discussions transferring the ideas between department and the system developer. Through discussion useful suggestions made by the user can be in corporate into the new system.

**2.1 EXISTING SYSTEM**

A such there is no existing system for the application. Normally consumers are forced to make phone call to complaint issues and are not well documented. Also, they are not able to set their limits usage of current and make frequent payment in small amount. This existing system forces the staff to make physical disconnection if proper payments are not made and also fraud activities can take place at consumers site.

**2.2 PROPOSED SYSTEM**

Proposed system is an interactive, highly user friendly and designed exclusively for both consumers and staff. It serves the need of agent who play between farmers and dealer and takes away their profit. It is a multi-user, multi-tasking system which is effective, flexible and robust. It operates via Internet where you can link many computer devices at the same time. It allow faster service and is easy to use with a simple Graphical User Interface This system is developed in ASP.Net, Android with MYSQL SERVER as backend.

In this system the keeping of records in files is considerably large since everything is automated. We can control the devices from the plantation itself and keeps refreshing the database with load send it periodically and work assignment can be done online and can be viewed by respective staffs . So here we need not maintain a paper record keeping. Here the time taken is reduced; generating the amount and paying salary can be done very easily. If there happens to be mistakes in data entry and the error is found out, updating of the other records can be done easily in the automated system.

**Advantages or benefits of the proposed system**

* Searching process is faster, easier and efficient in this system.
* Record keeping and data accessing is a simpler task.
* If an enquiry is made, then reports can be submitted quickly without any complications.
* Easy access to information.
* Healthy Interaction between users.
* Reduces workload.
* Less time consuming.
* Provide a user-friendly environment

**MODULAR DESCRIPTION**

Basically, the system is divided into several modules that play an important roles in the program.

1. **Complaint management**

This enables both consumer and staff register their complaints to the admin, and they can receive appropriate reply regarding their complaint. It is useful in building healthy relationship.

1. **Registration management**

This module deals with registration of both staff and consumer. Where valid data are entered and respective user name as well as password is also provided.

1. **Billing management**

Here consumer is given access to online payment. Where proper bank details are provided, and transaction is completed.

1. **Tender management**

This module deals with tender published by power plant and inviting consumer to be part of their company.

1. **Alert management**

This module enables users to set their limit of consumption of current and have a periodic checking. When an alert is set the program informs the user if they have crossed the limit.

1. **Leave management**

This module is for staff only because it is used to request leave by the staff and the admin can either approve or reject it.

1. **Price change management**
2. **Assign location management**

This module deals with work assigning to staff. The staff are assigned with location on the bases post pin code.

1. **Salary management**

This module deals with paying salary to staff, And the payed salary can be viewed.

CHAPTER 3

**SYSTEM ANALYSIS**

A system is an orderly grouping of interdependent components linked together according to a plan to achieve a specific objective. That is a system is a combination of resources working together to convert input into useful outputs. The resources used by a system include personal facilities and lateral equipment’s.

Analysis is the detailed study of various operations performs by the system and their relationship, available files, decision points and transactions handled by the present system. Training, experience and commonsense are required for collection of information needed to do the analysis.

System analysis is a general term that refers to an orderly structured process for identifying and solving problems. We call the system analysis process, the life cycle methodology, since it relates to our significant phases in the life cycle of all business information system. The definition of system analysis include not only the process of analysis, but also that of synthesis, which is the process of put in parts together to form a new system.

**3.1 Feasibility Study**

The feasibility study is carried out to test whether the proposed system is worth being implemented. The proposed system will be selected if it is best enough in meeting the performance requirements. This entails identification, description and evaluation of candidate system for the job.

There are three aspects in the feasibility study portion of the preliminary investigation.

**3.1.1 Technical Feasibility**

The assessment of technical feasibility must be based on the outline of the system requirements in terms of inputs, outputs, files, programs, procedures and staff. This can be quantified in terms of volumes of data, trends, frequency of updating, etc. Having identified an outline system, the investigator must go on to suggest the type of equipment required, methods of developing the system and methods of running the system.

With regard to the processing facilities, the feasibility study will need to consider the possibility of using a bureau or, if in-house equipment is available, the nature of the hardware to be used for data collection, storage, output and processing. There are number of technical issues, which are generally raised during the feasibility stage of the investigation. They are as follows

* Does a necessary technology exist to do what is suggested?
* Does the proposed equipment have the capacity to hold the data required to use the new system?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and security?

**3.1.2 ECONOMIC FEASIBILITY**

Economic analysis is the most frequently used method for evaluating the effectiveness of the candidate system. More commonly known as cost/benefit analysis, the procedure is to determine the benefit and saving that are expected form a candidate system and compare them with cost. If benefits outweigh cost then the decision is made to design and implement the cost. Otherwise further justification or alteration in the proposed system will have to be made if it is to have chance of being approved. This proposed system is profitable due to avoiding stationary things like paper, pens etc.

**3.1.3 OPERATIONAL FEASIBILITY**

The hierarchy of the system is very easier than the existing system. The new system is much easier and user friendly. Operational cost is bearable. The Operation with the new system is very easy. The maintenance and modification of this new system needs very less human efforts. Using command buttons throughout the application programmers enhances the operational feasibility. The Operational feasibility enhances the conversion of manual work to the computerized form of the system.

Proposed projects are beneficial only if they can be turned into information systems that will meet the operating requirements of the organization. The test of feasibility asks if the system will work when it is developed and installed. Some of the important questions that are useful to test the operational feasibility of a project are given below

* Is there sufficient support for the project from the management? From users? If the present system is well liked and used to the extent that people would not be able to see reasons for a change, there may be a resistance
* Are current methods acceptable to the users? If they are not, users may welcome a change that will bring about a more operational and useful system.
* Have the users been involved in the planning and development of the project, and then the changes of resistance can be possibly reduced.
* Issues that appear to be quite minor at the early stage can grow into major problems after implementation.

CHAPTER 4

**USER CHARACERTSTICS**

The user characteristics involves three sections in the Agriculture Assist Portal

1. Administrator

2. Staff

3. Consumer

**Administrator:** This module provides the administrator to control the whole website. He can view the database and control the flow of the project. He is the super-user who can delete any user. The administrator can approve rent request, farmers and dealer’s signup, view bank details, select district of function, view feedback, etc.

**Staff:** This module deals with all the necessary activities that a farmer can do using this website. He can make deals as he wishes, can rent-in and rent-out farming machines, view cart and can make online payments.

**Consumer:** This module deals with all the necessary activities that a dealer can do using this website. He can fix deals and can contact the farmer directly, can rent-out farming machines, view cart and can make online payments.

CHAPTER 5

SYSTEM SPECIFICATIONS

5.1. Hardware Specification for Development and Implementation

The selection of hardware is very important in the existence and proper working of any software. When selecting hardware, the size and capacity requirements are also important. Below are some of the hardware that is required by the system.

|  |  |
| --- | --- |
| Main Processor | Intel core i5 |
| RAM | 8 GB or Above |
| Keyboard | Standard 108 keys |
| Mouse | 3D Optical mouse |
| Monitor | 15” Standard |
| Hard disk | 500GB or Above |

5.2. Software Specification

We require much different software to make the application which is in making to work efficiently. It is very important to select the appropriate software so that the software works properly.

|  |  |
| --- | --- |
| Operating System | Microsoft Windows 10 |
| Programming Language | ASP.Net , Android |
| RDBMS | MYSQL |
| Web Server | Apache Tomcat |
| Scripting language | JSP |

5.3. ABOUT THE SOFTWARE TOOLS AND PLATFORM

It is very important to select appropriate software so that the software works properly.

ABOUT ASP.NET

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.

ASP.NET works on top of the HTTP protocol and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

The ASP.NET application codes can be written in any of the following languages:

* C#
* Visual Basic.Net
* Jscript
* J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

ASP.NET Web Forms Model

ASP.NET web forms extend the event-driven model of interaction to the web applications. The browser submits a web form to the web server and the server returns a full markup page or HTML page in response.

All client-side user activities are forwarded to the server for stateful processing. The server processes the output of the client actions and triggers the reactions.

Now, HTTP is a stateless protocol. ASP.NET framework helps in storing the information regarding the state of the application, which consists of:

* Page state
* Session state

The page state is the state of the client, i.e., the content of various input fields in the web form. The session state is the collective information obtained from various pages the user visited and worked with, i.e., the overall session state. To clear the concept, let us take an example of a shopping cart.

User adds items to a shopping cart. Items are selected from a page, say the items page, and the total collected items and price are shown on a different page, say the cart page. Only HTTP cannot keep track of all the information coming from various pages. ASP.NET session state and server-side infrastructure keep track of the information collected globally over a session.

The ASP.NET runtime carries the page state to and from the server across page requests while generating ASP.NET runtime codes and incorporates the state of the server-side components in hidden fields.

This way, the server becomes aware of the overall application state and operates in a two-tiered connected way.

The ASP.NET Component Model

The ASP.NET component model provides various building blocks of ASP.NET pages. Basically, it is an object model, which describes:

Server-side counterparts of almost all HTML elements or tags, such as <form> and <input>.

Server controls, which help in developing complex user-interface. For example, the Calendar control or the Grid view control.

ASP.NET is a technology, which works on the .Net framework that contains all web-related functionalities. The .Net framework is made of an object-oriented hierarchy. An ASP.NET web application is made of pages. When a user requests an ASP.NET page, the IIS delegates the processing of the page to the ASP.NET runtime system.

The ASP.NET runtime transforms the .aspx page into an instance of a class, which inherits from the base class page of the .Net framework. Therefore, each ASP.NET page is an object and all its components i.e., the server-side controls are also objects.

ABOUT ANDROID

Android is a Linux based operating system it is designed primarily for touch screen mobile devices such as smart phones and tablet computers. The operating system have developed a lot in last 15 years starting from black and white phones to recent smart phones or mini computers. One of the most widely used mobile OS these days is android. The android is software that was founded in Palo Alto of California in 2003.

The android is a powerful operating system and it supports large number of applications in Smartphones. These applications are more comfortable and advanced for the users. The hardware that supports android software is based on ARM architecture platform. The android is an open source operating system means that it’s free and any one can use it. The android has got millions of apps available that can help you managing your life one or other way and it is available low cost in market at that reason’s android is very popular.

The android development supports with the full java programming language. Even other packages that are API and JSE are not supported. The first version 1.0 of android development kit (SDK) was released in 2008 and latest updated version is jelly bean.

Android Architecture:

The android is a operating system and is a stack of software components which is divided into five sections and four main layers that is

* Linux kernel
* Libraries
* Android runtime

ABOUT MySQL

MySQL is a relational database management system (RDBMS) which is more than 11 million institutions. The program runs as a server providing multi-user access to a number of databases.

MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now a subsidiary of Sun Microsystems, which holds the copyright to most of the code base. The project’s source code is available under terms of the GNU General Public License, as well as under a variety of proprietary agreements.

SELECTION OF OPERATING SYSTEM

Windows10 an Overview

Windows 10 includes improved network, application and Web services. It provides increased reliability and scalability, lowers your cost of computing with powerful, flexible management services, and provides the best foundation for running business application.

Network Data Security

Network data can be protected on the wire or at the network interface. Securing data at the network requires a firewall to proxy services and mediate connections between the internal network, (LAN) and external network (Internet). This is the purpose of Proxy Server.

Internet Protocol Security

Internet Protocol Security (IPSec) is a framework of open standards for ensuring secure private communications over Internet Protocol networks, using cryptographic security services.

INTERNET EXPLORER WEB BROWSER

INTERNET EXPLORER(IE) makes it easier to get the most from the world wide web, whether we are searching for new information or browsing your favorite website. and built in intelligence technology can save us time completing web tasks, such as automatically completing web addresses and form for, and automatically detecting your network and connection status.

ADVANTAGES OF IE

* When we start typing a frequently used web address in the address bar, a list of similar appears that you can choose from. And if a web page address is wrong, IE can search for similar addresses to try to find a match.
* Search for websites by clicking the search button on the toolbar.
* In the search bar type a word or phrase that describes what you are looking for.
* Go to other web pages similar to the one you are viewing without even doing a search. Just use the show related sites feature.
* Browse through the list of web pages you recently visited by clicking the history button on the tool bar.

CHAPTER 6

SYSTEM DEISGN

6.1 INTRODUCTION

System designing is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. It is a solution to a “how to” approach compared to system to a system analysis which is a “what is” orientation. It translates the system requirements into the ways of making them operational. The design phase focuses on the detailed implementation of the system recommended in the feasibility study

6.2 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system, modeling its process aspects. Often they are preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). So it is the starting point of design phase that functionally decomposes the requirements specifications down to the lowest level of detail. A DFD consists of a series of bubbles joined by lines.

DFD Symbols

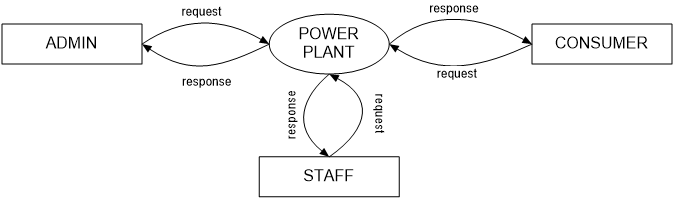
1) -------------> Source or destination

2) ------------> Process

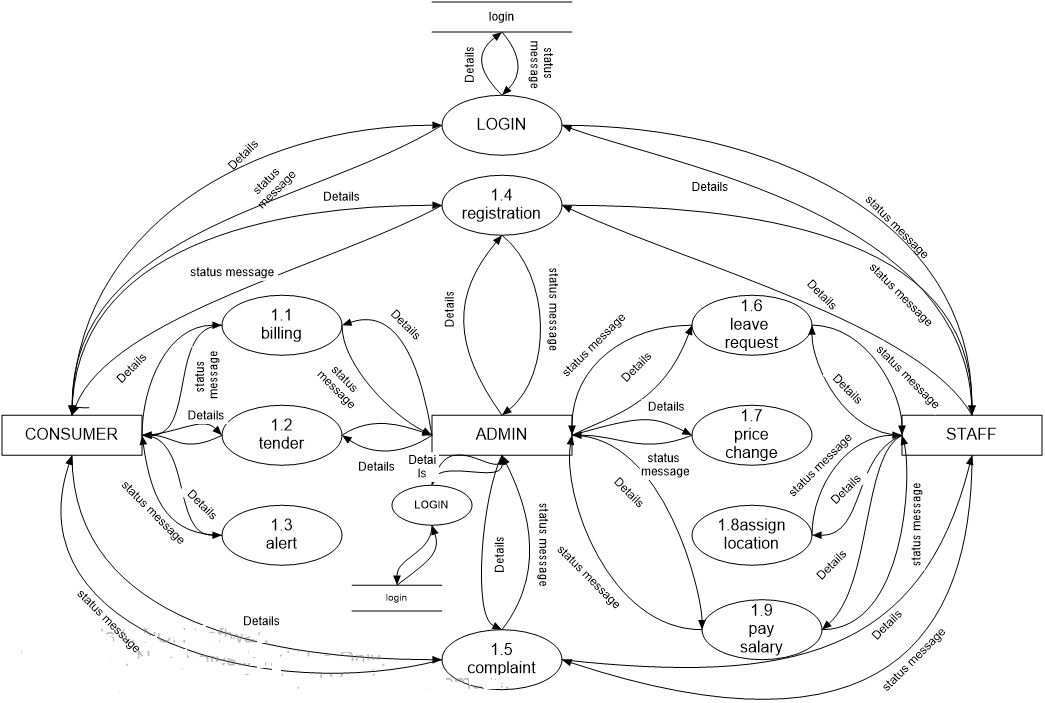
3) ------------> Data flow

4) ------------> Database

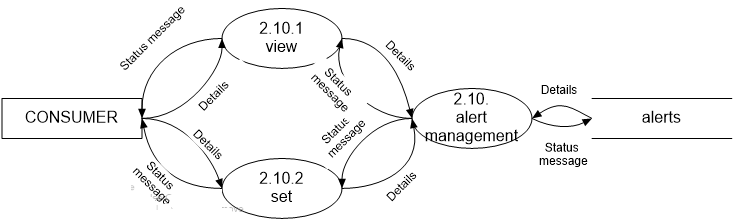
6.2.1 LEVEL 0 OF POWER PLANT MANAGEMENT

****

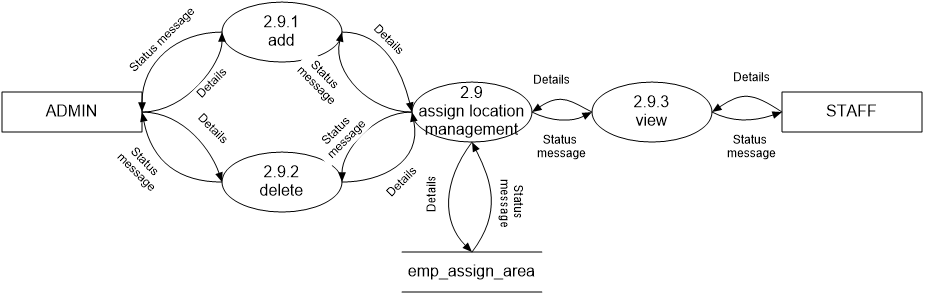
6.2.2 LEVEL 1 OF POWER PLANT MANAGEMENT



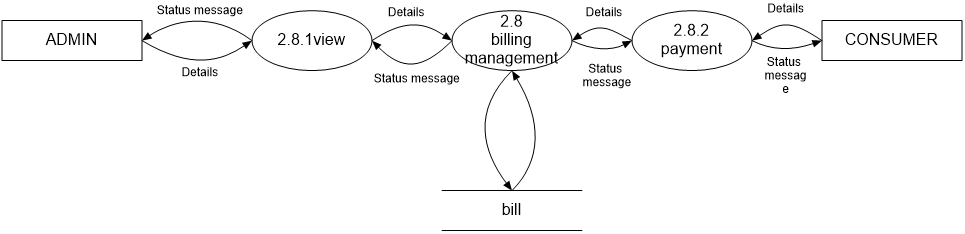
6.2.3 LEVEL 2 OF ALERT MANAGEMENT



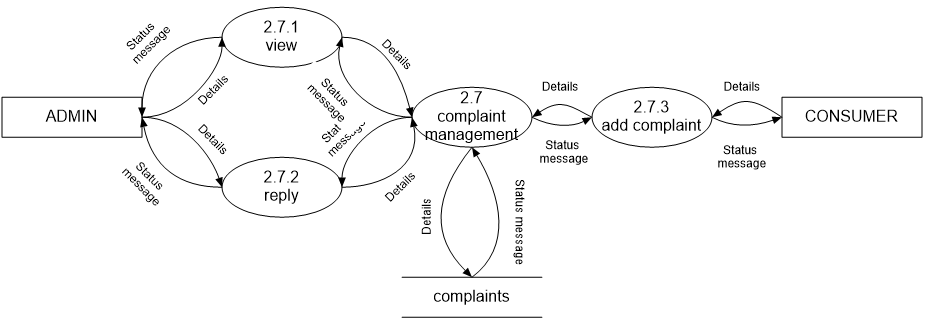
6.2.4 LEVEL 2 OF ASSIGN LOCATION MANAGEMENT



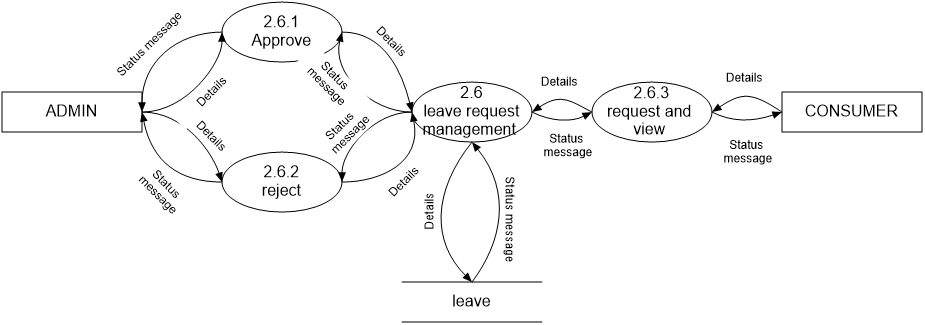
6.2.5 LEVEL 2 OF BILLING MANAGEMENT



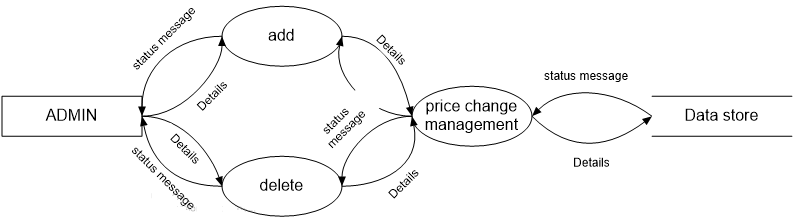
6.2.6 LEVEL 2 OF CONSUMER COMPLAINT MANAGEMENT



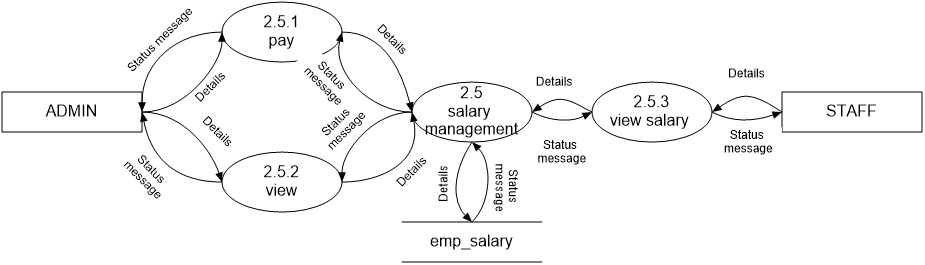
6.2.7 LEVEL 2 OF LEAVE REQUEST MANAGEMENT



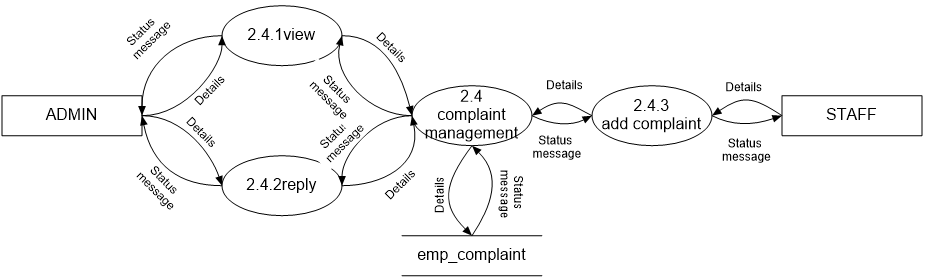
6.2.8 LEVEL 2 OF PRICE CHANGE MANAGEMENT



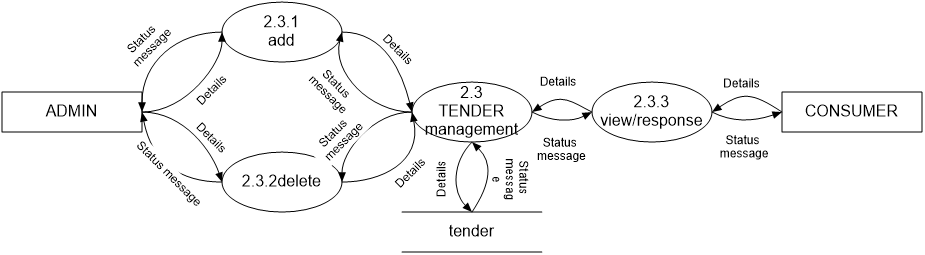
6.2.9 LEVEL 2 OF SALARY MANAGEMENT



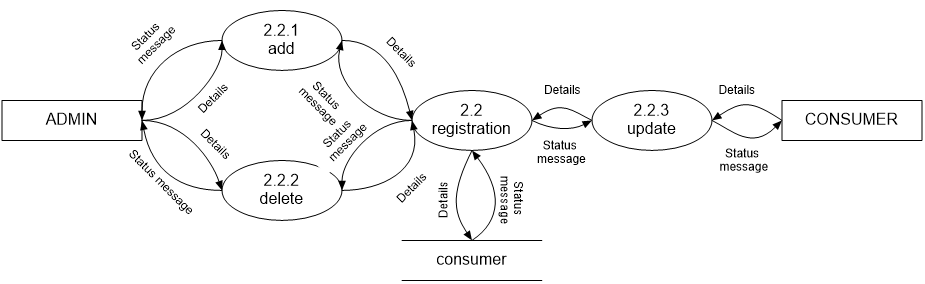
6.2.10 LEVEL 2 OF STAFF COMPLAINT MANAGEMENT



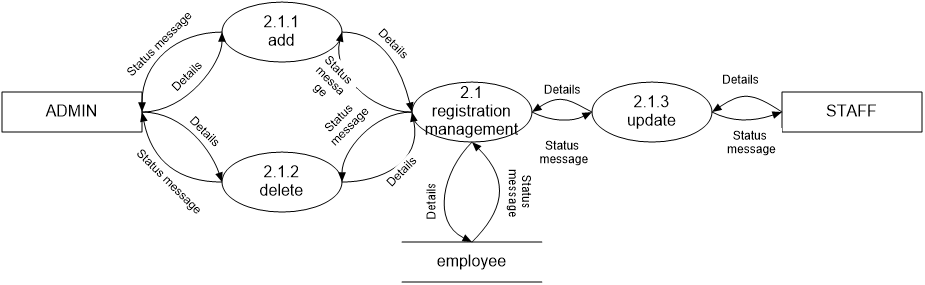
6.2.11 LEVEL 2 OF TENDER MANAGEMENT



6.2.12 LEVEL 2 OF CONSUMER REGISTRATION



6.2.13 LEVEL 2 OF STAFF REGISTRATION MANAGEMENT



6.3 INPUT DESIGN

Input design is the process of converting a user-oriented description of the inputs to a computer-based programmer-oriented specification. The quality of system input determines the quality of system output. Input specification describes the manner in which data enter the system for processing. Input design features can ensure the reliability of the system and produce result from accurate data, or they can result in the production or erroneous information. The input design also determines whether the user can interact efficiently with the system.

Input design requires consideration of the needs of the data entry operator.

Three data considerations are:

 The field length must be documented.

 The sequence of fields must match the sequence of the fields on the source document.

 The data format must be identified to the data entry operator.

In our system, almost all inputs are being taken from the database. To provide adequate inputs we have to select necessary values from the databases and arrange it to the appropriate controls.

Inaccurate input data are the most common cause of errors in data processing. Errors entered by data entry can be controlled by input design. Input design is the process of converting user-oriented inputs into a computer-based format. There are three major approaches in entering data into the computer. They are menus, formatted forms and prompts. A menu is a selection list that simplifies computer data access or entry. Instead of entering what to enter, the user chooses from a list of options and selects the option associated with it. A formatted form is a preprinted form or a template that request the user to enter data in appropriate location. It is a fill-in-the-blank type form. The form is flashed in the screen as a unit. In prompt the system displays one enquiry at a time, asking the user for a response.

6.4 OUTPUT DESIGN

One of the important features of an information system for users is the output it produces. Output is the information delivered to users through the information system. Without quality output, the entire system appears to be unnecessary that users will avoid using it. Users generally merit the system solely by its output. In order to create the most useful output possible. One works closely with the user through an interactive process, until the result is considered to be satisfactory. Output design generally deals with the result generated by the system i.e., reports. These reports can be generated from stored or calculated values. Reports are displayed either as screen window preview or

printed form. Most end users will not actually operate the information system or enter data through workstation, but they will use the output from the system.

Outputs from the computer systems are required to primarily to communicate the results of processing to the user. They are also used to provide a permanent copy of these results for later consultation.

6.5. DATABASE DESIGN

6.5.1 NORMALIZATION

Designing a database is a complex task and the normalization theory is a useful aid in this design process. The process of normalization is concerned with transformation of conceptual schema into computer representation form.

A bad database design may lead to certain undesirable situations such as,

 Repetition of information

 Inability to represent certain information

 Loss of information

To minimize these anomalies, normalization may be used. If the database is in a normalized form, the data can be restructured and can maintain it easily. This is important that the databases using that we are using may free from data redundancy and inconsistency. For this need we maintain the tables in a normalized manner.

6.5.1.1 First Normal Form

A relation is in first Normal Form (1NF), if and only if all its attributes are based on single domain. The objective of normalizing a table is in to remove its repeating groups and ensure that all entries of the resulting table have at most single value.

6.5.1.2 Second Normal Form

A table is said to be in second Normal Form (2 NF), when it is in 1 NF and every attribute in the record is functionally dependent upon the whole key, and not just a part of the key.

6.5.1.3 Third Normal Form

A table is in third Normal Form (3NF), when it is in 2NF and every non-key attribute is functionally dependent on just the primary key

6.6 TABLES

6.6.1 Table Name: Account Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| accountNo | int(10) | Primary Key |
| bank | varchar(45) |  |
| cardProvider | varchar(45) |  |
| cardKey | varchar(45) |  |
| cardType | varchar(45) |  |
| cardExp | varchar(45) |  |
| balance | varchar(45) |  |

6.6.2 Table Name: **alerts** table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| alertid | int(10) | Primary Key |
| consumer\_no | int(10) |  |
| limit | int(10) |  |
| status | varchar(45) |  |

6.6.3 Table Name: banks Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| ac\_no | varchar(45) | Primary Key |
| cvv\_no | varchar(45) |  |
| card\_no | varchar(45) |  |
| name | varchar(45) |  |
| card\_pin | varchar(45) |  |
| exp\_year | varchar(45) |  |
| exp\_month | varchar(45) |  |
| amount | double(12,4) |  |

6.6.4 Table Name: **bill** table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| bill\_id | Int(10) | Primary Key |
| consumer\_no | Varchar(45) |  |
| createdDate | Varchar(45) |  |
| amount | double |  |
| paidDate | Varchar(45) |  |
| billStatus | Varchar(45) |  |
| fineDate | Varchar(45) |  |
| lastDate | Varchar(45) |  |
| netamt | Varchar(45) |  |
| account | int(10) |  |

6.6.5 Table Name: **bill\_slab** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| slab\_id | Int(10) | Primary Key |
| unit\_limit\_below | int(10) |  |
| unit\_limit\_top | int(11) |  |
| amount | int(10) |  |
| slab\_name | varchar(45) |  |

6.6.6 Table Name: **complaints** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| comp\_no | Int(10) | Primary Key |
| consumer\_no | varchar(45) |  |
| category | varchar(45) |  |
| description | Varchar(45) |  |
| person | varchar(45) |  |
| adrs | varchar(45) |  |
| mobile | varchar(45) |  |
| areas | varchar(45) |  |
| pincode | varchar(45) |  |
| dates | varchar(45) |  |
| reply | varchar(45) |  |
| status | varchar(45) |  |

6.6.7 Table Name **consumer** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| consumer\_no | varchar(20) | Primary Key |
| cname | varchar(45) |  |
| address | varchar(45) |  |
| dob | varchar(45) |  |
| pincode | Varchar(20) |  |
| area | varchar(45) |  |
| plot | varchar(45) |  |
| street | varchar(45) |  |
| city | varchar(45) |  |
| phone | varchar(45) |  |
| mobile | varchar(45) |  |
| email | varchar(45) |  |
| con\_status | varchar(45) |  |

6.6.8 Table Name: **emp\_assign\_area** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| assign\_id | Int(10) | Primary Key |
| emp\_id | varchar(45) |  |
| pincode | varchar(45) |  |

6.6.9 Table Name: **emp\_complaint** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| cmp\_id | Int(10) | Primary Key |
| date | varchar(45) |  |
| complaint | varchar(45) |  |
| reply | varchar(45) |  |
| status | varchar(45) |  |
| emp\_id | varchar(45) |  |

6.6.10 Table Name: **emp\_salary** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| salary\_id | Int(10) | Primary Key |
| emp\_id | varchar(20) |  |
| amount | Int(10) |  |
| accountno | varchar(20) |  |
| date | varchar(45) |  |

6.6.11 Table Name: **employee** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| emp\_id | varchar(45) | Primary Key |
| address | varchar(45) |  |
| contact | varchar(45) |  |
| designation | varchar(45) |  |

6.6.12 Table Name: **history** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| hid\_id | int(10) | Primary Key |
| day | varchar(45) |  |
| usage | int(10) |  |
| consumer\_no | varchar(45) |  |

6.6.13 Table Name: **leave** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| leave\_id | int(10) | Primary Key |
| emp\_id | varchar(45) |  |
| start\_date | varchar(45) |  |
| reason | varchar(450) |  |
| days | varchar(45) |  |
| status | varchar(45) |  |
| request\_date | varchar(25) |  |

6.6.14 Table Name: load Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| load\_id | int(10) | Primary Key |
| consumer\_no | varchar(45) |  |
| total | int(10) |  |

6.6.15 Table Name: **load\_history** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| his\_id | int(10) | Primary Key |
| consumer\_no | varchar(45) |  |
| history\_date | varchar(45) |  |
| load | varchar(45) |  |

6.6.16 Table Name: **login** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| login\_id | int(10) | Primary Key |
| username | varchar(45) |  |
| password | varchar(45) |  |
| role | varchar(45) |  |

6.6.17 Table Name: **setlimit** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| consumer\_no | int(10) | Primary Key |
| lm | varchar(45) |  |
| msg\_status | varchar(45) |  |

6.6.18 Table Name: **tender** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| tender\_id | int(10) | Primary Key |
| work\_description | varchar(450) |  |
| date | varchar(45) |  |
| status | varchar(45) |  |

6.6.19 Table Name: **unit\_price** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| unit\_price\_id | int(10) | Primary Key |
| h\_price | int(10) |  |

6.6.20 Table Name: **user\_tender** Table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Constraint |
| user\_tender\_id | int(10) | Primary Key |
| tender\_id | int(10) |  |
| user\_id | varchar(45) |  |
| date | varchar(45) |  |
| tender\_description | varchar(45) |  |

CHAPTER 7

SOFTWARE TESTING

SOFTWARE TESTING

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, Testing presents an interesting anomaly for the software engineer.

Testing Objectives include:

1. Testing is a process of executing a program with the intent of finding an error.

2. A good test case is one that has a probability of finding an as yet undiscovered error.

3. A successful test is one that uncovers an undiscovered error.

Testing Principles:

* All tests should be traceable to end user requirements
* Tests should be planned long before testing begins
* Testing should begin on a small scale and progress towards testing in large
* Exhaustive testing is not possible.
* To be most effective testing should be conducted by a independent third party.

TESTING STRATEGIES

A strategy for software testing integrates software test case design method in to a well-planned series of steps that result in the successful construction of the software. The strategy provides a road map that describes the step to be conducted as part of testing, when these steps are planned and then undertaken, and how much effort, time and resources will be required. Therefore any testing strategy must incorporate test planning, test case, design, test execution and resultant data collection and evaluation. A software testing strategy should be flexible enough to promote a customized testing approach. At the same time, it must be rigid enough to promote reasonable planning and management tracking as the project progresses. The general characteristics of software testing strategies are

* Testing begins at the component level and works “outward” toward the integration of the entire computer system.
* Different testing techniques are appropriate at different points in time.

A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements. A strategy must provide guidance for the practitioner and set of mild stones for the

manager. Because the step on the test strategy occurs at a time when deadline pressure begins to rise, progress must be measurable, and problem must surface as early as possible.

The software team’s approach to testing is defining a plan that describes an overall strategy and a procedure that defines specific testing steps and tests that will be conducted. In the proposed system, if the administrator makes any attempt to login to the application without entering his password, then the system will not allow the user to login to the application.

TESTING TECHNIQUES

The various testing techniques are given below

White-Box Testing

White-box testing, also called as glass-box testing, is a test case design method that goes to the control structure of the procedural design to derive test cases. Using white-box testing methods, the software engineer can derive test cases that

* Guarantee that all independent paths within a module have been exercised at least once.
* Exercise all logical decision on their true and false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structure to ensure their validity.

Black-box testing was successfully conducted on our system. The system was divided into a number of modules and testing was conducted on each module. We have tested the system for incorrect or missing functions and interface errors. Performance errors and the flow of information between modules ensuring interface.

Unit Testing

Unit testing comprises the set of tests performed by an individual programmer prior to the integration of the system. Testing removes residual bugs and improves the reliability of the system. Testing allows the developer to find out the design faults if any, and enable correction if needed. Exhaustive unit testing has to be carried out to ensure the validity of the data. In order to successfully test the entire package, unit testing is carried out. Each module was tested as and when it was developed. Thus, it proved easier to conduct minute testing operation and correct them then and there.

Integration Testing

A subsystem consists of several modules that communicate with each other through well defined interfaces. Normally, a subsystem implements a major segment of the total system. The primary purpose of the subsystem testing is to verify operation of the interfaces between modules in the subsystem. Both control and data interfaces must be tested. Large software system may require several levels of subsystem testing. Lower level subsystems are successively combined to form higher level subsystems. In most software systems, exhaustive testing of subsystem capabilities is not feasible due to the combination complexity of the module interfaces. Therefore, test cases must be carefully chosen to exercise the interfaces in the desired manner.

Output Testing

Output testing of the proposed system is important since no system could be useful if it does not produce the required output.

The output format on the screen is found to be correct, as the format was designed in the system design phase according to the user needs. For the hard copy also the output comes out as the specified requirements by the user. Hence output testing doesn’t result in any correction on the system.

CHAPTER 8

IMPLEMENTATION AND MAINTENANCE

8.1. INTRODUCTION

Implementation is that process plan where the theoretical design is put into real test. All the theoretical and practical works are now implemented as a working system. This is most crucial stage in the life cycle of a project. The project may be accepted or rejected depending on how it gathers confidence among the users. The implementation stage involves the following tasks

* Careful planning
* Investigation of systems and constraints
* Design of methods to achieve the changeover
* Training of the staff in the changeover phase
* Evaluation of the changeover method

8.2. INSTALLATION PROCEDURE

Implementation of software refers to final installation of package in the real environment, to the satisfaction of the intended users and the successful operation of the system. Implementation is the stage of the project where the theoretical design is turned into a working system. Implementation includes all those activities that takes place to convert from the old system to new one. Proper implementation is essential to provide a reliable system to meet the organizational requirements.

8.3. IMPLEMENTATION PLAN

The Implementation Plan describes how the information system will be deployed, installed and transitioned into an operational system. The plan contains an overview of the system, a brief description of the major tasks involved in the implementation, the overall resources needed to support the implementation effort, and any site-specific implementation requirements. The plan is developed during the Design Phase and is updated during the Development Phase the final version is provided in the Integration and Test Phase and is used for guidance during the Implementation Phase.

8.4. SYSTEM MAINTANENCE

The definition of system maintenance can be given by describing four activities that are undertaken after the program is released for use. The first maintenance activity occurs since it is unreasonable to assume that software will uncover all errors in a larger software system. The process of including the diagonals and correction of one or more errors is called corrective maintenance. The second activity that contributes to a definition of maintenance occurs since rapid change is encountered in every aspect of

computing. Therefore, adaptive maintenance modifies software to properly interface with a changing environment. The third activity involves recommendation for new capabilities, modification to the existing function and general enhancement when the software is used. To satisfy requests, prefecture maintenance is performed. The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability. This is called preventive maintenance.

CHAPTER 9

CONCLUSION

The fundamental goal of the project is to develop an POWE PALNT MANAGEMENT. The software was developed and tested with sample data and outputs obtained according to the requirements. The performance of the system is evaluated and is found to be much more efficient than the existing system though it cannot be claimed that it is not an ideal project, it will meet the primary requirements of the concern.

So the project has to be improved by having modifications as and when the necessity arises in due course. Any system, which has been in use for a number of years, gradually, decays and become less effective because of the change in environment to which one has to adapt. For a time it is possible to overcome problems by amendments and minor modification to acknowledge the need of fundamental changes. Computerization was proposed as a solution to the problem of being outdated with the fast present technologies. The project meets all the requirements as described in the proposed system and satisfies user requirements to a great extent. The software thus developed is easy to maintain and is quite user friendly.

9.1. FUTURE ENHANCEMENT

The World of computer is not always static. It is always subject to change. The software too has to suit the requirements of the future. The system is open to reasonable changes and these changes can be bought very easily. But drastic changes like changes in the processing flow, changes in system functioning etc. may not able to accommodate in the existing system. It can be strongly said that the system has a fair scope for future development that doesn’t affect the complete process flow severely. Efficiency can be further enhanced and boosted up to a great extent by normalizing and de-normalizing the database tables used in the project as well as taking the kind of the alternative set of data structures and advanced calculation algorithms available. We can in future generalize the application from its current customized status wherein other vendors developing and working on similar applications can utilize this software and make changes to it according to their business needs.

CHAPTER 10

BIBILIOGRAPHY

BOOK REFERENCES

1. Software Engineering, Pankaj Jalote

WEBSITE REFERENCES

1. [www.wikipedia.com](http://www.wikipedia.com)

2. www.scribd.com

3. www.investopedia.com/walkthrough

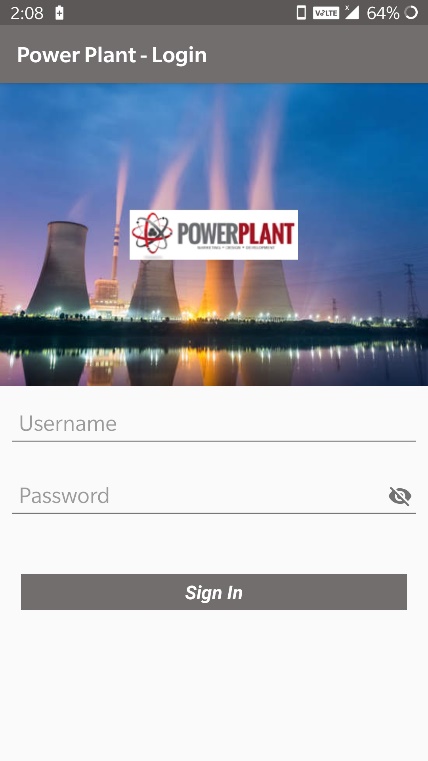
4. [www.codexworld.com](http://www.codexworld.com)

CHAPTER 11

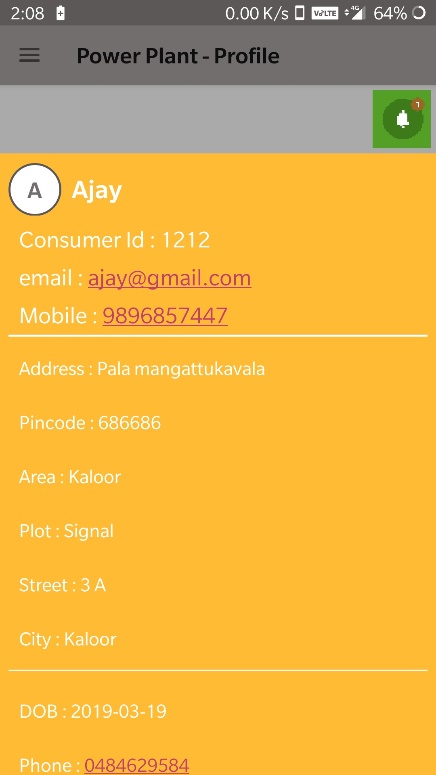
APPENDIX

11.1 Appendix A:Screen Layout

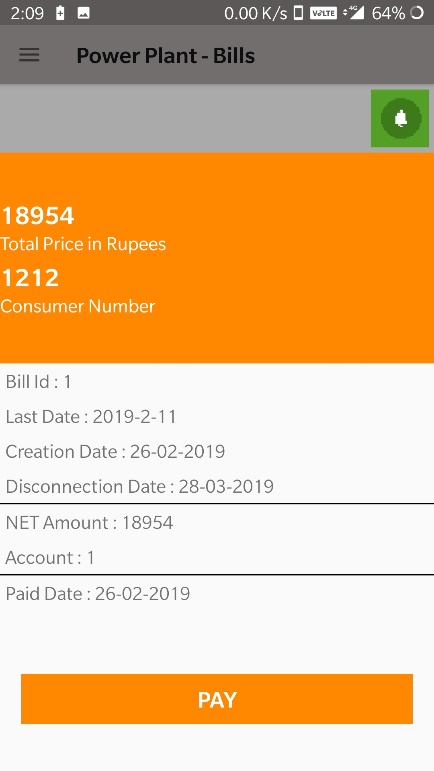
11.1.1 Login Android

****

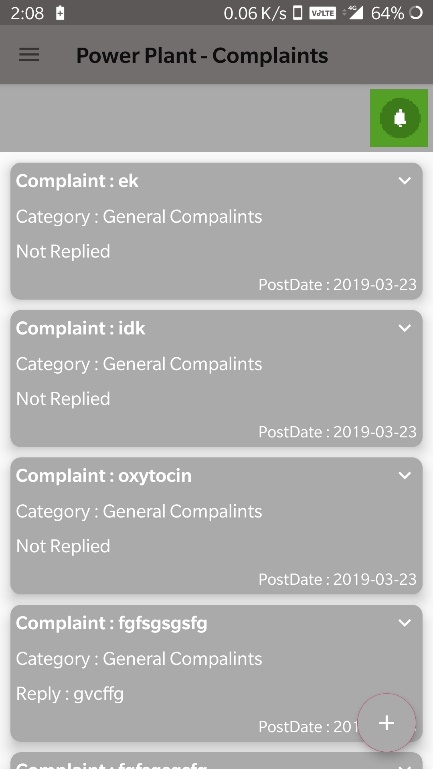
11.1.2 Consumer Profile Android

****

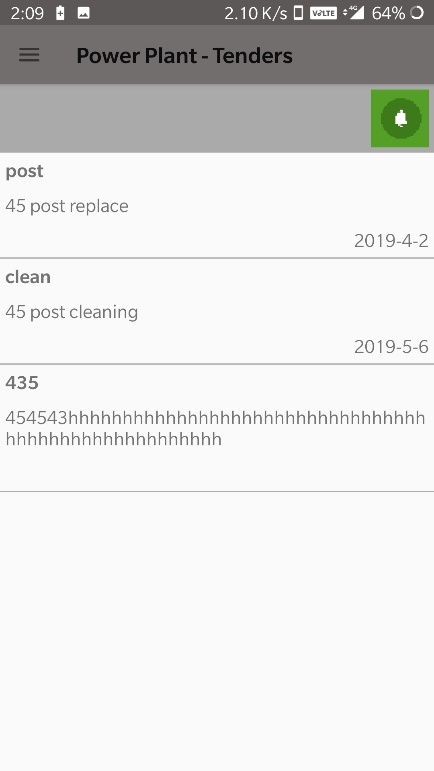
11.1.3 Consumer Billing Android

****

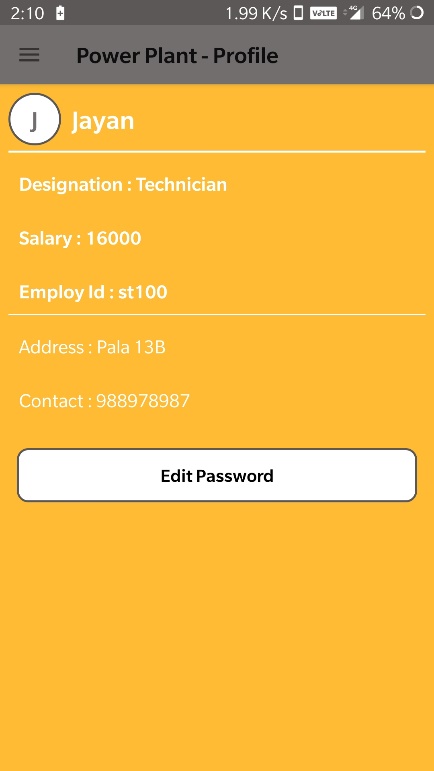
11.1.4 Consumer Complaints Android

****

11.1.5 Consumer View Tender Android

****

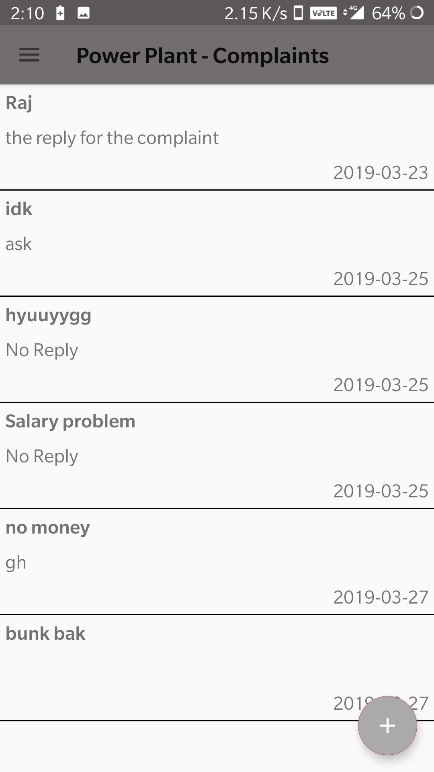
11.1.6 Staff Profile Android

****

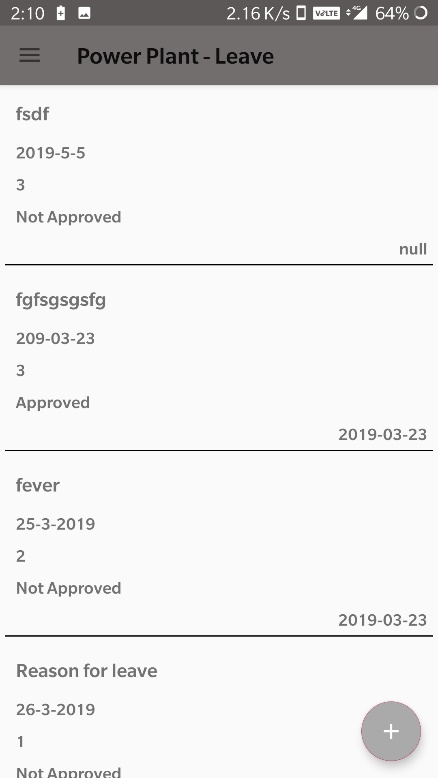
11.1.7 Staff Work View Android

****

11.1.8 Staff complaint Android

****

11.1.9 Staff leave Android

****

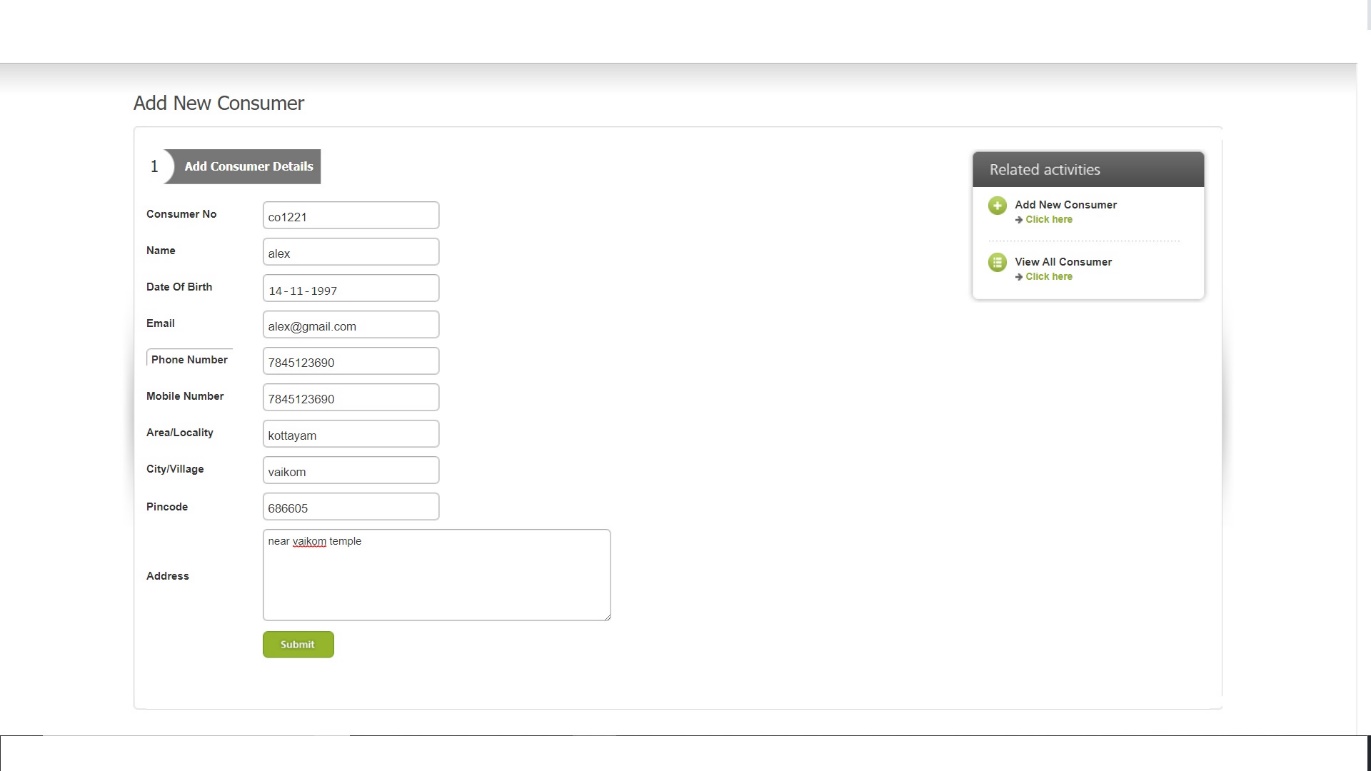
11.1.10 WEB load page

****

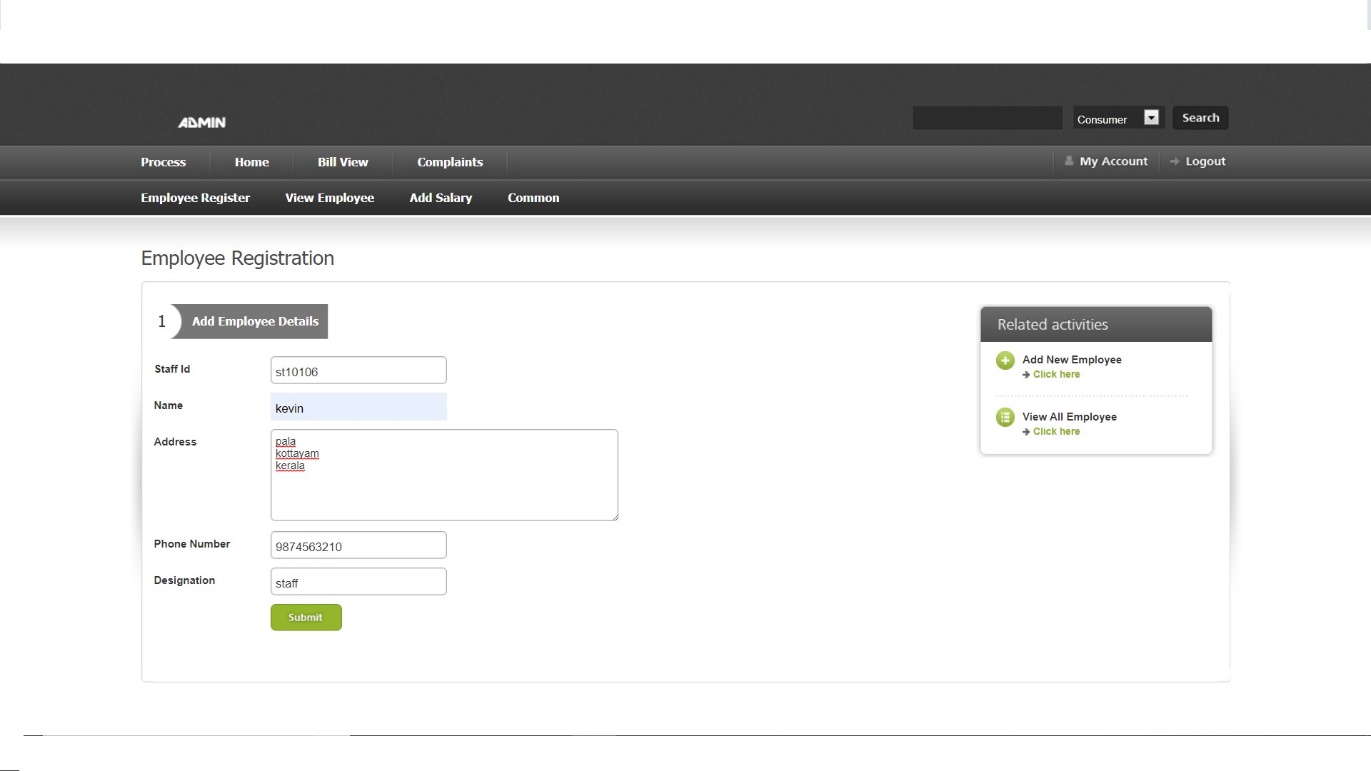
11.1.11 WEB login

****

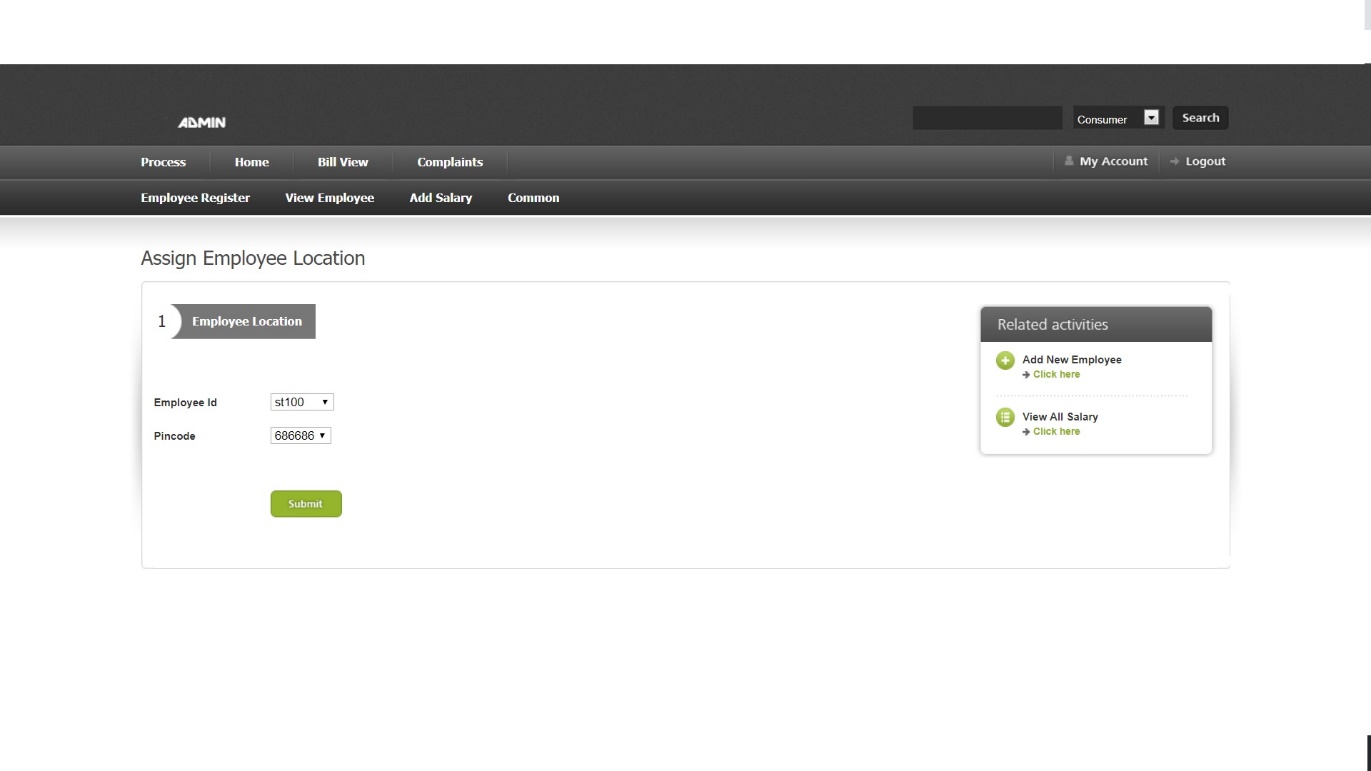
11.1.12 WEB consumer registration

****

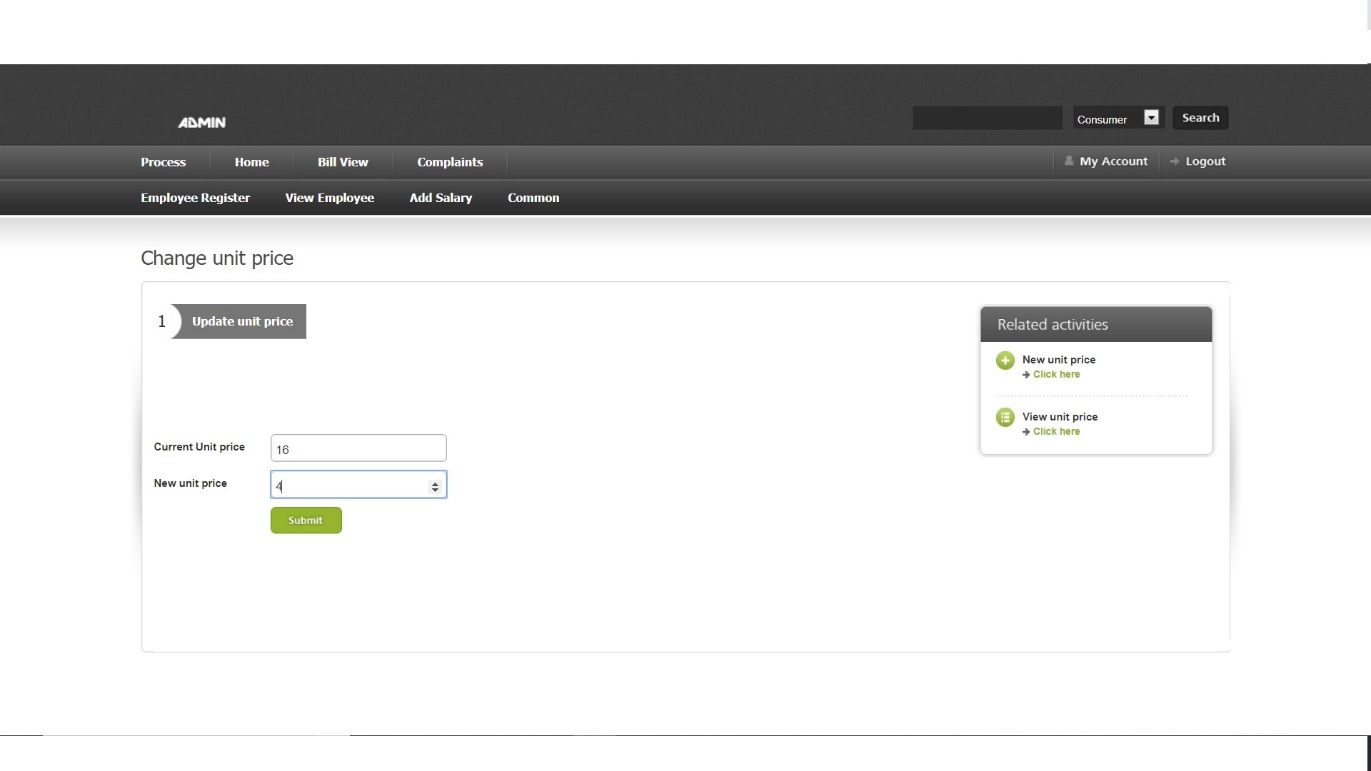
11.1.13 WEB staff registration

****

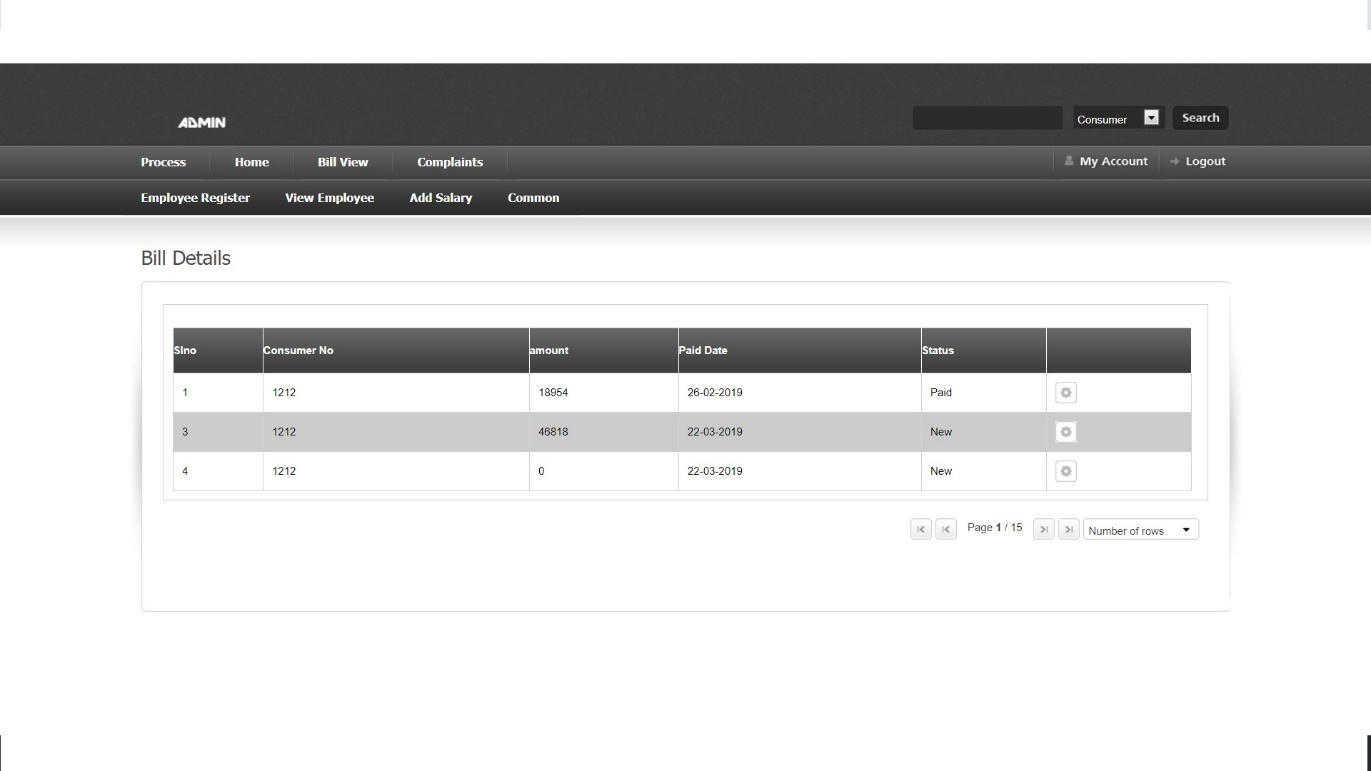
11.1.14 WEB assign location

****

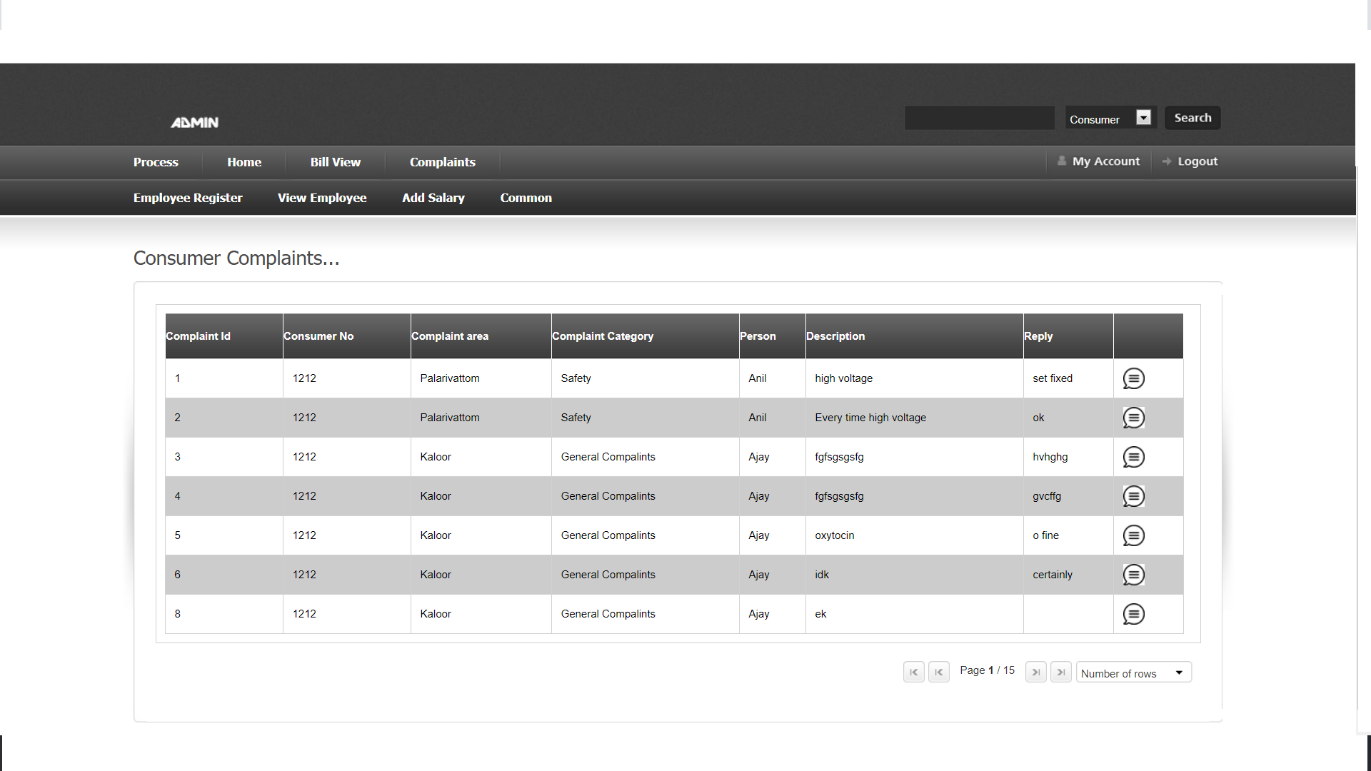
11.1.15 WEB change price

****

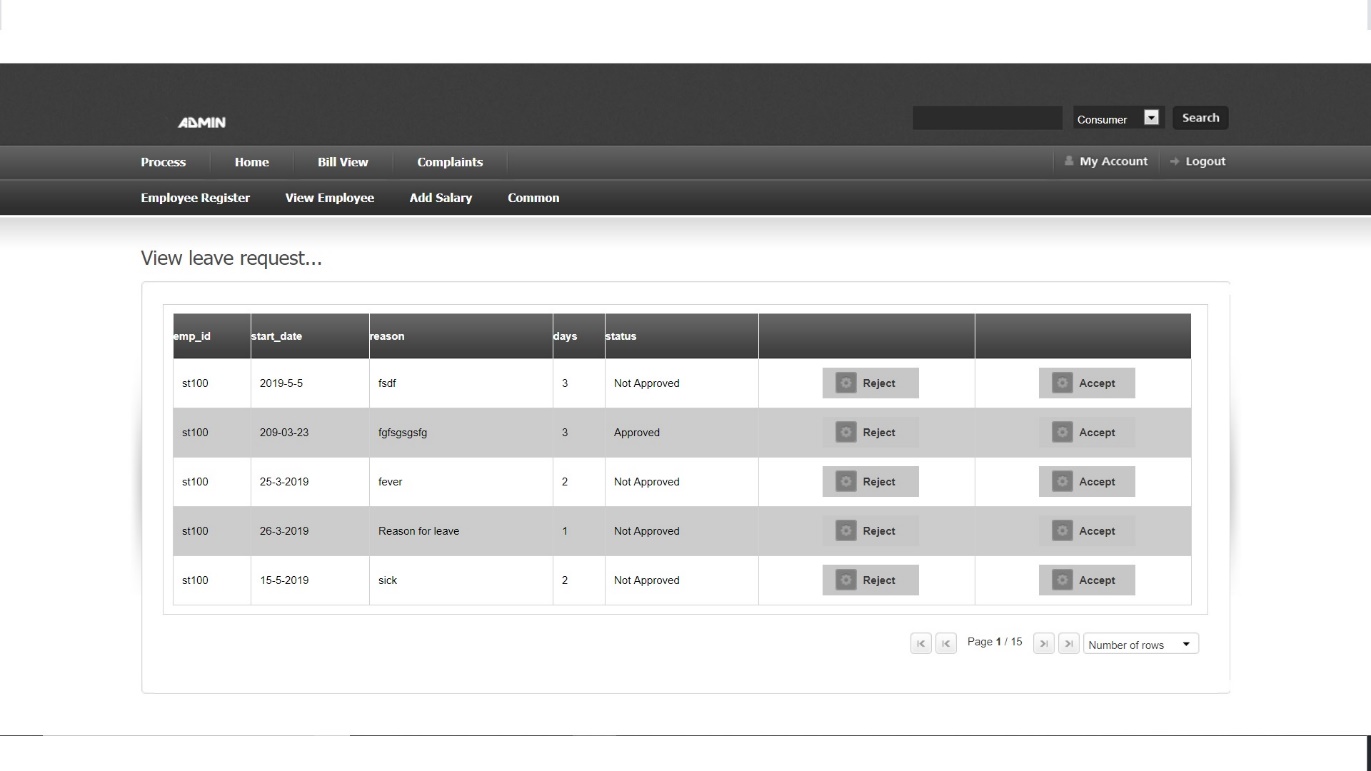
11.1.16 WEB bill view



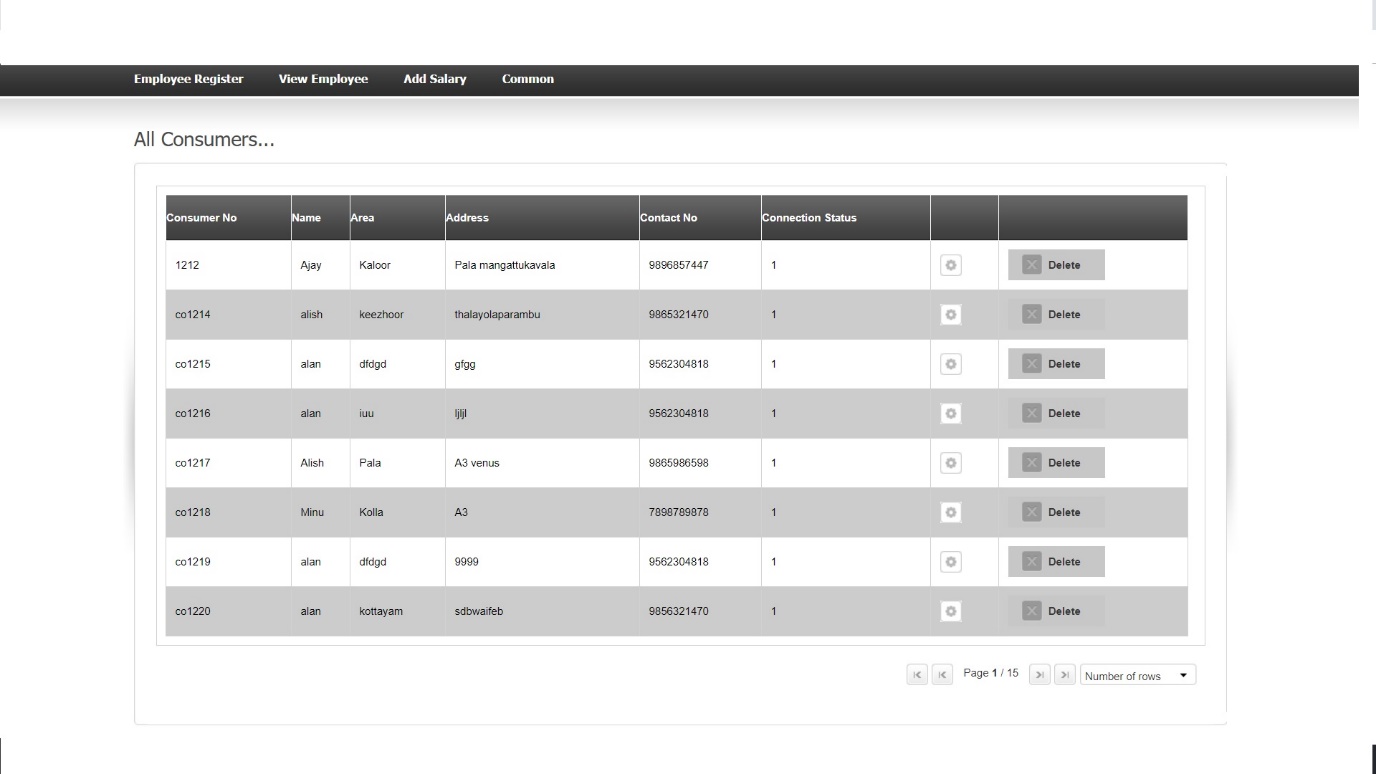
11.1.17 WEB consumer complaint

****

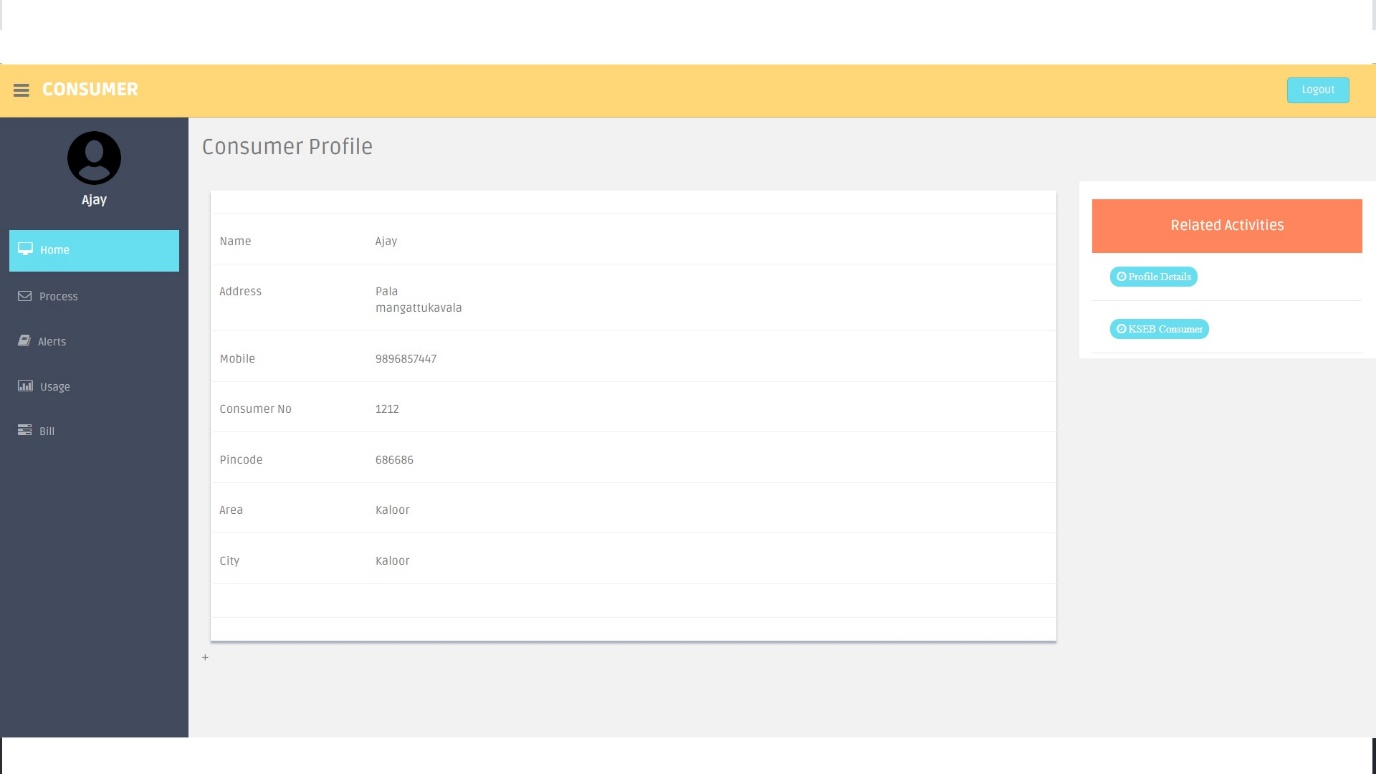
11.1.18 WEB leave request

****

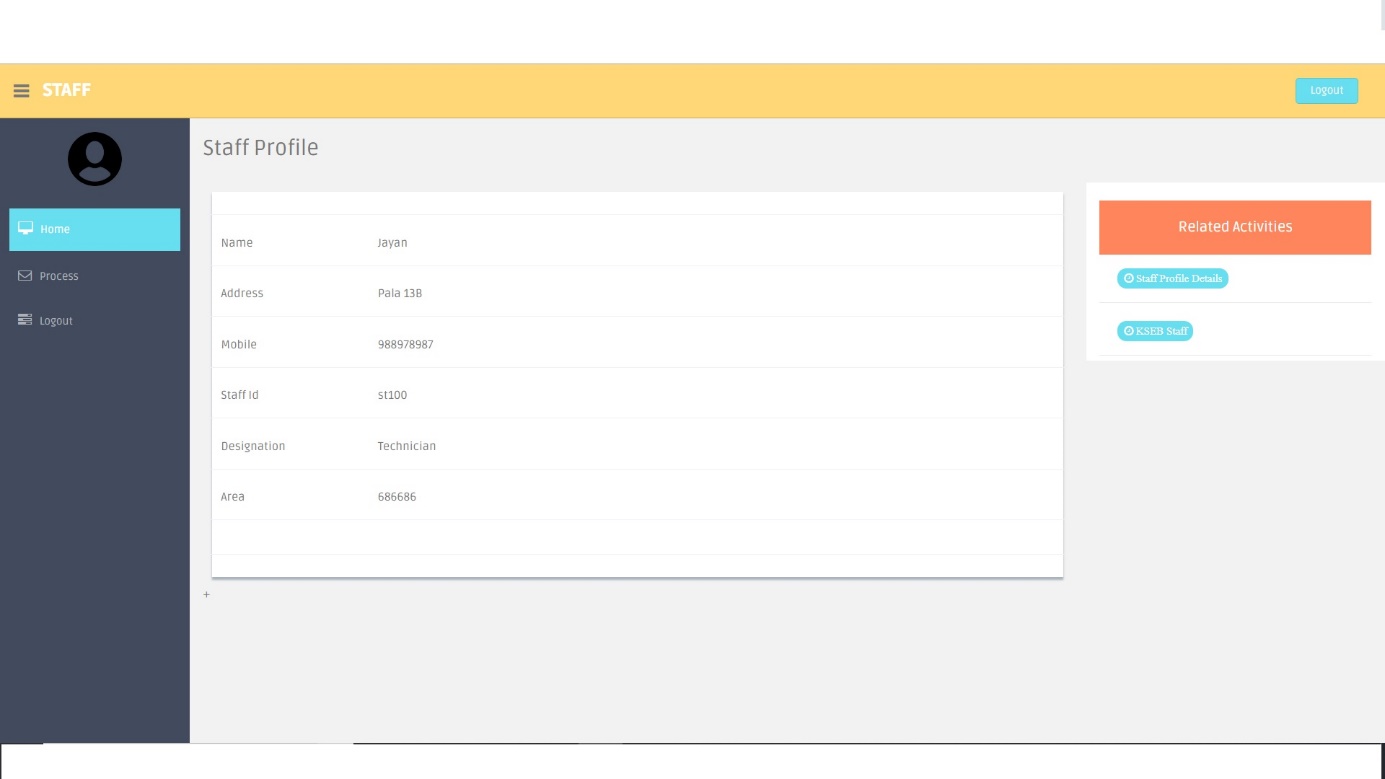
11.1.19 View consumers

****

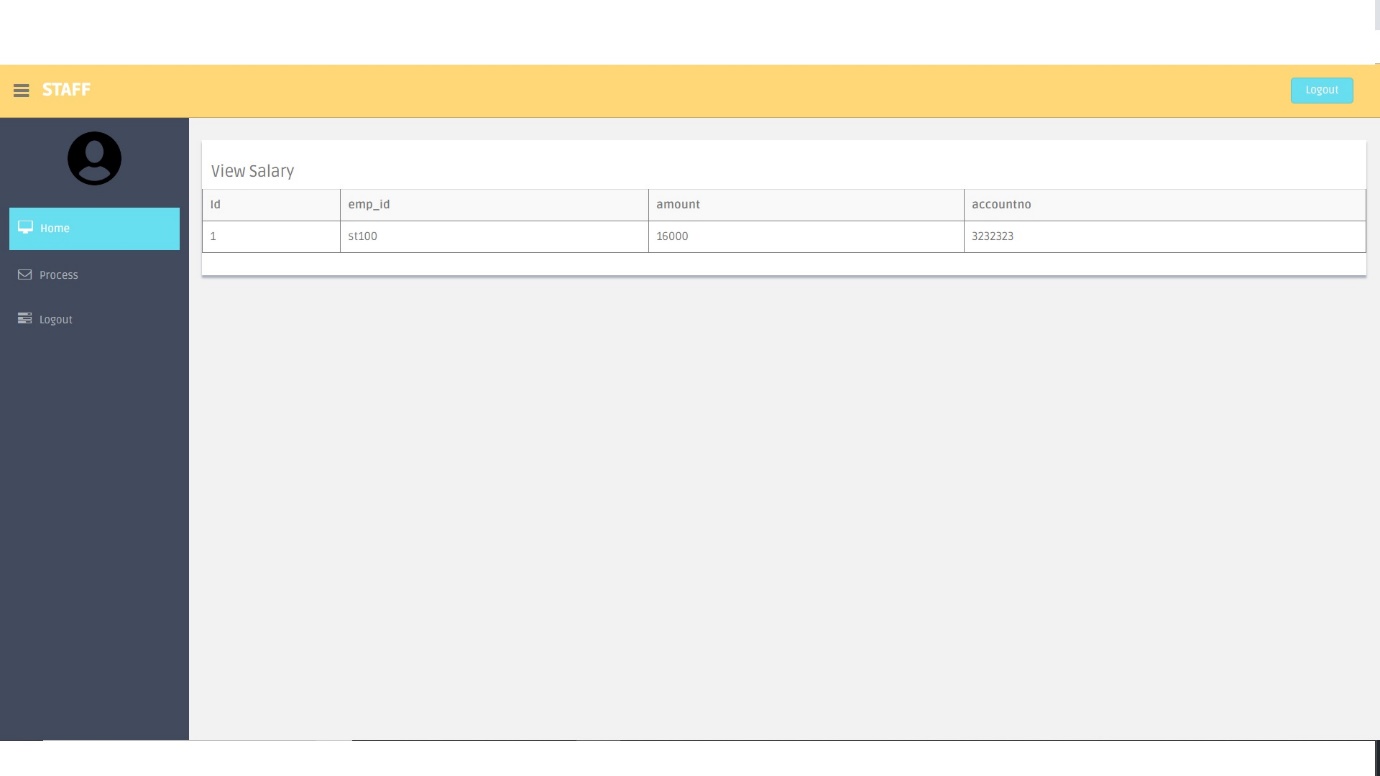
11.1.20 WEB consumer profile

****

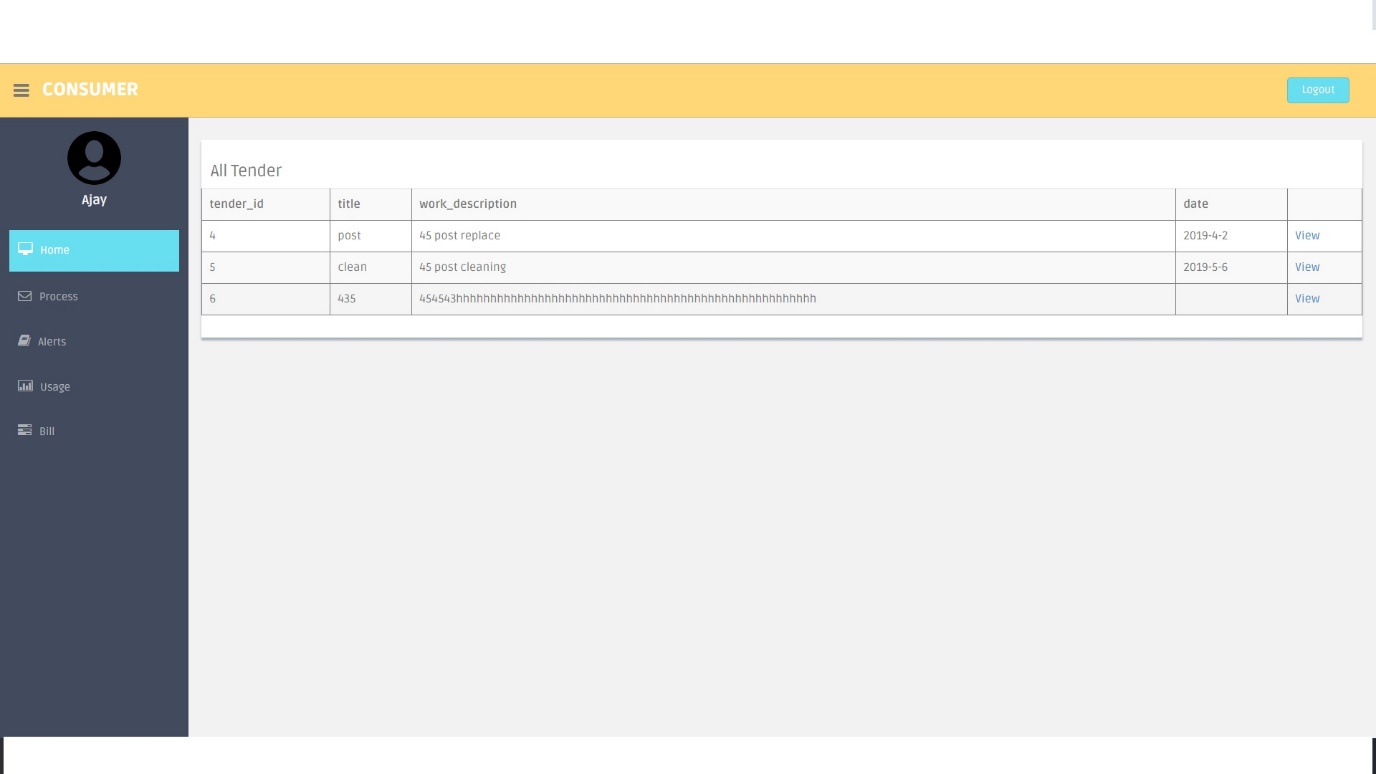
11.1.21 WEB staff profile

****

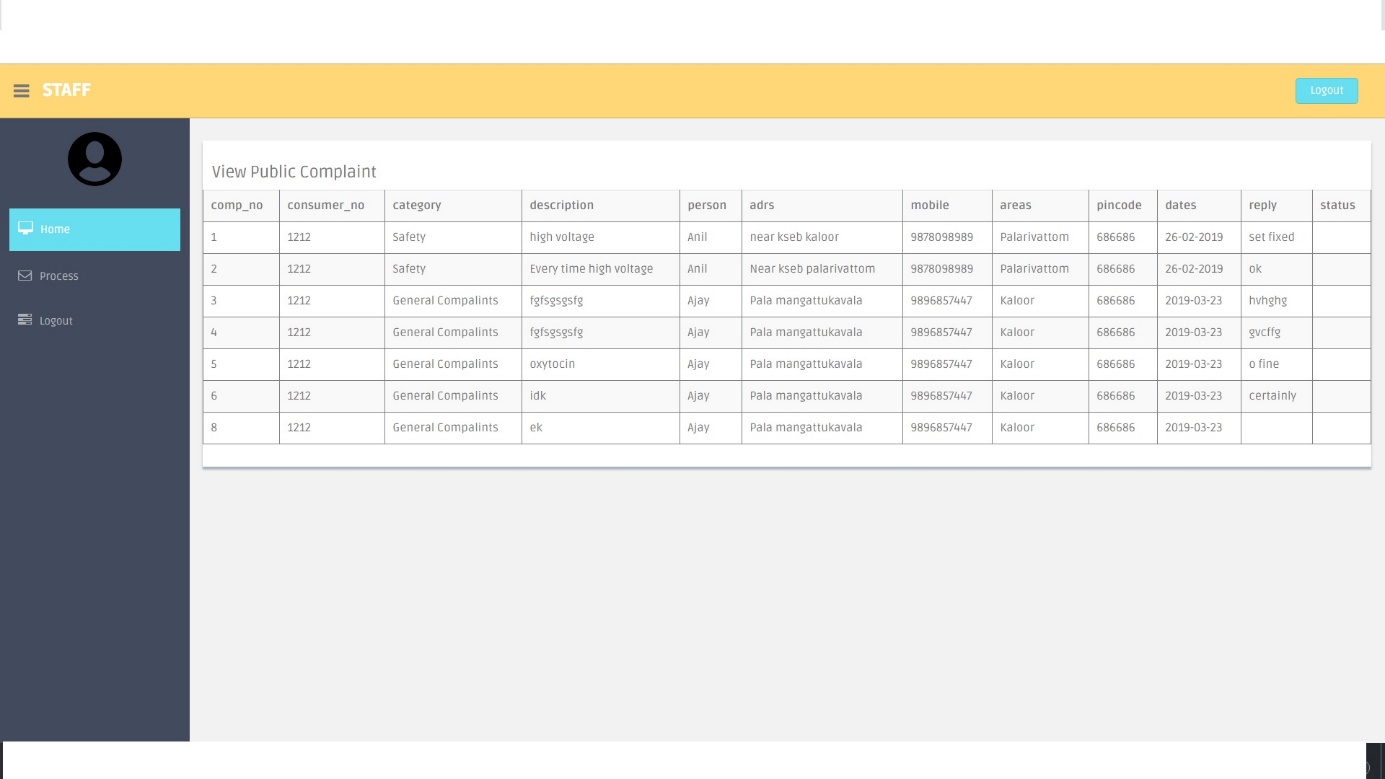
11.1.22 WEB staff salary view

****

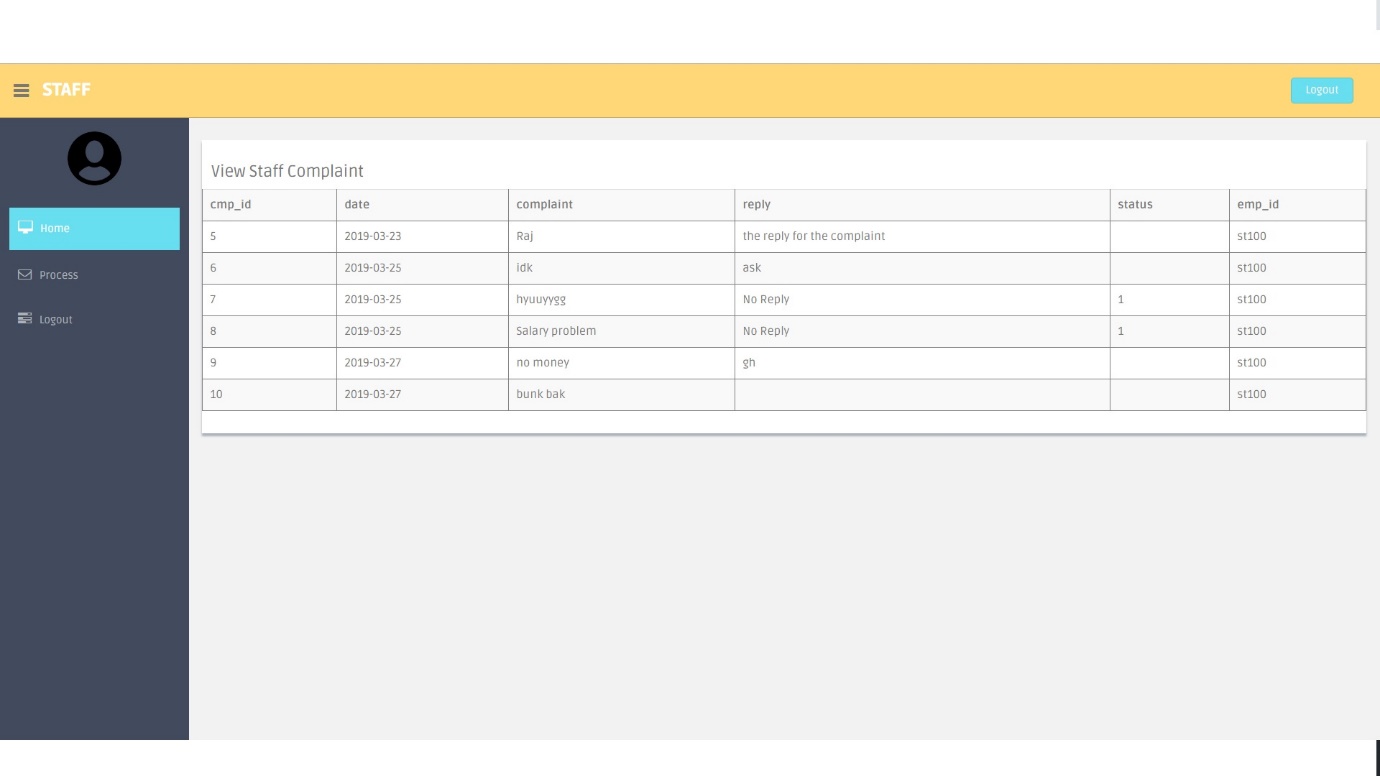
11.1.23 WEB consumer tender view

****

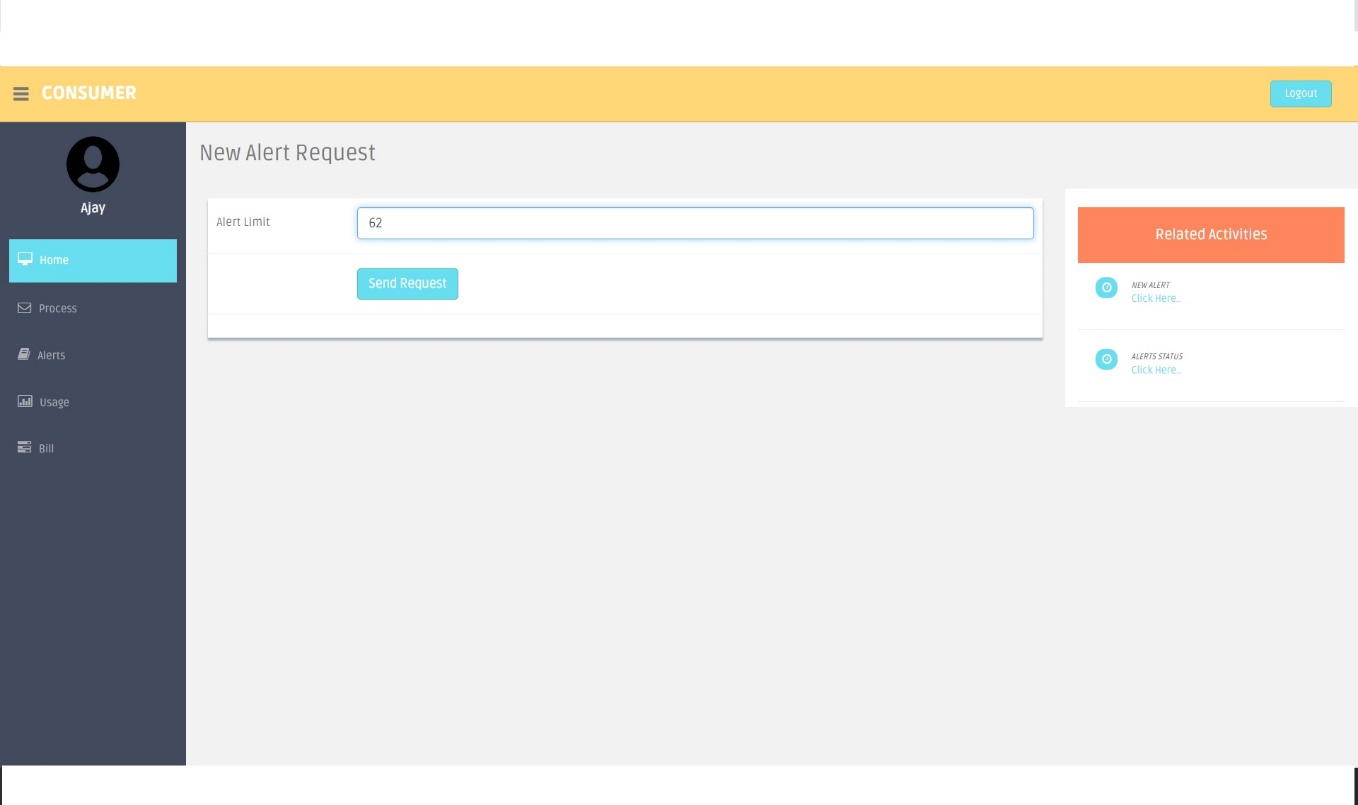
11.1.24 WEB consumer complaint

****

11.1.25 WEB staff complaint view

****

11.1.26 WEB consumer alert

****

11.2 Appendix B:Program Codes

11.2.1 Login Page Android

package lapis.apps.lapislearning.kseb.activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.support.design.widget.TextInputEditText;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.View;

import android.widget.ProgressBar;

import android.widget.TextView;

import android.widget.Toast;

import com.google.gson.JsonObject;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import lapis.apps.lapislearning.kseb.MainActivityNavDrawer;

import lapis.apps.lapislearning.kseb.R;

import lapis.apps.lapislearning.kseb.Rectrofit.ResponseCallback;

import lapis.apps.lapislearning.kseb.Rectrofit.ResponseHandler;

import lapis.apps.lapislearning.kseb.Rectrofit.Retrofit\_Helper;

import lapis.apps.lapislearning.kseb.SessionManager;

import lapis.apps.lapislearning.kseb.Utility;

import lapis.apps.lapislearning.kseb.datamodels.DataUser;

import retrofit2.Call;

public class LoginActivity extends AppCompatActivity {

TextInputEditText userNameTextInputEditText,passwordTextInputEditText ;

TextView signUpTextView,signInTextView ;

public ArrayList<DataUser> dataUser;

public ProgressBar progIndicator;

private SessionManager session;//global variable

@Override

protected void onStart() {

super.onStart();

String uid = session.getuid();

if (!uid.isEmpty()){

Utility.UID = uid;

Utility.USERROLE =session.getuserrole();

Utility.NAME = session.getNameUser();

if (Utility.USERROLE.equals("staff")){

Utility.PINCODE = session.getPincode();

}else{

// Utility.lm = session.getLm();

}

// Utility.MOBILE = session.getMobileUser();

startActivity(new Intent(getApplicationContext(), MainActivityNavDrawer.class));

}

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

getSupportActionBar().setTitle(getString(R.string.app\_name)+" - Login");

session = new SessionManager(getApplicationContext());

init();

signInTextView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

boolean stat = validate();

if (stat){

progIndicator.setVisibility(View.VISIBLE);

getLogin();

}else {

Toast.makeText(LoginActivity.this, "Login Failed"+stat, Toast.LENGTH\_SHORT).show();

}

}

});

// signUpTextView.setOnClickListener(new View.OnClickListener() {

// @Override

// public void onClick(View view) {

// Intent i = new Intent(LoginActivity.this,UserRegistrationActivity.class);

// startActivity(i);

// finish();

// }

// });

}

public boolean validate() {

boolean stat = true;

if (userNameTextInputEditText.getText().toString().trim().equals("")){

stat = false;

userNameTextInputEditText.setError("Field required");

}

if (passwordTextInputEditText.getText().toString().trim().equals("")){

passwordTextInputEditText.setError("Field required");

stat = false;

}

return stat;

}

public void init(){

userNameTextInputEditText = findViewById(R.id.username\_textinputedittext);

passwordTextInputEditText = findViewById(R.id.password\_textinputedittext);

signUpTextView = findViewById(R.id.signup\_textview);

signInTextView = findViewById(R.id.signin\_textview);

progIndicator = findViewById(R.id.progIndication);

}

@Override

public void onBackPressed() {

new AlertDialog.Builder(this)

.setTitle("Confirm Exit!")

.setMessage("Do want to close the application?")

.setNegativeButton(android.R.string.no, null)

.setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface arg0, int arg1) {

finishAffinity();

finish();

}

}).create().show();

}

//i. {"uid":"abc","upwd":"abcp"}

private void getLogin() {

Map<String, String> params = new HashMap<>();

params.put("uid", ""+userNameTextInputEditText.getText().toString().trim());

params.put("upwd", ""+passwordTextInputEditText.getText().toString().trim());

//i. {“input”:”hi hello”,”uid”:””}

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.LOGINURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(LoginActivity.this, new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

//{"result":[{"login\_id":"2","username":"1212","password":"1212","role":"user","show\_alert":1,"alert\_msg":"Your Usage 578. You have exceeded the limit 25"}]}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("result")) {

JSONArray login\_array = jsonObj.getJSONArray("result");

dataUser = new ArrayList<>();

DataUser ca;

// {"login\_id":"2","username":"1212","password":"1212","role":"user","name":"Ajay","lm":"34","show\_alert":0}

//{"login\_id":"3","username":"st100","password":"st100","role":"staff","pincode":"686686","name":"Jayan"}

// for(int k=0;k<1;k++){

ca=new DataUser();

actor = login\_array.getJSONObject(0);

ca.setLogin\_id(actor.getString("login\_id"));

ca.setUsername(actor.getString("username"));

ca.setPassword(actor.getString("password"));

ca.setRole(actor.getString("role"));

ca.setNameU(actor.getString("name"));

if (actor.getString("role").equals("staff")){

ca.setPinCode(actor.getString("pincode"));

}else {

ca.setShow\_alert(actor.getString("show\_alert"));

ca.setLimitU(actor.getString("lm"));

if (actor.has("alert\_msg"))

ca.setAlert\_msg(actor.getString("alert\_msg"));

}

Log.d("getSearchList", jsonObj.toString());

dataUser.add(ca);

// }

if (!dataUser.isEmpty()){

Utility.UID = dataUser.get(0).getUsername();

Utility.USERROLE = dataUser.get(0).getRole();

Utility.NAME = dataUser.get(0).getNameU();

session.setuid(Utility.UID);

session.setuserrole(Utility.USERROLE);

session.setNameUser(Utility.NAME);

// Intent i = new Intent(getApplicationContext(),MainActivity.class);

Intent i = new Intent(getApplicationContext(),MainActivityNavDrawer.class);

if (Utility.USERROLE.equals("staff")){

Log.d("getResponse: ",""+dataUser.get(0).getPinCode());

session.setPincode(dataUser.get(0).getPinCode());

Utility.PINCODE = dataUser.get(0).getPinCode();

}else{

// Utility.lm = dataUser.get(0).getLimitU();

// session.setLm(Utility.lm);

if (dataUser.get(0).getShow\_alert().equals("1")){

i.putExtra("alert\_msg",dataUser.get(0).getAlert\_msg());

}}

startActivity(i);

finish();

}else{

progIndicator.setVisibility(View.GONE);

Toast.makeText(LoginActivity.this, "Login Failed", Toast.LENGTH\_SHORT).show();

}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

progIndicator.setVisibility(View.GONE);

Toast.makeText(getApplicationContext(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

progIndicator.setVisibility(View.GONE);

Toast.makeText(getApplicationContext(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

}

11.2.3 Android Billing

public class BillsFragment extends Fragment {

TextView billId,consumerNumberBill,createdDateBill,amountBill,paidDateBill,payBtn,

fineDateBill,lastDateBill,netAmountBill,accountBill;

ImageView alertIV;

RelativeLayout relativeLayout ;

public ArrayList<DataBill> dataBills;

// <!-- billId,consumerNumberBill,createdDateBill,amountBill,

// paidDateBill,billStatus,fineDateBill,

// lastDateBIll,netAmountBill,accountBill;-->

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.bills\_fragment, container, false);

/\*\*Get String from Bundle which is passed from MainActivityNavDrawer.java\*\*/

String text = getArguments().getString("text", "");

init(view);

getBill();

if (Utility.USERROLE.equals("staff"))

relativeLayout.setVisibility(View.GONE);

else

relativeLayout.setVisibility(View.VISIBLE);

alertIV.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Utility.alertAdd(getActivity());

}

});

payBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

addPayment(amountBill.getText().toString().trim(),billId.getText().toString().trim());

Toast.makeText(getActivity(), "PAY Action", Toast.LENGTH\_SHORT).show();

}

});

return view;

}

private void init(View view) {

billId = view.findViewById(R.id.idBill);

consumerNumberBill = view.findViewById(R.id.consumerNumberBill);

createdDateBill = view.findViewById(R.id.createdDateBill);

amountBill = view.findViewById(R.id.amountBill);

paidDateBill = view.findViewById(R.id.paidDate);

fineDateBill = view.findViewById(R.id.fineDateBill);

lastDateBill = view.findViewById(R.id.lastDateBill);

netAmountBill = view.findViewById(R.id.netAmountBill);

accountBill = view.findViewById(R.id.accountBill);

payBtn = view.findViewById(R.id.payBtn);

alertIV = view.findViewById(R.id.alertiv);

relativeLayout = view.findViewById(R.id.rL);

//"http://i.imgur.com/Vth6CBz.gif"

Glide.with(getActivity())

.load(Utility.URLT)

.override(40,40)

.centerCrop()

.diskCacheStrategy(DiskCacheStrategy.ALL)

.placeholder(android.R.drawable.stat\_sys\_download)

.error(android.R.drawable.stat\_notify\_error)

.into(alertIV);

}

private void getBill() {

Map<String, String> params = new HashMap<>();

params.put("cno", ""+ Utility.UID);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.SEARCHURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("result")) {

JSONArray login\_array = jsonObj.getJSONArray("result");

dataBills = new ArrayList<>();

DataBill ca;

// for(int k=0;k<login\_array.length();k++){

ca=new DataBill();

//{"bill\_id":"1","consumer\_no":"1212","createdDate":"26-02-2019","amount":"18954","paidDate":"26-02-2019",

// "billStatus":"New","fineDate":"28-03-2019","lastDate":"2019-2-11","netamt":"18954","account":"1"}

actor = login\_array.getJSONObject(0);

ca.setBillId(actor.getString("bill\_id"));

ca.setConsumerNumberBill(actor.getString("consumer\_no"));

ca.setCreatedDateBill(actor.getString("createdDate"));

ca.setAmountBill(actor.getString("amount"));

ca.setPaidDateBill(actor.getString("paidDate"));

ca.setBillStatus(actor.getString("billStatus"));

ca.setFineDateBill(actor.getString("fineDate"));

ca.setLastDateBIll(actor.getString("lastDate"));

ca.setNetAmountBill(actor.getString("netamt"));

ca.setAccountBill(actor.getString("account"));

Log.d("getmessagesList", jsonObj.toString());

dataBills.add(ca);

// }

// recyclerViewAdapterTenders = new RecyclerViewAdapterTenders(getActivity(),dataTenders);

// recyclerViewRV.setAdapter(recyclerViewAdapterTenders);

if (!dataBills.isEmpty()){

DataBill obj = dataBills.get(0);

//billId,consumerNumberBill,createdDateBill,amountBill,paidDateBill,payBtn,fineDateBill,lastDateBill,

// netAmountBill,accountBill;

billId.setText("Bill Id : "+obj.getBillId());

consumerNumberBill.setText(obj.getConsumerNumberBill());

createdDateBill.setText("Creation Date : "+obj.getCreatedDateBill());

amountBill.setText(obj.getAmountBill());

paidDateBill.setText("Paid Date : "+obj.getPaidDateBill());

fineDateBill.setText("Disconnection Date : "+obj.getFineDateBill());

lastDateBill.setText("Last Date : "+obj.getLastDateBIll());

netAmountBill.setText("NET Amount : "+obj.getNetAmountBill());

accountBill.setText("Account : "+obj.getAccountBill());

}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

public void addPayment(final String amt, final String bill\_id){

final Dialog dialog = new Dialog(getActivity());

dialog.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

dialog.setContentView(R.layout.new\_payment\_layout);

dialog.setCancelable(true);

WindowManager.LayoutParams lp = new WindowManager.LayoutParams();

lp.copyFrom(dialog.getWindow().getAttributes());

lp.width = WindowManager.LayoutParams.MATCH\_PARENT;

lp.height = WindowManager.LayoutParams.WRAP\_CONTENT;

final AppCompatButton sendAppCompatButton = dialog.findViewById(R.id.postBtn);

final AppCompatButton cancelAppCompatButton = dialog.findViewById(R.id.cancelBtn);

final TextView totalAmountTV = dialog.findViewById(R.id.totalamountBill);

final TextView billIdPaymentTV = dialog.findViewById(R.id.paymentbillid);

final TextInputEditText acnoTIET = dialog.findViewById(R.id.accountNumberTIET);

final TextInputEditText acnameTIET = dialog.findViewById(R.id.accountNameTIET);

final TextInputEditText cvvTIET = dialog.findViewById(R.id.cvvTIET);

final TextInputEditText cardTIET = dialog.findViewById(R.id.cardTIET);

final TextInputEditText cpinTIET = dialog.findViewById(R.id.pinTIET);

final TextInputEditText expyearTIET = dialog.findViewById(R.id.yearTIET);

final TextInputEditText expmonthTIET = dialog.findViewById(R.id.monthTIET);

// cpinTIET.addTextChangedListener(new TextWatcher() {

// @Override

// public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

// }

// @Override

// public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

// sendAppCompatButton.setEnabled(!charSequence.toString().trim().isEmpty());

// }

// @Override

// public void afterTextChanged(Editable editable) {

// }

// });

cancelAppCompatButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

dialog.dismiss();

}

});

sendAppCompatButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

boolean stat = true;

// acno,acname,cvv,card,cpin,expyear,expmonth,amt,bill\_id;

String acno= acnoTIET.getText().toString().trim();

String acname = acnameTIET.getText().toString().trim();

String cvv = cvvTIET.getText().toString().trim();

String card = cardTIET.getText().toString().trim();

String cpin = cpinTIET.getText().toString().trim();

String expyear = expyearTIET.getText().toString().trim();

String expmonth = expmonthTIET.getText().toString().trim();

if (acname.isEmpty()){

stat = false;

acnameTIET.setError("Field Required");

Toast.makeText(getActivity(), "Account Name Field Empty", Toast.LENGTH\_SHORT).show();

}else if (acno.isEmpty()){

stat = false;

acnoTIET.setError("Field Required");

Toast.makeText(getActivity(), "Account Number Field Empty", Toast.LENGTH\_SHORT).show();

}else if (cvv.isEmpty()){

stat = false;

cvvTIET.setError("Field Required");

Toast.makeText(getActivity(), "CVV Field Empty", Toast.LENGTH\_SHORT).show();

}else if (card.isEmpty()){

stat = false;

cardTIET.setError("Field Required");

Toast.makeText(getActivity(), "Card Field Empty", Toast.LENGTH\_SHORT).show();

}else if (cpin.isEmpty()){

stat = false;

cpinTIET.setError("Field Required");

Toast.makeText(getActivity(), "Pin Field Empty", Toast.LENGTH\_SHORT).show();

}else if (expyear.isEmpty()){

stat = false;

expyearTIET.setError("Field Required");

Toast.makeText(getActivity(), "Expiry Year Field Empty", Toast.LENGTH\_SHORT).show();

}else if (expmonth.isEmpty()){

stat = false;

expmonthTIET.setError("Field Required");

Toast.makeText(getActivity(), "Expiry Month Field Empty", Toast.LENGTH\_SHORT).show();

}

if (stat == true){

setPayment(acno,acname,cvv,card,cpin,expyear,

expmonth,amt,bill\_id);

dialog.dismiss();

}

}

});

dialog.show();

totalAmountTV.setText("Total Amount : "+amt+" Rs");

billIdPaymentTV.setText(""+bill\_id);

dialog.getWindow().setAttributes(lp);

// dialog.getWindow().setBackground(Color.TRANSPARENT);

}

private void setPayment(String acno,String acname,String cvv,String card,String cpin,String expyear,

String expmonth,String amt,String bill\_id) {

Map<String, String> params = new HashMap<>();

//{"acno":"","acname":"","cvv":"","card":"","cpin":"","expyear":"","expmonth":"","amt":"","bill\_id":""}

params.put("acno", ""+acno);

params.put("acname", ""+acname);

params.put("cvv", ""+cvv);

params.put("card", ""+card);

params.put("cpin", ""+cpin);

params.put("expyear", ""+expyear);

params.put("expmonth", ""+expmonth);

params.put("amt", ""+amt);

params.put("bill\_id", ""+bill\_id);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.PAYAMOUNTURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("status")&&jsonObj.getInt("status")>0) {

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

Toast.makeText(getActivity(), "Payment Completed Successfully", Toast.LENGTH\_SHORT).show();

getBill();

} else

Toast.makeText(getActivity(), "Payment failed,Try again", Toast.LENGTH\_SHORT).show();

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

}

11.2.4 Consumer complaint android

public class ComplaintsFragment extends Fragment {

RecyclerView recyclerViewRV;

ImageView alertIV;

RelativeLayout relativeLayout ;

FloatingActionButton addComplaintBtn;

public ArrayList<DataComplaint> dataComplaints;

public RecyclerViewAdapterComplaints recyclerViewAdapterComplaints;

public boolean rresult = false;

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.complaints\_fragment, container, false);

/\*\*Get String from Bundle which is passed from MainActivityNavDrawer.java\*\*/

String text = getArguments().getString("text", "");

init(view);

addComplaintBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

addComplaint();

}

});

RecyclerView.LayoutManager mLayoutManager = new GridLayoutManager(getActivity(), 1, GridLayoutManager.VERTICAL,false);

recyclerViewRV.setNestedScrollingEnabled(true);

recyclerViewRV.setLayoutManager(mLayoutManager);

recyclerViewRV.setItemAnimator(new DefaultItemAnimator());

getComplaints();

if (Utility.USERROLE.equals("staff"))

relativeLayout.setVisibility(View.GONE);

else

relativeLayout.setVisibility(View.VISIBLE);

alertIV.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Utility.alertAdd(getActivity());

}

});

return view;

}

private void init(View view) {

recyclerViewRV = view.findViewById(R.id.recyclerview\_rv);

addComplaintBtn = view.findViewById(R.id.addComplaintBtn);

alertIV = view.findViewById(R.id.alertiv);

relativeLayout = view.findViewById(R.id.rL);

//"http://i.imgur.com/Vth6CBz.gif"

Glide.with(getActivity())

.load(Utility.URLT)

.override(40,40)

.centerCrop()

.diskCacheStrategy(DiskCacheStrategy.ALL)

.placeholder(android.R.drawable.stat\_sys\_download)

.error(android.R.drawable.stat\_notify\_error)

.into(alertIV);

}

private void getComplaints() {

Map<String, String> params = new HashMap<>();

params.put("uid", ""+ Utility.UID);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.LISTALLCOMPLAINTURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("result")) {

JSONArray login\_array = jsonObj.getJSONArray("result");

dataComplaints = new ArrayList<>();

DataComplaint ca;

for(int k=0;k<login\_array.length();k++){

ca=new DataComplaint();

//{"comp\_no":"25","cons\_no":"1212","category":"GeneralCompalints","description":"Hkkjhh","person":"anoop",

// "adrs":"Gghhj","mobile":"9847131977","areas":"yty","dist":"","dates":"2017-10-19","reply":"8545m,l"}

actor = login\_array.getJSONObject(k);

ca.setComplaintNumber(actor.getString("comp\_no"));

ca.setConsumerNumber(actor.getString("consumer\_no"));

ca.setCategoryComplaint(actor.getString("category"));

ca.setDescriptionComplaint(actor.getString("description"));

ca.setPersonComplaint(actor.getString("person"));

ca.setAddressComplaint(actor.getString("adrs"));

ca.setMobileComplaint(actor.getString("mobile"));

ca.setAreaComplaint(actor.getString("areas"));

ca.setDistrictComplaint(actor.getString("pincode"));

ca.setDatecomplaint(actor.getString("dates"));

ca.setReplyComplaint(actor.getString("reply"));

Log.d("getmessagesList", jsonObj.toString());

dataComplaints.add(ca);

}

recyclerViewAdapterComplaints = new RecyclerViewAdapterComplaints(getActivity(),dataComplaints);

recyclerViewRV.setAdapter(recyclerViewAdapterComplaints);

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

public void addComplaint(){

final Dialog dialog = new Dialog(getActivity());

dialog.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

dialog.setContentView(R.layout.new\_complaint\_layout)

dialog.setCancelable(true);

WindowManager.LayoutParams lp = new WindowManager.LayoutParams();

lp.copyFrom(dialog.getWindow().getAttributes());

lp.width = WindowManager.LayoutParams.MATCH\_PARENT;

lp.height = WindowManager.LayoutParams.WRAP\_CONTENT;

final AppCompatButton sendAppCompatButton = dialog.findViewById(R.id.postBtn);

final TextInputEditText contentTIET = dialog.findViewById(R.id.contentTIET);

// final TextInputEditText consumerNumberTIET = dialog.findViewById(R.id.consumerNumberTIET);

// final TextInputEditText nameTIET = dialog.findViewById(R.id.personTIET);

// final TextInputEditText addressTIET = dialog.findViewById(R.id.addressAddTIET);

// final TextInputEditText mobileTIET = dialog.findViewById(R.id.phoneTIET);

contentTIET.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

sendAppCompatButton.setEnabled(!charSequence.toString().trim().isEmpty());

}

@Override

public void afterTextChanged(Editable editable) {

}

});

sendAppCompatButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

boolean stat = true;

String contentString = contentTIET.getText().toString().trim();

// String cno = consumerNumberTIET.getText().toString().trim();

// String nm = nameTIET.getText().toString().trim();

// String adr = addressTIET.getText().toString().trim();

// String mob = mobileTIET.getText().toString().trim();

// if (nm.isEmpty()){

// stat = false;

// nameTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Name Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (cno.isEmpty()){

// stat = false;

// consumerNumberTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Consumer Number Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (adr.isEmpty()){

// stat = false;

// addressTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Address Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (mob.isEmpty()){

// stat = false;

// mobileTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Phone Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (mob.length()<10||mob.length()>10){

// stat = false;

// mobileTIET.setError("Valid Number Required");

// Toast.makeText(getActivity(), "Invalid Phone Number", Toast.LENGTH\_SHORT).show();

// }

if (stat == true){

setComplaint(contentString);

dialog.dismiss();

}

}

});

dialog.show();

dialog.getWindow().setAttributes(lp);

// dialog.getWindow().setBackground(Color.TRANSPARENT);

}

//send FeedBack

private void setComplaint(String content) {

rresult = false;

Map<String, String> params = new HashMap<>();

// {"cno":"1212","content":"2","nm":"fd","adr":"ekm","mob":"5655656565"}

params.put("cno", ""+ Utility.UID);

params.put("content", ""+content);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.SENDCOMPLAINTURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

//JSONObject jsonObj = new JSONObject(datafromserver);

// {"status":4}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("status")&&jsonObj.getInt("status")>0) {

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

// rresult = true;

getComplaints();

} else

Toast.makeText(getActivity(), "Complaint sending failed,Try again", Toast.LENGTH\_SHORT).show();

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

// return rresult;

}

}

11.2.5 Staff complaint Android

public class ComplaintsStaffFragment extends Fragment {

RecyclerView recyclerViewRV;

FloatingActionButton addComplaintBtn;

public ArrayList<DataComplaintStaff> dataComplaints;

public RecyclerViewAdapterStaffComplaints recyclerViewAdapterComplaints;

public boolean rresult = false;

@SuppressLint("RestrictedApi")

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.complaints\_fragment, container, false);

/\*\*Get String from Bundle which is passed from MainActivityNavDrawer.java\*\*/

String text = getArguments().getString("text", "");

init(view);

// addComplaintBtn.setVisibility(View.GONE);

addComplaintBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

addComplaint();

}

});

RecyclerView.LayoutManager mLayoutManager = new GridLayoutManager(getActivity(), 1, GridLayoutManager.VERTICAL,false);

recyclerViewRV.setNestedScrollingEnabled(true);

recyclerViewRV.setLayoutManager(mLayoutManager);

recyclerViewRV.setItemAnimator(new DefaultItemAnimator());

getComplaints();

return view;

}

private void init(View view) {

recyclerViewRV = view.findViewById(R.id.recyclerview\_rv);

addComplaintBtn = view.findViewById(R.id.addComplaintBtn);

}

public void getComplaints() {

Map<String, String> params = new HashMap<>();

params.put("uid", ""+ Utility.UID);

params.put("pincode", ""+ Utility.PINCODE);

// {"uid":"","pincode":""}

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.LISTALLSTAFFCOMPLAINTURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("result")) {

JSONArray login\_array = jsonObj.getJSONArray("result");

dataComplaints = new ArrayList<>();

DataComplaintStaff ca;

for(int k=0;k<login\_array.length();k++){

ca=new DataComplaintStaff();

//{"cmp\_id":"2","date":"","complaint":"fgfgfss","reply":"","status":"","emp\_id":"st100"}

actor = login\_array.getJSONObject(k);

ca.setCmp\_idStaff(actor.getString("cmp\_id"));

ca.setDateStaff(actor.getString("date"));

ca.setComplaintStaff(actor.getString("complaint"));

ca.setReplyStaff(actor.getString("reply"));

ca.setStatusStaff(actor.getString("status"));

ca.setEmp\_idStaff(actor.getString("emp\_id"));

Log.d("getmessagesList", jsonObj.toString());

dataComplaints.add(ca);

}

recyclerViewAdapterComplaints = new RecyclerViewAdapterStaffComplaints(getActivity(),dataComplaints, ComplaintsStaffFragment.this);

recyclerViewRV.setAdapter(recyclerViewAdapterComplaints);

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

public void addComplaint(){

final Dialog dialog = new Dialog(getActivity());

dialog.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

dialog.setContentView(R.layout.new\_complaintstaff\_layout);

dialog.setCancelable(true);

WindowManager.LayoutParams lp = new WindowManager.LayoutParams();

lp.copyFrom(dialog.getWindow().getAttributes());

lp.width = WindowManager.LayoutParams.MATCH\_PARENT;

lp.height = WindowManager.LayoutParams.WRAP\_CONTENT;

final AppCompatButton sendAppCompatButton = dialog.findViewById(R.id.postBtn);

final TextInputEditText contentTIET = dialog.findViewById(R.id.contentTIET);

// final TextInputEditText consumerNumberTIET = dialog.findViewById(R.id.consumerNumberTIET);

// final TextInputEditText nameTIET = dialog.findViewById(R.id.personTIET);

// final TextInputEditText addressTIET = dialog.findViewById(R.id.addressAddTIET);

// final TextInputEditText mobileTIET = dialog.findViewById(R.id.phoneTIET);

contentTIET.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

sendAppCompatButton.setEnabled(!charSequence.toString().trim().isEmpty());

}

@Override

public void afterTextChanged(Editable editable) {

}

});

sendAppCompatButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

boolean stat = true;

String contentString = contentTIET.getText().toString().trim();

// String cno = consumerNumberTIET.getText().toString().trim();

// String nm = nameTIET.getText().toString().trim();

// String adr = addressTIET.getText().toString().trim();

// String mob = mobileTIET.getText().toString().trim();

// if (nm.isEmpty()){

// stat = false;

// nameTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Name Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (cno.isEmpty()){

// stat = false;

// consumerNumberTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Consumer Number Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (adr.isEmpty()){

// stat = false;

// addressTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Address Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (mob.isEmpty()){

// stat = false;

// mobileTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Phone Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (mob.length()<10||mob.length()>10){

// stat = false;

// mobileTIET.setError("Valid Number Required");

// Toast.makeText(getActivity(), "Invalid Phone Number", Toast.LENGTH\_SHORT).show();

// }

if (stat == true){

setComplaint(contentString);

dialog.dismiss();

}

}

});

dialog.show();

dialog.getWindow().setAttributes(lp);

// dialog.getWindow().setBackground(Color.TRANSPARENT);

}

//send FeedBack

private void setComplaint(String content) {

rresult = false;

Map<String, String> params = new HashMap<>();

// {"cno":"1212","content":"2","nm":"fd","adr":"ekm","mob":"5655656565"}

params.put("uid", ""+ Utility.UID);

params.put("content", ""+content);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.ADDSTAFFCOMPLAINTURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

//JSONObject jsonObj = new JSONObject(datafromserver);

// {"status":4}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("status")&&jsonObj.getInt("status")>0) {

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

// rresult = true;

getComplaints();

} else

Toast.makeText(getActivity(), "Complaint sending failed,Try again", Toast.LENGTH\_SHORT).show();

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

// return rresult;

}

}

11.2.6 Consumer leave Android

public class LeaveFragment extends Fragment {

RecyclerView recyclerViewRV;

FloatingActionButton addComplaintBtn;

public ArrayList<DataLeave> dataComplaints;

public boolean rresult = false;

public int mYear;

public int mMonth;

public int mDay;

private RecyclerViewAdapterLeave recyclerViewAdapterLeave;

@SuppressLint("RestrictedApi")

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.complaints\_fragment, container, false);

/\*\*Get String from Bundle which is passed from MainActivityNavDrawer.java\*\*/

String text = getArguments().getString("text", "");

init(view);

// addComplaintBtn.setVisibility(View.GONE);

addComplaintBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

addComplaint();

}

});

RecyclerView.LayoutManager mLayoutManager = new GridLayoutManager(getActivity(), 1, GridLayoutManager.VERTICAL,false);

recyclerViewRV.setNestedScrollingEnabled(true);

recyclerViewRV.setLayoutManager(mLayoutManager);

recyclerViewRV.setItemAnimator(new DefaultItemAnimator());

getComplaints();

return view;

}

private void init(View view) {

recyclerViewRV = view.findViewById(R.id.recyclerview\_rv);

addComplaintBtn = view.findViewById(R.id.addComplaintBtn);

}

public void getComplaints() {

Map<String, String> params = new HashMap<>();

params.put("uid", ""+ Utility.UID);

// params.put("pincode", ""+ Utility.PINCODE);

// {"uid":"","pincode":""}

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.LISTLEAVEREQUESTURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("result")) {

JSONArray login\_array = jsonObj.getJSONArray("result");

dataComplaints = new ArrayList<>();

DataLeave ca;

for(int k=0;k<login\_array.length();k++){

ca=new DataLeave();

//{"leave\_id":"2","emp\_id":"st100","start\_date":"2019-5-5","reason":"fsdf","days":"3",

// "status":"Not Approved","request\_date":null}

actor = login\_array.getJSONObject(k);

ca.setLeave\_id(actor.getString("leave\_id"));

ca.setEmp\_id(actor.getString("emp\_id"));

ca.setStart\_date(actor.getString("start\_date"));

ca.setReason(actor.getString("reason"));

ca.setDays(actor.getString("days"));

ca.setStatus(actor.getString("status"));

ca.setRequest\_date(actor.getString("request\_date"));

Log.d("getmessagesList", jsonObj.toString());

dataComplaints.add(ca);

}

recyclerViewAdapterLeave = new RecyclerViewAdapterLeave(getActivity(),dataComplaints, LeaveFragment.this);

recyclerViewRV.setAdapter(recyclerViewAdapterLeave);

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

public void addComplaint(){

final Dialog dialog = new Dialog(getActivity());

dialog.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

dialog.setContentView(R.layout.new\_leave\_layout);

dialog.setCancelable(true);

WindowManager.LayoutParams lp = new WindowManager.LayoutParams();

lp.copyFrom(dialog.getWindow().getAttributes());

lp.width = WindowManager.LayoutParams.MATCH\_PARENT;

lp.height = WindowManager.LayoutParams.WRAP\_CONTENT;

final AppCompatButton sendAppCompatButton = dialog.findViewById(R.id.postBtn);

final TextInputEditText contentTIET = dialog.findViewById(R.id.contentTIET);

final TextInputEditText noDaysTIET = dialog.findViewById(R.id.numberofdaysTIET);

final TextView startDateTV = dialog.findViewById(R.id.startdate);

startDateTV.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

// Get Current Date

final Calendar c = Calendar.getInstance();

mYear = c.get(Calendar.YEAR);

mMonth = c.get(Calendar.MONTH);

mDay = c.get(Calendar.DAY\_OF\_MONTH);

DatePickerDialog datePickerDialog = new DatePickerDialog(getActivity(),

new DatePickerDialog.OnDateSetListener() {

@Override

public void onDateSet(DatePicker view, int year,

int monthOfYear, int dayOfMonth) {

startDateTV.setText(dayOfMonth + "-" + (monthOfYear + 1) + "-" + year);

}

}, mYear, mMonth, mDay);

datePickerDialog.show();

}

});

contentTIET.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

sendAppCompatButton.setEnabled(!charSequence.toString().trim().isEmpty());

}

@Override

public void afterTextChanged(Editable editable) {

}

});

sendAppCompatButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

boolean stat = true;

String contentString = contentTIET.getText().toString().trim();

String startDate = startDateTV.getText().toString().trim();

String noDays = noDaysTIET.getText().toString().trim();

if (startDate.isEmpty()|| startDate.equals("Start Date")){

stat = false;

// startDateTV.setText("Field Required");

Toast.makeText(getActivity(), "Date Field Empty or Invalid Date", Toast.LENGTH\_SHORT).show();

}else if (noDays.isEmpty()){

stat = false;

noDaysTIET.setError("Field Required");

Toast.makeText(getActivity(), "Number of Days Field Empty", Toast.LENGTH\_SHORT).show();

}

if (stat == true){

setComplaint(contentString,startDate,noDays);

dialog.dismiss();

}

}

});

dialog.show();

dialog.getWindow().setAttributes(lp);

// dialog.getWindow().setBackground(Color.TRANSPARENT);

}

//send FeedBack

private void setComplaint(String reason,String startDate,String noDays) {

rresult = false;

Map<String, String> params = new HashMap<>();

// uid, reason,start\_date, no\_days

params.put("uid", ""+ Utility.UID);

params.put("reason", ""+reason);

params.put("start\_date", ""+startDate);

params.put("no\_days", ""+noDays);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.ADDLEAVEREQUESTURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

//JSONObject jsonObj = new JSONObject(datafromserver);

// {"status":4}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("status")&&jsonObj.getInt("status")>0) {

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

// rresult = true;

Toast.makeText(getActivity(), "Leave Request sent succesfully", Toast.LENGTH\_SHORT).show();

getComplaints();

} else

Toast.makeText(getActivity(), "Leave Request failed,Try again", Toast.LENGTH\_SHORT).show();

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

// return rresult;

}

}

11.2.7 Consumer Profile Android

public class ProfileConsumerFragment extends Fragment {

TextView iconTV,empIdTV,namePTV,addressPTV,contactPTV,dobPTV,pincodePTV,areaPTV,

plotPTV,streetPTV,cityPTV,phonePTV,emailPTV,conStatusPTV

,editPasswordButton;

ImageView alertIV;

RelativeLayout relativeLayout ;

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.profileconsumer\_fragment, container, false);

/\*\*Get String from Bundle which is passed from MainActivityNavDrawer.java\*\*/

String text = getArguments().getString("text", "");

init(view);

getProfile();

if (Utility.USERROLE.equals("staff"))

relativeLayout.setVisibility(View.GONE);

else

relativeLayout.setVisibility(View.VISIBLE);

alertIV.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Utility.alertAdd(getActivity());

}

});

editPasswordButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

addPassword();

}

});

return view;

}

private void init(View view) {

iconTV = view.findViewById(R.id.iconTV);

empIdTV = view.findViewById(R.id.consumerNumberProfile);

namePTV = view.findViewById(R.id.nameProfile);

addressPTV = view.findViewById(R.id.addressProfile);

contactPTV = view.findViewById(R.id.contactProfile);

dobPTV = view.findViewById(R.id.dobPTV);

pincodePTV = view.findViewById(R.id.pincodePTV);

areaPTV = view.findViewById(R.id.areaPTV);

plotPTV = view.findViewById(R.id.plotPTV);

streetPTV = view.findViewById(R.id.streetPTV);

cityPTV = view.findViewById(R.id.cityPTV);

phonePTV = view.findViewById(R.id.phonePTV);

emailPTV = view.findViewById(R.id.emailProfile);

conStatusPTV = view.findViewById(R.id.conStatusPTV);

editPasswordButton = view.findViewById(R.id.editPassword);

alertIV = view.findViewById(R.id.alertiv);

relativeLayout = view.findViewById(R.id.rL);

//"http://i.imgur.com/Vth6CBz.gif"

Glide.with(getActivity())

.load(Utility.URLT)

.override(40,40)

.centerCrop()

.diskCacheStrategy(DiskCacheStrategy.ALL)

.placeholder(android.R.drawable.stat\_sys\_download)

.error(android.R.drawable.stat\_notify\_error)

.into(alertIV);

//iconTV,empIdTV,namePTV,addressPTV,contactPTV,dobPTV,pincodePTV,areaPTV,plotPTV,streetPTV,cityPTV,

// phonePTV,emailPTV,conStatusPTV;

}

private void getProfile() {

Map<String, String> params = new HashMap<>();

params.put("cno", ""+ Utility.UID);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.VIEWCONSUMERPROFILEURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("result")) {

JSONArray login\_array = jsonObj.getJSONArray("result");

actor = login\_array.getJSONObject(0);

//{"consumer\_no":"1212","cname":"Ajay","address":"Pala mangattukavala","dob":"","pincode":"686686",

// "area":"Kaloor","plot":"Signal","street":"3 A","city":"Kaloor","phone":"0484629584",

// "mobile":"9896857447","email":"ajay@gmail.com","con\_status":"1"}

String c = String.valueOf(actor.getString("cname").charAt(0)).toUpperCase();

iconTV.setText(""+c);

empIdTV.setText("Consumer Id : "+actor.getString("consumer\_no"));

namePTV.setText(actor.getString("cname"));

addressPTV.setText("Address : "+actor.getString("address"));

dobPTV.setText("DOB : "+actor.getString("dob"));

pincodePTV.setText("Pincode : "+actor.getString("pincode"));

areaPTV.setText("Area : "+actor.getString("area"));

plotPTV.setText("Plot : "+actor.getString("plot"));

streetPTV.setText("Street : "+actor.getString("street"));

cityPTV.setText("City : "+actor.getString("city"));

phonePTV.setText("Phone : "+actor.getString("phone"));

contactPTV.setText("Mobile : "+actor.getString("mobile"));

emailPTV.setText("email : "+actor.getString("email"));

conStatusPTV.setText("Connection Status : "+actor.getString("con\_status"));

Utility.lm = actor.getString("lm");

Log.d("getmessagesList", jsonObj.toString());

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

public void addPassword(){

final Dialog dialog = new Dialog(getActivity());

dialog.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

dialog.setContentView(R.layout.changepass\_layout);

dialog.setCancelable(true);

WindowManager.LayoutParams lp = new WindowManager.LayoutParams();

lp.copyFrom(dialog.getWindow().getAttributes());

lp.width = WindowManager.LayoutParams.MATCH\_PARENT;

lp.height = WindowManager.LayoutParams.WRAP\_CONTENT;

final AppCompatButton sendAppCompatButton = dialog.findViewById(R.id.postBtn);

final TextInputEditText oldTIET = dialog.findViewById(R.id.oldpasswordTIET);

final TextInputEditText newTIET = dialog.findViewById(R.id.newPasswordTIET);

final TextInputEditText confirmTIET = dialog.findViewById(R.id.confirmPassTIET);

confirmTIET.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) { sendAppCompatButton.setEnabled(!charSequence.toString().trim().isEmpty());

}

@Override

public void afterTextChanged(Editable editable) {

}

});

sendAppCompatButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

boolean stat = true;

String oldPass = oldTIET.getText().toString().trim();

String newPass = newTIET.getText().toString().trim();

String confirmPass = confirmTIET.getText().toString().trim();

if (oldPass.isEmpty()){

stat = false;

oldTIET.setError("Field Required");

Toast.makeText(getActivity(), "Old Password Field Empty", Toast.LENGTH\_SHORT).show();

}else if (newPass.isEmpty()){

stat = false;

newTIET.setError("Field Required");

Toast.makeText(getActivity(), "New Password Field Empty", Toast.LENGTH\_SHORT).show();

}else if (newPass.equals(confirmPass)) { }else{

stat = false;

confirmTIET.setError("Field Required");

Toast.makeText(getActivity(), "Password Mismatch", Toast.LENGTH\_SHORT).show();

}

if (stat == true){

setPass(newPass,oldPass);

dialog.dismiss();

}

}

});

dialog.show();

dialog.getWindow().setAttributes(lp);

// dialog.getWindow().setBackground(Color.TRANSPARENT);

}

//send FeedBack

private void setPass(String newPass,String oldPass) {

Map<String, String> params = new HashMap<>();

//{"uid":"1212","upwd":"12123","opwd":"1212"}

params.put("uid", ""+ Utility.UID);

params.put("upwd", ""+newPass);

params.put("opwd", ""+oldPass);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.CHANGEUSERPASSWORDURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

//JSONObject jsonObj = new JSONObject(datafromserver);

// {"status":4}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("status")&&jsonObj.getInt("status")>0) {

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

Toast.makeText(getActivity(), "Password Changed", Toast.LENGTH\_SHORT).show();

} else

Toast.makeText(getActivity(), "Password Changing failed", Toast.LENGTH\_SHORT).show();

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

// return rresult;

}

}

11.2.8 Tender Android

lapis.apps.lapislearning.kseb.recycler\_adapters.RecyclerViewAdapterComplaints;

import lapis.apps.lapislearning.kseb.recycler\_adapters.RecyclerViewAdapterTenders;

import retrofit2.Call;

public class TendersFragment extends Fragment {

RecyclerView recyclerViewRV;

ImageView alertIV;

RelativeLayout relativeLayout ;

public ArrayList<DataTenders> dataTenders;

public RecyclerViewAdapterTenders recyclerViewAdapterTenders;

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.tenders\_fragment, container, false);

/\*\*Get String from Bundle which is passed from MainActivityNavDrawer.java\*\*/

String text = getArguments().getString("text", "");

init(view);

RecyclerView.LayoutManager mLayoutManager = new GridLayoutManager(getActivity(), 1, GridLayoutManager.VERTICAL,false);

recyclerViewRV.setNestedScrollingEnabled(true);

recyclerViewRV.setLayoutManager(mLayoutManager);

recyclerViewRV.setItemAnimator(new DefaultItemAnimator());

getTenders();

if (Utility.USERROLE.equals("staff"))

relativeLayout.setVisibility(View.GONE);

else

relativeLayout.setVisibility(View.VISIBLE);

alertIV.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Utility.alertAdd(getActivity());

}

});

return view;

}

private void init(View view) {

recyclerViewRV = view.findViewById(R.id.recyclerview\_rv);

alertIV = view.findViewById(R.id.alertiv);

relativeLayout = view.findViewById(R.id.rL);

//"http://i.imgur.com/Vth6CBz.gif"

Glide.with(getActivity())

.load(Utility.URLT)

.override(40,40)

.centerCrop()

.diskCacheStrategy(DiskCacheStrategy.ALL)

.placeholder(android.R.drawable.stat\_sys\_download)

.error(android.R.drawable.stat\_notify\_error)

.into(alertIV);

}

public void getTenders() {

Map<String, String> params = new HashMap<>();

params.put("uid", ""+ Utility.UID);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.LISTALLTENDERURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("result")) {

JSONArray login\_array = jsonObj.getJSONArray("result");

dataTenders = new ArrayList<>();

DataTenders ca;

for(int k=0;k<login\_array.length();k++){

ca=new DataTenders();

//{"tender\_id":"2","title":"Street light","work\_description":"300 meter","date":"2019-3-15","status":"1"}

actor = login\_array.getJSONObject(k);

ca.setIdTender(actor.getString("tender\_id"));

ca.setTitleTender(actor.getString("title"));

ca.setWorkDescriptionTender(actor.getString("work\_description"));

ca.setDateTender(actor.getString("date"));

ca.setStatusTender(actor.getString("status"));

Log.d("getmessagesList", jsonObj.toString());

dataTenders.add(ca);

}

recyclerViewAdapterTenders = new RecyclerViewAdapterTenders(getActivity(),dataTenders,TendersFragment.this);

recyclerViewRV.setAdapter(recyclerViewAdapterTenders);

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

}

11.2.9 Worker Android

public class WorkFragment extends Fragment {

RecyclerView recyclerViewRV;

FloatingActionButton addComplaintBtn;

public ArrayList<DataComplaint> dataComplaints;

public RecyclerViewAdapterWorks recyclerViewAdapterComplaints;

public boolean rresult = false;

@SuppressLint("RestrictedApi")

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.complaints\_fragment, container, false);

/\*\*Get String from Bundle which is passed from MainActivityNavDrawer.java\*\*/

String text = getArguments().getString("text", "");

init(view);

addComplaintBtn.setVisibility(View.GONE);

addComplaintBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

addComplaint();

}

});

RecyclerView.LayoutManager mLayoutManager = new GridLayoutManager(getActivity(), 1, GridLayoutManager.VERTICAL,false);

recyclerViewRV.setNestedScrollingEnabled(true);

recyclerViewRV.setLayoutManager(mLayoutManager);

recyclerViewRV.setItemAnimator(new DefaultItemAnimator());

getComplaints();

return view;

}

private void init(View view) {

recyclerViewRV = view.findViewById(R.id.recyclerview\_rv);

addComplaintBtn = view.findViewById(R.id.addComplaintBtn);

}

public void getComplaints() {

Map<String, String> params = new HashMap<>();

params.put("uid", ""+ Utility.UID);

params.put("pincode", ""+ Utility.PINCODE);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.LISTALLWORKSURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("result")) {

JSONArray login\_array = jsonObj.getJSONArray("result");

dataComplaints = new ArrayList<>();

DataComplaint ca;

for(int k=0;k<login\_array.length();k++){

ca=new DataComplaint();

//{"comp\_no":"25","cons\_no":"1212","category":"GeneralCompalints","description":"Hkkjhh","person":"anoop",

// "adrs":"Gghhj","mobile":"9847131977","areas":"yty","dist":"","dates":"2017-10-19","reply":"8545m,l"}

actor = login\_array.getJSONObject(k);

ca.setComplaintNumber(actor.getString("comp\_no"));

ca.setConsumerNumber(actor.getString("consumer\_no"));

ca.setCategoryComplaint(actor.getString("category"));

ca.setDescriptionComplaint(actor.getString("description"));

ca.setPersonComplaint(actor.getString("person"));

ca.setAddressComplaint(actor.getString("adrs"));

ca.setMobileComplaint(actor.getString("mobile"));

ca.setAreaComplaint(actor.getString("areas"));

ca.setDistrictComplaint(actor.getString("pincode"));

ca.setDatecomplaint(actor.getString("dates"));

ca.setReplyComplaint(actor.getString("reply"));

Log.d("getmessagesList", jsonObj.toString());

dataComplaints.add(ca);

}

recyclerViewAdapterComplaints = new RecyclerViewAdapterWorks(getActivity(),dataComplaints,WorkFragment.this);

recyclerViewRV.setAdapter(recyclerViewAdapterComplaints);

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

}

public void addComplaint(){

final Dialog dialog = new Dialog(getActivity());

dialog.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

dialog.setContentView(R.layout.new\_complaint\_layout);

dialog.setCancelable(true);

WindowManager.LayoutParams lp = new WindowManager.LayoutParams();

lp.copyFrom(dialog.getWindow().getAttributes());

lp.width = WindowManager.LayoutParams.MATCH\_PARENT;

lp.height = WindowManager.LayoutParams.WRAP\_CONTENT;

final AppCompatButton sendAppCompatButton = dialog.findViewById(R.id.postBtn);

final TextInputEditText contentTIET = dialog.findViewById(R.id.contentTIET);

// final TextInputEditText consumerNumberTIET = dialog.findViewById(R.id.consumerNumberTIET);

// final TextInputEditText nameTIET = dialog.findViewById(R.id.personTIET);

// final TextInputEditText addressTIET = dialog.findViewById(R.id.addressAddTIET);

// final TextInputEditText mobileTIET = dialog.findViewById(R.id.phoneTIET);

contentTIET.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

sendAppCompatButton.setEnabled(!charSequence.toString().trim().isEmpty());

}

@Override

public void afterTextChanged(Editable editable) {

}

});

sendAppCompatButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

boolean stat = true;

String contentString = contentTIET.getText().toString().trim();

// String cno = consumerNumberTIET.getText().toString().trim();

// String nm = nameTIET.getText().toString().trim();

// String adr = addressTIET.getText().toString().trim();

// String mob = mobileTIET.getText().toString().trim();

// if (nm.isEmpty()){

// stat = false;

// nameTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Name Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (cno.isEmpty()){

// stat = false;

// consumerNumberTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Consumer Number Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (adr.isEmpty()){

// stat = false;

// addressTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Address Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (mob.isEmpty()){

// stat = false;

// mobileTIET.setError("Field Required");

// Toast.makeText(getActivity(), "Phone Field Empty", Toast.LENGTH\_SHORT).show();

// }else if (mob.length()<10||mob.length()>10){

// stat = false;

// mobileTIET.setError("Valid Number Required");

// Toast.makeText(getActivity(), "Invalid Phone Number", Toast.LENGTH\_SHORT).show();

// }

if (stat == true){

setComplaint(contentString);

dialog.dismiss();

}

}

});

dialog.show();

dialog.getWindow().setAttributes(lp);

// dialog.getWindow().setBackground(Color.TRANSPARENT);

}

//send FeedBack

private void setComplaint(String content) {

rresult = false;

Map<String, String> params = new HashMap<>();

// {"cno":"1212","content":"2","nm":"fd","adr":"ekm","mob":"5655656565"}

params.put("cno", ""+ Utility.UID);

params.put("content", ""+content);

Call<JsonObject> jsonObjectCall = new Retrofit\_Helper().getRetrofitBuilder().getfromServer(Utility.SENDCOMPLAINTURL,params);

jsonObjectCall.clone().enqueue(new ResponseHandler(getActivity(), new ResponseCallback() {

@Override

public void getResponse(int code, JsonObject jsonObject) {

JSONObject jsonObj = null;

try {

jsonObj = new JSONObject(jsonObject.toString());

} catch (JSONException e) {

e.printStackTrace();

}

//JSONObject jsonObj = new JSONObject(datafromserver);

// {"status":4}

if (jsonObj != null) {

JSONObject actor = null;

try {

if (jsonObj.has("status")&&jsonObj.getInt("status")>0) {

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

// rresult = true;

getComplaints();

} else

Toast.makeText(getActivity(), "Complaint sending failed,Try again", Toast.LENGTH\_SHORT).show();

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

} catch (Exception eee) {

eee.printStackTrace();

}

}

}

@Override

public void getFailure(Call<JsonObject> call, int code) {

if (code==1) {

Toast.makeText(getActivity(), "No Internet Connection", Toast.LENGTH\_SHORT).show();

} else {

try{

Toast.makeText(getActivity(), "Can't Connect with server", Toast.LENGTH\_SHORT).show();}catch(Exception e){}

}

if(ResponseHandler.progressDialog!=null)

ResponseHandler.progressDialog.dismiss();

}

}, jsonObjectCall,1));

// return rresult;

}

}

11.2.10 consumer registration

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using KSEBController;

using System.Data;

using System.Net;

public partial class Admin\_NewConsumerReg : System.Web.UI.Page

{

Controller cobj = new Controller();

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

DataTable dt = new DataTable();

string q = "SELECT max(consumer\_no) FROM `lapisco\_kseb`.`consumer`";

dt = cobj.controllerGetData(q);

string id = dt.Rows[0][0].ToString();

id = id.Remove(0, 2);

int sid = Convert.ToInt32(id);

sid = sid + 1;

txtConsNo.Text = "co" + sid;

}

}

protected void btnReg\_Click(object sender, EventArgs e)

{

string q = "insert into consumer values('" + txtConsNo.Text + "','" + txtcname.Text + "','" + txtaddr.Text + "','" + txtdob.Text + "','" + txtpin.Text + "','" + txtarea.Text + "','','','" + txtcity.Text + "'," + txtphone.Text + "," + txtmobile.Text + ",'" + txtemail.Text + "','1')";

cobj.controllerExeNonQuery(q);

msgDiv.Visible = true;

}

protected void btnNew\_Click(object sender, EventArgs e)

{

ClearTextBoxes(this.Controls);

}

void ClearTextBoxes(ControlCollection cc)

{

foreach (Control ctrl in cc)

{

if (ctrl is TextBox)

((TextBox)ctrl).Text = string.Empty;

ClearTextBoxes(ctrl.Controls);

}

}

}

11.2.10 consumer registration

public partial class Staff\_StaffLeaveRequest : System.Web.UI.Page

{

Controller cobj = new Controller();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void btnSend\_Click(object sender, EventArgs e)

{

string q = "insert into lapisco\_kseb.leave values(null,'"+Session["staff"].ToString()+"','"+txtReason.Text+"','"+txtaddress.Text+"','Pending')";

cobj.controllerExeNonQuery(q);

}

}

11.2.11 consumer bill

public partial class Consumer\_ViewandPayBill : System.Web.UI.Page

{

Controller cobj = new Controller();

protected void Page\_Load(object sender, EventArgs e)

{

if(!IsPostBack)

{

ViewBill();

}

}

DataTable ViewBill()

{

DataTable dt = new DataTable();

int cno = Convert.ToInt32(Session["consumer"].ToString());

string q1 = "select \* from `load` where `consumer\_no`=" + cno;

dt = cobj.controllerGetData(q1);

if (dt.Rows.Count > 0)

{

txtLoad.Text = dt.Rows[0]["total"].ToString();

}

txtcdate.Text = DateTime.Now.ToString("dd-MM-yyyy");

txtfine.Text = DateTime.Now.AddDays(30).ToString("dd-MM-yyyy");

txtLdate.Text = DateTime.Now.AddDays(29).ToString("yyyy-mm-dd");

int load = Convert.ToInt32(txtLoad.Text);

int amt = load \* 9;

txtamt.Text = Convert.ToString(amt);

return (dt);

}

protected void btnPaynow\_Click(object sender, EventArgs e)

{

lblAccount.Visible = true;

string cno = Session["consumer"].ToString();

string cdate = DateTime.Now.ToString("dd-MM-yyyy");

string fdate = DateTime.Now.AddDays(30).ToString("dd-MM-yyyy");

string ldate = DateTime.Now.AddDays(29).ToString("dd-MM-yyyy");

int load = Convert.ToInt32(txtamt.Text);

int amnt = load \* 9;

string q3 = "insert into `bill` values(null,'" + cno + "','" + cdate + "'," + amnt + ",'"+cdate+"','New','" + fdate + "','" + ldate + "','"+amnt.ToString()+"',1)";

cobj.controllerExeNonQuery(q3);

string q1 = "update `load` set total=0 where consumer\_no='" + cno + "'";

cobj.controllerExeNonQuery(q1);

}

protected void btncancel\_Click(object sender, EventArgs e)

{

Response.Redirect("ConsumerHome.aspx");

}

}

11.2.12 Staff assign location

public partial class Admin\_AssignEmployee : System.Web.UI.Page

{

Controller cobj = new Controller();

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

gridData();

}

}

void gridData()

{

DataTable dt = new DataTable();

string q = "select \* from lapisco\_kseb.employee ";

dt = cobj.controllerGetData(q);

DropDownList1.DataSource = dt;

DropDownList1.DataTextField = "emp\_id";

DropDownList1.DataBind();

dt = new DataTable();

q = "select distinct pincode from lapisco\_kseb.complaints ";

dt = cobj.controllerGetData(q);

DropDownList2.DataSource = dt;

DropDownList2.DataTextField = "pincode";

DropDownList2.DataBind();

}

protected void btnReg\_Click(object sender, EventArgs e)

{

string q = "insert into emp\_assign\_area values(null,'" + DropDownList1.SelectedItem.Text + "','" + DropDownList2.SelectedItem.Text + "')";

cobj.controllerExeNonQuery(q);

msgDiv.Visible = true;

}

}

11.2.13 consumer complaint

public partial class Consumer\_AddNewComplaint : System.Web.UI.Page

{

Controller cobj = new Controller();

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

string q = "select address,mobile,area,city from consumer where consumer\_no='" + Session["consumer"].ToString() + "' ";

DataTable dt = new DataTable();

dt = cobj.controllerGetData(q);

txtaddress.Text = dt.Rows[0][0].ToString();

txtMob.Text = dt.Rows[0][1].ToString();

txtarea.Text = dt.Rows[0][2].ToString();

txtdist.Text = dt.Rows[0][3].ToString();

}

}

protected void btnSend\_Click(object sender, EventArgs e)

{

//Session["consumer"]=3421058;

string cate = ddlCat.SelectedValue.ToString();

string q = "insert into complaints values(null,'" + Session["consumer"].ToString() + "','" + cate + "','" + txtDesc.Text + "','" + txtperson.Text + "','" + txtaddress.Text + "','" + txtMob.Text + "','" + txtarea.Text + "','"+txtdist.Text+"','"+DateTime.Now.ToShortDateString()+"','no reply')";

cobj.controllerExeNonQuery(q);

}

protected void btnCancel\_Click(object sender, EventArgs e)

{

Response.Redirect("ConsumerHome.aspx");

}

}

11.2.14 staff complaint

public partial class Staff\_SendComplaint : System.Web.UI.Page

{

Controller cobj = new Controller();

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

txtEmpId.Text = "Current Employee Id: "+Session["staff"].ToString();

txtDate.Text = DateTime.Now.ToShortDateString();

}

}

protected void btnSend\_Click(object sender, EventArgs e)

{

string q = "insert into emp\_complaint values(null,curdate(),'"+ txtComp.Text+"','No Reply','1','"+Session["staff"].ToString()+"' )";

cobj.controllerExeNonQuery(q);

Response.Redirect("ViewComplaint.aspx");

}

}