COMPLAINT

MANAGEMENT SYSTEM

*Project Report Submitted By*

# RUBIN SIBY

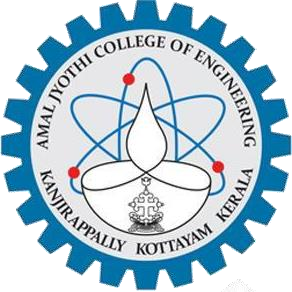
# Reg. No:AJC17MCA-I050

*In Partial fulfillment for the Award of the Degree Of*

**INTEGRATED MASTER OF COMPUTER APPLICATIONS**

# (INMCA)

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



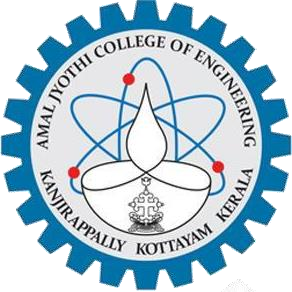
# AMAL JYOTHI COLLEGE OF ENGINEERING

# KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with ‘A’ grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

# 2021-2022

**DEPARTMENT OF COMPUTER APPLICATIONS AMAL JYOTHI COLLEGE OF ENGINEERING KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**COMPLAINT MANAGEMENT SYSTEM”** is the bonafide work of **RUBIN SIBY (Reg.No: AJC17MCA-I050)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

### Mr. Binumon Joseph Ms. Meera Rose Mathew Internal Guide Coordinator

**Fr. Dr. Rubin Thottupurathu Jose**

**Head of the Department**

I hereby declare that the project report **“COMPLAINT MANAGEMENT SYSTEM”** is a bonafided work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Integrated Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

**Date:20/12/2021 RUBIN SIBY**

**KANJIRAPPALLY Reg.No: AJC17MCA-I050**

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Z V Lakaparampil** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department

**Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms**. **Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide**, Mr. Binumon Joseph** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

RUBIN SIBY

**Complaint Management System** provides an online way of solving the problems faced by the public by saving time and eradicate corruption. The objective of the complaints management system is to make complaints easier to coordinate, monitor, track and resolve. Online Complaint Management is a management technique for assessing, analyzing and responding to customer complaints. Complaints management Application is used to record, resolve and respond to customer complaints, requests as well as facilitate any other feedback. The main purpose of this project is to help the public in knowing their place details and getting their problems solved in online without going to the officer regularly until the problem is solved. By this system the public can save his time and eradicate corruption in government offices. In the proposed system officer can set priority based on the complaints and generate reports and also user can raise a ticket if he is not satisfied with the complaint process.

.

**CONTENT**

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No** | **Topic** | **Page**  **No** |
| **1** | **INTRODUCTION** | **1** |
| **1.1** | **PROJECT OVERVIEW** | **2** |
| **1.2** | **PROJECT SPECIFICATION** | **2** |
| **2** | **SYSTEM STUDY** | **4** |
| **2.1** | **INTRODUCTION** | **5** |
| **2.2** | **EXISTING SYSTEM** | **6** |
| **2.3** | **DRAWBACKS OF EXISTING**  **SYSTEM** | **6** |
| **2.4** | **PROPOSED SYSTEM** | **6** |
| **2.5** | **ADVANTAGES OF PROPOSED**  **SYSTEM** | **7** |
| **3** | **REQUIREMENT ANALYSIS** | **8** |
| **3.1** | **FEASIBILITY STUDY** | **9** |
| **3.1.1** | **ECONOMICAL FEASIBILITY** | **9** |
| **3.1.2** | **TECHNICAL FEASIBILITY** | **10** |
| **3.1.3** | **BEHAVIORAL FEASIBILITY** | **10** |
| **3.2** | **SYSTEM SPECIFICATION** | **11** |
| **3.2.1** | **HARDWARE SPECIFICATION** | **11** |
| **3.2.2** | **SOFTWARE SPECIFICATION** | **11** |
| **3.3** | **SOFTWARE DESCRIPTION** | **11** |
| **3.3.1** | **REACT JS** | **11** |
| **3.3.2** | **NODE JS** | **11** |
| **4** | **SYSTEM DESIGN** | **13** |
| **4.1** | **INTRODUCTION** | **14** |
| **4.2** | **UML DIAGRAM** | **14** |
| **4.2.1** | **USE CASE DIAGRAM** | **15** |
| **4.2.2** | **SEQUENCE DIAGRAM** | **18** |
| **4.2.3** | **ACTIVITY DIAGRAM** | **20** |
| **4.2.4** | **COMPONENT DIAGRAM** | **24** |
| **4.2.5** | **CLASS DIAGRAM** | **26** |
| **4.3** | **USER INTERFACE DESIGN** | **28** |
| **4.4** | **DATA BASE DESIGN** | **33** |
| **5** | **SYSTEM TESTING** | **38** |
| **5.1** | **INTRODUCTION** | **39** |
| **5.2** | **TEST PLAN** | **40** |
| **5.2.1** | **UNIT TESTING** | **40** |
| **5.2.2** | **INTEGRATION TESTING** | **41** |
| **5.2.3** | **VALIDATION TESTING** | **41** |
| **5.2.4** | **USER ACCEPTANCE TESTING** | **42** |
| **6** | **IMPLEMENTATION** | **43** |
| **6.1** | **INTRODUCTION** | **44** |
| **6.2** | **IMPLEMENTATION**  **PROCEDURE** | **44** |
| **6.2.1** | **USER TRAINING** | **45** |
| **6.2.2** | **TRAINING ON APPLICATION**  **SOFTWARE** | **45** |
| **6.2.3** | **SYSTEM MAINTENANCE** | **45** |
| **7** | **CONCLUSION & FUTURE SCOPE** | **46** |
| **7.1** | **CONCLUSION** | **47** |
| **7.2** | **FUTURE SCOPE** | **47** |
| **8** | **BIBLIOGRAPHY** | **48** |
| **9** | **APPENDIX** | **50** |
| **9.1** | **SAMPLE CODE** | **51** |
| **9.2** | **SCREEN SHOTS** | **65** |

**List of Abbreviation**

IDE - Integrated Development Environment HTML - Hyper Text Markup Language.

CSS - Cascading Style Sheet

UML - Unified Modeling Language

# CHAPTER 1 INTRODUCTION

### PROJECT OVERVIEW

**“COMPLAINT MANAGEMENT SYSTEM”** is a web application which provides

an online way of solving the problems faced by the public by saving time and eradicate corruption. The objective of the complaints management system is to make complaints easier to coordinate, monitor, track and resolve. Online Complaint Management is a management technique for assessing, analyzing and responding to customer complaints. Complaints management Application is used to record, resolve and respond to customer complaints, requests as well as facilitate any other feedback. The main purpose of this project is to help the public in knowing their place details and getting their problems solved in online without going to the officer regularly until the problem is solved. By this system the public can save his time and eradicate corruption in government offices. In the proposed system the citizen need not go to the government office for getting his problem solved.

### PROJECT SPECIFICATION

Proposed system will change the way of maintaining the society complaints will also prove to be beneficial, improve efficiency and save us time. By using this application people can register their complaints in easy and proper format. Users gets notification on each step of proceeding the complaint and it provides a tracking facility. Users can raise ticket to alert the concerned higher authority in case of escalation. They can also provide feedback about their complaints progress weather they are satisfied or not. Inform the authorized person on receipt of complaint/application etc.

The system includes 4 modules. They are:

### Admin Module

Admin must have a login into this system. Admin can add or update officer. Admin can manage users and main officers. Admin can add officers to the departments and can view reports of the complaint status.

### User Module

User can register and they can post complaints with full details. User can check complaint status using the Complaint Id and view the detailed status report. Users can raise a complaint ticket if the posted complaint is not resolved or not satisfied to higher authority

### Officer Module

Officer can login and they can see all the Reported complaints. Officer can ask clarification on the ticket raised complaints to the assigned officer. Officer can verify the complaint and generate report for the complaint. Officer can contact the user using chat function. Officer can update the status and give feedback to the user.

### Sub-Officer Module

### Sub officers have a login and they can view all the complaints based on their location. They have main three functionality. First, they can take action based on the complaints. Second, they can ask more details to the complainer using clarification module and last, they can report the complaint to the higher authority by setting the priority and details.

# CHAPTER 2 SYSTEM STUDY

### INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute’s detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

### EXISTING SYSTEM

Existing system is not a fully automated system. In the existing system it provides only the user to post complaints, there is no tracking facility. Users wants to check the system always to know about the status. It’s does not provide a notification facility. It is a time-consuming process and there is no feedback facility.

### DRAWBACKS OF EXISTING SYSTEM

* + - Less convenient in managing complaints approval, less transparency.
    - No tracking facility.
    - Users wants to check the system always to know about the status.
    - Human effort is needed.
    - It is a time-consuming process and there is no feedback facility

### PROPOSED SYSTEM

Proposed system will change the way of maintaining the society complaints will also prove to be beneficial, improve efficiency and save us time. By using this application people can register their complaints in easy and proper format. Users gets notification on each step of proceeding the complaint and it provides a tracking facility. Users can raise ticket to alert the concerned higher authority in case of escalation. They can also provide feedback about their complaints progress weather they are satisfied or not. Inform the authorized person on receipt of complaint/application

### ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features:

### Better security: -

For data to remain secure measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction. The system security problem can be divided into four related issues: security, integrity, privacy and confidentiality. Username and password requirement to sign in ensures security. It will also provide data security as we are using the secured databases for maintaining the documents.

### Ensure data accuracy: -

The proposed system eliminates the manual errors while entering the details of the users during the registration.

### Better service: -

The system will avoid the burden of hard copy storage. We can also conserve the time and human resources for doing the same task. The data can be maintained for longer period with no loss of data.

# CHAPTER 3 REQUIREMENT ANALYSIS

## FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features: -

### Economic Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

* + - * The costs conduct a full system investigation.
      * The cost of the hardware and software.
      * The benefits in the form of reduced costs or fewer costly errors.

The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

The cost of project, COMPLAINT MANAGEMENT SYSTEM was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open-source software.

### Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

* + - * Does the existing technology sufficient for the suggested one?
      * Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project requires High Resolution Scanning device and utilizes Cryptographic techniques. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So, there are minimal constraints involved with this project. The system has been developed using REACT JS in front end and NODE JS in server in back end, the project is technically feasible for development. The System used was also of good performance of Processor Intel i7 core; RAM 12GB and, Hard disk 1TB

### Behavioral Feasibility

The proposed system includes the following questions:

* + - * Is there sufficient support for the users?
      * Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

COMPLAINT MANAGEMENT SYSTEM, GUI is simple so that

users can easily use it. COMPLAINT MANAGEMENT SYSTEM is simple enough so that no training is needed.

## SYSTEM SPECIFICATION

### Hardware Specification

Processor - Intel core i3 RAM - 4 GB

Hard disk - 1 TB

### Software Specification

Front End - REACT

Backend - NODE

Client on PC - Windows 7 and above.

Technologies used - REACT JS, NODE JS, CSS, MONGO DB, JS, GIT

## SOFTWARE DESCRIPTION

### REACT JS

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, react is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality

### NODE JS

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

### Asynchronous and Event Driven.

### All APIs of Node.js library is asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call

### Very Fast.

Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

### Single Threaded but Highly Scalable

Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

### No Buffering.

Node.js applications never buffer any data. These applications simply output the data in chunks

# CHAPTER 4 SYSTEM DESIGN

* 1. **INTRODUCTION**

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user-oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

## UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for **Unified Modeling Language**. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. After

some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

* Class diagram
* Object diagram
* Use case diagram
* Sequence diagram
* Collaboration diagram
* Activity diagram
* State chart diagram
* Deployment diagram
* Component diagram

## USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation forth modeling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service- oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

* + - * The boundary, which defines the system of interest in relation to the world around it.
      * The actors, usually individuals involved with the system defined according to their

roles.

* + - * The use cases, which are the specific roles are played by the actors within and around the system.
      * The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

* + - * The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
      * Give a suitable name for actors.
      * Show relationships and dependencies clearly in the diagram.
      * Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
      * Use notes whenever required to clarify some important points.

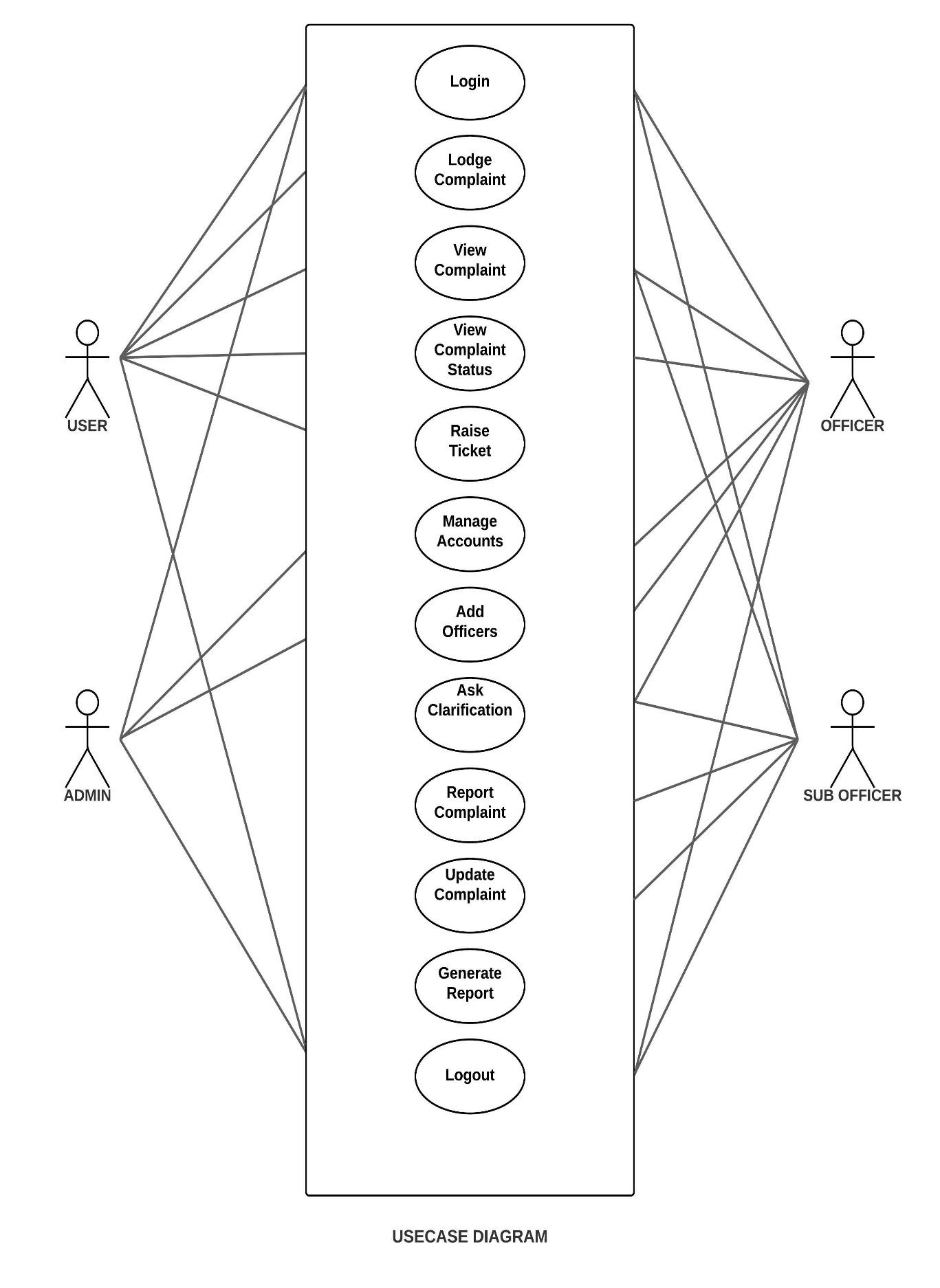


Fig 1 : Use case diagram for Complaint Management System

## SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order

i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

### Sequence Diagram Notations –

1. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.
2. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically, each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.
3. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

* + Synchronous messages
  + Asynchronous Messages
  + Create message
  + Delete Message
  + Self-Message
  + Reply Message
  + Found Message
  + Lost Message

1. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

### Uses of sequence diagrams –

* Used to model and visualize the logic behind a sophisticated function, operation or procedure.
* They are also used to show details of UML use case diagrams.
* Used to understand the detailed functionality of current or future systems.
* Visualize how messages and tasks move between objects or components in a system.

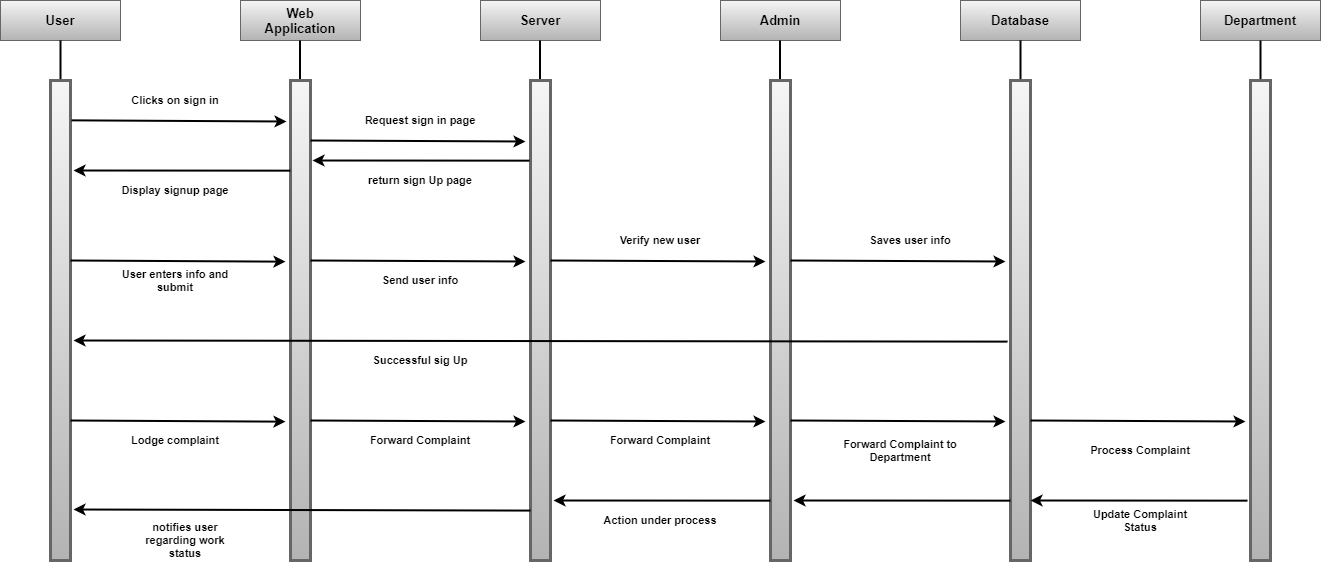
****

Fig 1 : Sequence diagram for Complaint Management System

## ACTIVITY DIAGRAM

**Diagrams** to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.

UML models basically three types of diagrams, namely, structure diagrams, interaction diagrams, and behavior diagrams. An activity diagram is a **behavioral diagram** i.e. it depicts the behavior of a system.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system.

An activity diagram is very **similar to a flowchart**. So let us understand if an activity diagrams or flowcharts are any different:

Activity diagrams can be regarded as a form of a structured flowchart combined with a traditional data flow diagram. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

While in UML 1.x, activity diagrams were a specialized form of state diagrams, in UML 2.x, the activity diagrams were renormalized to be based on Petri net-like semantics, increasing the scope of situations that can be modeled using activity diagrams. These changes cause many UML 1.x activity diagrams to be interpreted differently in UML 2.x.

UML activity diagrams in version 2.x can be used in various domains, e.g., in design of embedded systems. It is possible to verify such a specification using model checking technique.

## 

Fig 1: User Activity diagram for Complaint Management System

## 

Fig 2: Officer Activity diagram for Complaint Management System

## 

Fig 3: Sub-Officer Activity diagram for Complaint Management System

## 

Fig 4: Admin Activity diagram for Complaint Management System

## COMPONENT DIAGRAM

## Component diagrams are integral to building your software system. Drawn out with UML diagramming software, they help your team understand the structure of existing systems and then build new ones. Keep reading to learn more about component diagrams.

## The purpose of a component diagram is to show the relationship between different components in a system. For the purpose of UML 2.0, the term "component" refers to a module of classes that represent independent systems or subsystems with the ability to interface with the rest of the system.

## There exists a whole development approach that revolves around components: component-based development (CBD). In this approach, component diagrams allow the planner to identify the different components so the whole system does what it's supposed to do.

## More commonly, in an OO programming approach, the component diagram allows a senior developer to group classes together based on common purpose so that the developer and others can look at a software development project at a high level

## Benefits of component diagrams

## Though component diagrams may seem complex at first glance, they are invaluable when it comes to building your system. Component diagrams can help your team:

## Imagine the system’s physical structure.

## Pay attention to the system’s components and how they relate.

## Emphasize the service behavior as it relates to the interface

Fig 1: Component diagram for Complaint Management System

## 

## CLASS DIAGRAM

## Class diagrams are one of the most useful types of diagrams in UML as they clearly map out the structure of a particular system by modeling its classes, attributes, operations, and relationships between objects. With our UML diagramming software, creating these diagrams is not as overwhelming as it might appear. This guide will show you how to understand, plan, and create your own class diagrams.

## The Unified Modeling Language (UML) can help you model systems in various ways. One of the more popular types in UML is the class diagram. Popular among software engineers to document software architecture, class diagrams are a type of structure diagram because they describe what must be present in the system being modeled. No matter your level of familiarity with UML or class diagrams, our UML software is designed to be simple and easy to use.

## UML was set up as a standardized model to describe an object-oriented programming approach. Since classes are the building block of objects, class diagrams are the building blocks of UML. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects.

## The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.

## The UML shape library in Lucid chart can help you create nearly any custom class diagram using our UML diagram tool.

## Benefits of class diagrams

## Class diagrams offer a number of benefits for any organization. Use UML class diagrams to:

## Illustrate data models for information systems, no matter how simple or complex.

## Better understand the general overview of the schematics of an application.

## Visually express any specific needs of a system and disseminate that information throughout the business.

## Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.

## Provide an implementation-independent description of types used in a system that are later passed between its components.

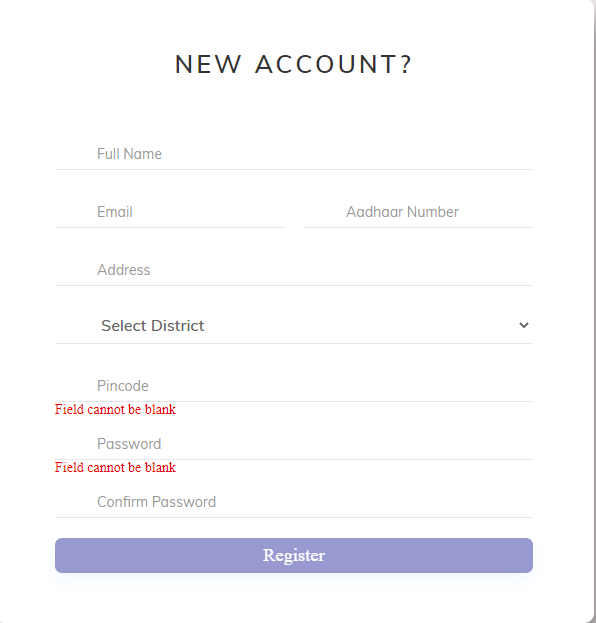
## 

## Fig 1: Class diagram for Complaint Management System

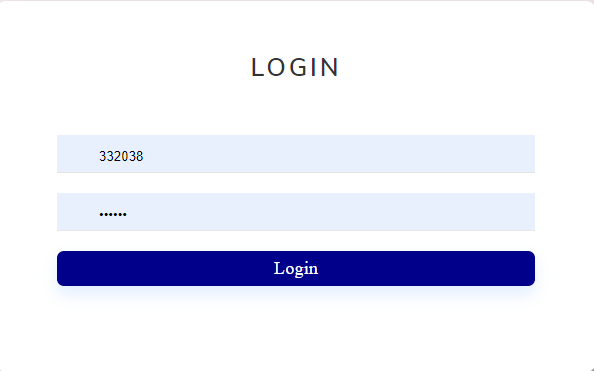
### USER INTERFACE DESIGN

**4.3.1-INPUT DESIGN**

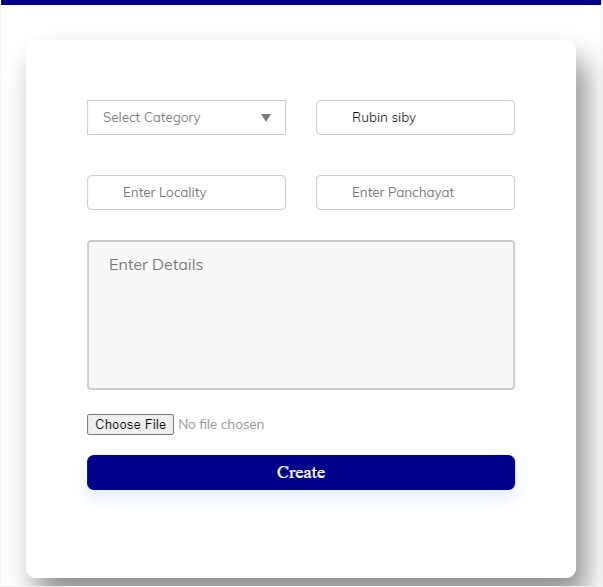
Form Name : User Registration



Form Name : User Login

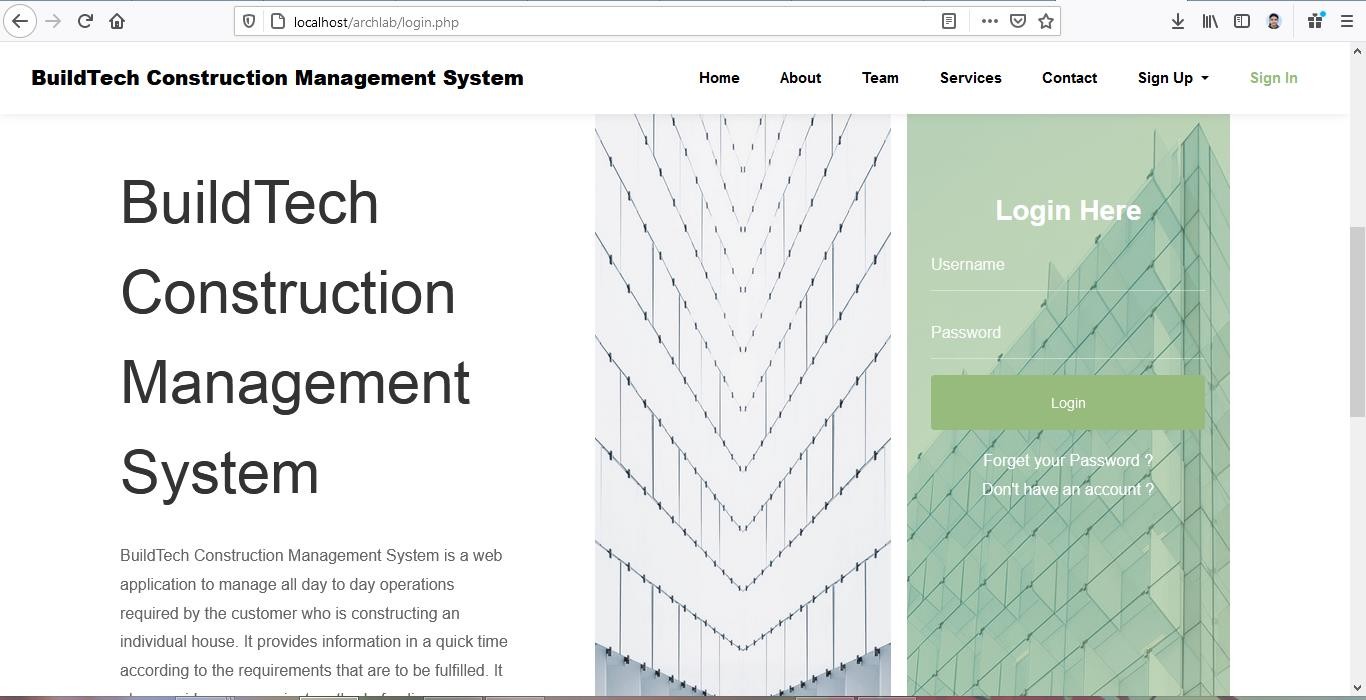
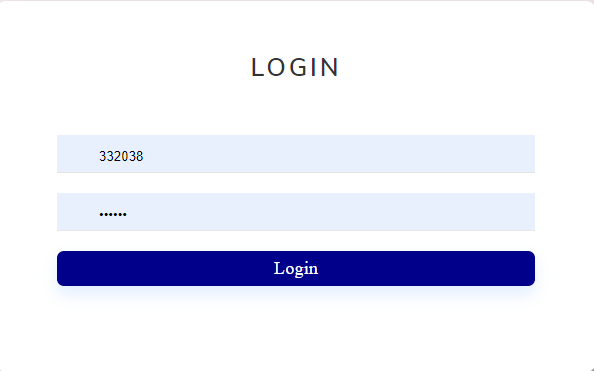


Form Name : User Complaint form

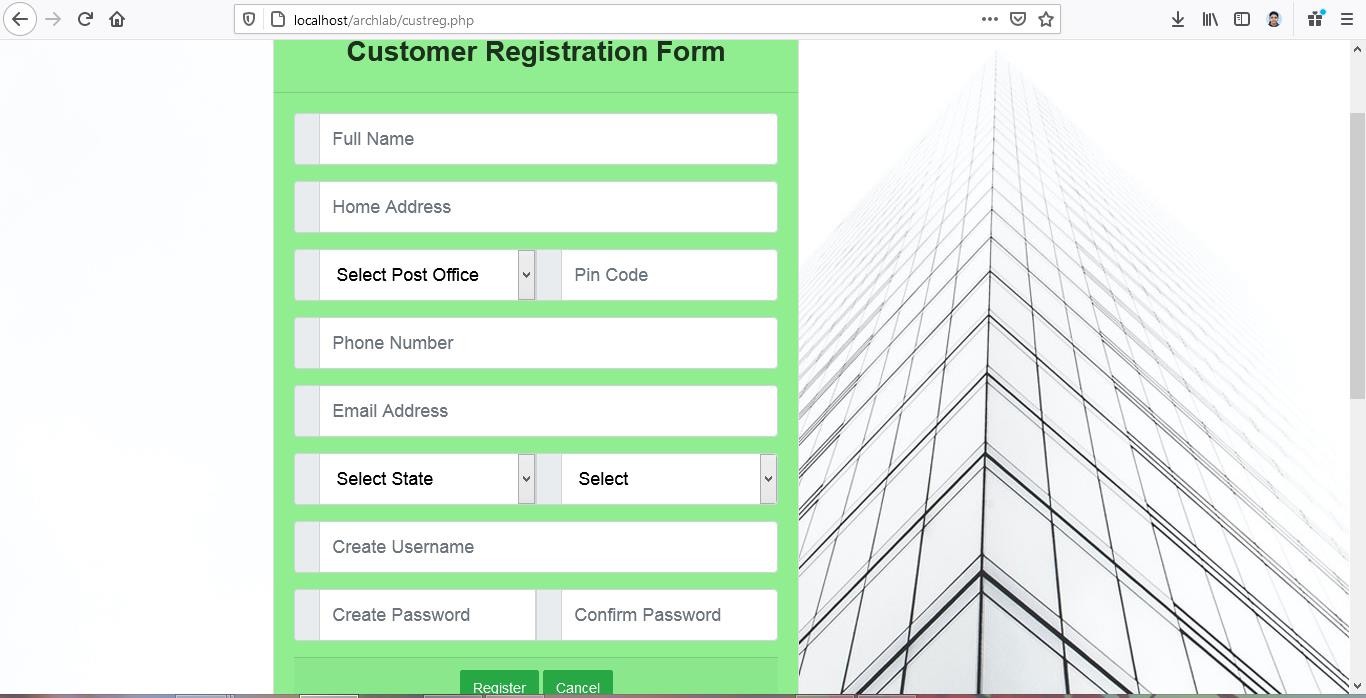
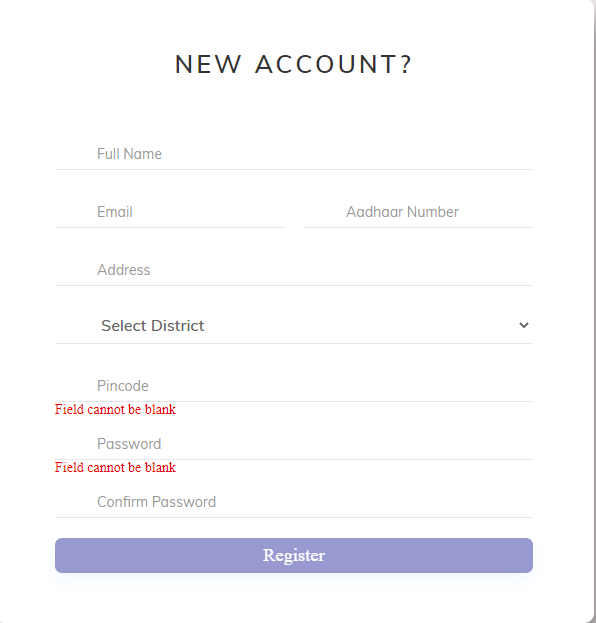


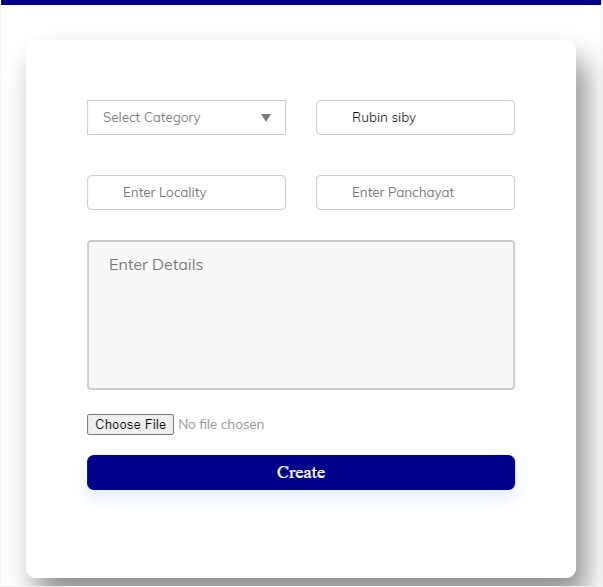
### 4.3.2 OUTPUT DESIGN

**User Login**



**User Registration**



**User Complaint Registration form**

## DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two-level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

* + - Data Integrity
    - Data independence

### Non-relational data and NoSQL

A non-relational database is a database that does not use the tabular schema of rows and columns found in most traditional database systems. Instead, non-relational databases use a storage model that is optimized for the specific requirements of the type of data being stored. For example, data may be stored as simple key/value pairs, as JSON documents, or as a graph consisting of edges and vertices.

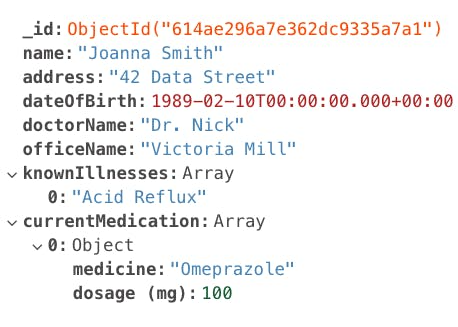
The term NoSQL refers to data stores that do not use SQL for queries. Instead, the data stores use other programming languages and constructs to query the data. In practice,

"NoSQL" means "non-relational database," even though many of these databases do support SQL-compatible queries. However, the underlying query execution strategy is usually very different from the way a traditional RDBMS would execute the same SQL query.

Non-relational databases are often used when large quantities of complex and diverse data need to be organized. For example, a large store might have a database in which each customer has their own document containing all of their information, from name

and address to order history and credit card information. Despite their differing formats, each of these pieces of information can be stored in the same document.

Non-relational databases often perform faster because a query doesn’t have to view several tables in order to deliver an answer, as relational datasets often do. Non-relational databases are therefore ideal for storing data that may be changed frequently or for applications that handle many different kinds of data. They can support rapidly developing applications requiring a dynamic database able to change quickly and to accommodate large amounts of complex, unstructured data.



**The benefits of a non-relational database**

Today’s applications collect and store increasingly vast quantities of ever-more complex customer and user data. The benefits of this data to businesses, of course, lie in their potential for analysis. Using a non-relational database can unlock patterns and value even within masses of variegated data.

There are several advantages to using non-relational databases, including:

* Massive dataset organization

In the age of Big Data, non-relational databases can not only store massive quantities of information, but they can also query these datasets with ease. Scale and speed are crucial advantages of non-relational databases.

* Flexible database expansion

Data is not static. As more information is collected, a non-relational database can absorb these new data points, enriching the existing database with new levels of granular value even if they don’t fit the data types of previously existing information.

* Multiple data structures

The data now collected from users takes on a myriad of forms, from numbers and strings, to photo and video content, to message histories. A database needs the ability to store these various information formats, understand relationships between them, and perform detailed queries. No matter what format your information is in, non-relational databases can collate different information types together in the same document.

* Built for the cloud

A non-relational database can be massive. And as they can, in some cases, grow exponentially, they need a hosting environment that can grow and expand with them. The cloud’s inherent scalability makes it an ideal home for non-relational databases.

**Non-relational databases and application development**

Applications must be able to query data efficiently and deliver results almost instantly. Non-relational databases are a natural choice for this kind of environment. They offer both security and agility, allowing for rapid development of applications in an agile environment. Easier and less complex to manage than relational databases, they can also yield lower data management costs while providing superior performance and speed.

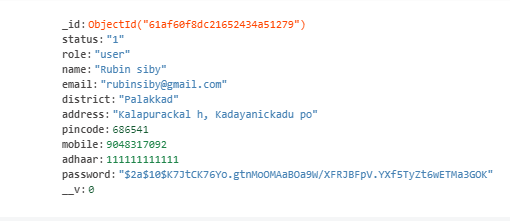
Naturals for agile development, non-relational databases can accommodate the complexity of data inputs more efficiently than structured databases. In an age of increasing data complexity, non-relational databases provide the flexibility in database design that has become increasingly indispensable. Especially when paired with the cloud, non-relational databases lift the limits on your data collection, organization, and analysis, allowing you to get the most out of your data.

### 4.5 TABLE DESIGN

**Table No 01**

### Table Name : Users

**Table Description: To store user Login information**

****

### Table No 02

**Table Name : Complaints**

**Table Description: To store user Complaints**

****

### Table No 03

**Table Name : Departments**

### Table Description: To store Department Details



**Table No 04**

### Table Name : Officer

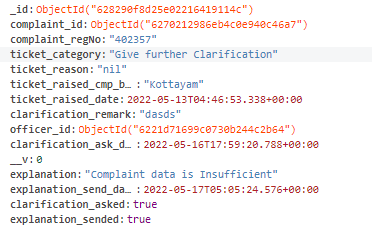
### Table Description: To store officer Details

****

**Table No 05**

### Table Name : ticket\_datas

### Table Description: To store Ticket raised Complaints



**Table No 06**

### Table Name : bank\_datas

### Table Description: To store Bank Details

### 

**Table No 07**

### Table Name : complaint\_datas

### Table Description: To store Complaint datas

### 

# CHAPTER 5 SYSTEM TESTING

## INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the term’s verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is complete without testing, as its vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

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

* + 1. A good test case is one that has high possibility of finding an undiscovered error.
    2. A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appears to be working according to the specification, that performance requirement appears to have been met.

There are three ways to test program.

* + 1. For correctness
    2. For implementation efficiency
    3. For computational complexity

Test for correctness is supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

## TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers are always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

* Unit testing
* Integration Testing
* Data validation Testing
* Output Testing

### Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code were removed and ensured that all modules are working, and gives the expected result.

### Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover, differences in program structures were removed and a unique program structure was evolved.

### Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

### Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

* + - 1. Input Screen Designs,
      2. Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

# CHAPTER 6 IMPLEMENTATION

## 6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

The implementation state involves the following tasks:

Careful planning.

Investigation of system and constraints. Design of methods to achieve the changeover.

## 6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to

ensure that the resistance does not build up, as one has to make sure that:

The active user must be aware of the benefits of using the new system. Their confidence in the software is built up.

Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won’t take place.

### 6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

### 6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy

### System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

# CHAPTER 7 CONCLUSION AND FUTURE SCOPE

## 7.1 CONCLUSION

The current system working technology is old fashioned and not user friendly. The proposed system introduces facility for users to track their complaint status. Provides lots of advantages like raise ticket, view officer details, feedback functionality, create complaint report, complaint status and enhanced user interface many more.

### 7.2 FUTURE SCOPE

7.2.1 Officer can able to apply leave

7.2.2 Users can able to add live location

# CHAPTER 8 BIBLIOGRAPHY

### REFERENCES:

8.1.1 PankajJalote, “So*ftware engineering*: a precise approach”, 2006.

8.1.2 James lee and Brent ware Addison, “Open-source web development with LAMP”, 2003

8.1.3 IEEE Std 1016 Recommended Practice for Software Design Descriptions.

### WEBSITE

* + 1. [www.jquery.com](http://www.jquery.com/)
    2. <https://www.lovelycoding.org/construction-work-management-system/>
    3. [https://www.slideshare.net/chiragbarasiya/construction-management-system-final-year-](https://www.slideshare.net/chiragbarasiya/construction-management-system-final-year-report) [report](https://www.slideshare.net/chiragbarasiya/construction-management-system-final-year-report)
    4. https://app.diagrams.net
    5. <http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf>
    6. [www.agilemodeling.com/artifacts/useCaseDiagram.html](http://www.agilemodeling.com/artifacts/useCaseDiagram.html)

# CHAPTER 9 APPENDIX

* 1. **Sample Code User**

**Post Complaint.jsx**

import React, { useState, useEffect, useContext } from "react";

import "./ComplaintForm.css";

import axios from "../../../../axios";

import { Close } from "@material-ui/icons";

import { Link, useLocation } from "react-router-dom";

import { userContext } from "../../../../AppContext";

import Modal from "react-modal";

function ComplaintForm() {

const [departmentName, setDepartmentName] = useState(null);

const [category, setCategory] = useState();

const [name, setName] = useState();

const [area, setArea] = useState();

const [panchayat, setPanchayat] = useState();

const [description, setDescription] = useState();

// const [regID, setRegId] = useState();

const location = useLocation();

const { userdata } = useContext(userContext);

console.log(userdata);

useEffect(() => {

const { department } = location;

setDepartmentName(department);

}, [location]);

const customStyles = {

content: {

top: "50%",

left: "50%",

right: "auto",

bottom: "auto",

width: "600px",

transform: "translate(-50%, -50%)",

},

};

const [modalIsOpen, setIsOpen] = useState(false);

function openModal() {

setIsOpen(true);

}

function closeModal() {

setIsOpen(false);

}

const handleComplaint = async () => {

const data = {

userid: userdata.userid,

department: departmentName,

category: category,

name: name,

area: area,

panchayat: panchayat,

description: description,

};

const res = await axios.post("/postcomplaint", data);

if (res.status === 200) {

// await setRegId(res.data.regno);

openModal();

} else {

alert("Some error occured");

}

};

return (

<div className="complaint-form">

<div class="comp\_form\_wrapper">

<div class="form\_container">

<div class="row clearfix">

<div class="">

<form>

{" "}

<div class="row clearfix">

<div class="col\_half">

<div class="input\_field select\_option">

<select onChange={(e) => setCategory(e.target.value)}>

<option value=" " selected>

Select Category

</option>

<option>Payment Related</option>

<option>Employ Related</option>

<option>Otheres</option>

</select>

<div class="select\_arrow"></div>

</div>

</div>

<div class="col\_half">

<div class="input\_field">

<input

type="text"

name="name"

placeholder="Full name"

value={userdata.username}

onChange={(e) => setName(e.target.value)}

required

/>

</div>

</div>

</div>

<div class="row clearfix">

<div class="col\_half">

<div class="input\_field">

{" "}

<input

type="text"

placeholder="Enter Locality"

onChange={(e) => setArea(e.target.value)}

required

/>

</div>

</div>

<div class="col\_half">

<div class="input\_field">

{" "}

<input

type="text"

placeholder="Enter Panchayat"

onChange={(e) => setPanchayat(e.target.value)}

required

/>

</div>

</div>

</div>

<div class="input\_field">

{" "}

<textarea

placeholder="Enter Details"

onChange={(e) => setDescription(e.target.value)}

></textarea>

</div>

<div class="input\_field">

{" "}

<input

type="file"

className="complaint-proof"

name=""

id=""

/>

</div>

<input

class="offcer-btn"

type="button"

value="Create"

onClick={handleComplaint}

/>

<div className="officerErr err-hidden">

Officer Already Created

</div>

</form>

</div>

</div>

</div>

</div>

<Modal

isOpen={modalIsOpen}

onRequestClose={closeModal}

style={customStyles}

contentLabel="Example Modal"

>

<div className="dep-header">

<div class="modal-close-btn">

<Close className="closeBtn" onClick={closeModal} />

</div>

</div>

<form>

<div className="title">

<h1>Complaint Registered Successfully</h1>

</div>

<div className="body">

<div className="username">

<p>Registration ID has been sended to your mobile number </p>

</div>

</div>

<div className="footer">

<Link to="/dashboard">

<button>Continue</button>

</Link>

</div>

</form>

</Modal>

</div>

);

}

export default ComplaintForm;

**Post Complaint Back-end**

router.post("/postcomplaint", async (req, res) => {

userOperation.postComplaint(req.body).then((response) => {

if (response) {

res

.status(200)

.json({

message: "complaint registered successfully",

regno: response.registrationNo,

});

} else {

res.status(401).json({ message: "some error occured" });

}

});

})

postComplaint: (data) => {

try {

return new Promise(async (resolve, reject) => {

let user = await userSchema.find({

\_id: data.userid,

});

let usermobile = `+${91}` + user[0].mobile;

let regno = randomstring.generate({

length: 6,

charset: "numeric",

});

client.messages

.create({

body:

"Your complaint is registered successfully and you register number is " +

regno,

from: "+19704382955",

to: usermobile,

})

.then(async (message) => {

if (message) {

console.log(message.sid);

var postcomplaint = new complaintSchema({

registrationNo: regno,

department: data.department,

userId: data.userid,

name: user[0].name,

description: data.description,

complaint\_type: data.category,

panchayat: data.panchayat,

area: data.area,

});

var complaint = await postcomplaint.save();

if (complaint) {

console.log(complaint);

resolve(complaint);

}

} else {

console.log("network issue");

}

});

}).catch((err) => console.log("error", err));

} catch (error) {

console.log(error);

}

},

**Officer**

**UpdateComplaint**

## import React, { useState } from "react";

## import "./OfficerComplaintList.css";

## function OfficerComplaintList({ complaint }) {

## return (

## <div className="complaintDetails">

## <div className="complaintTitle">Complaints</div>

## <div class="datas">

## <div class="data">

## <table>

## <tr>

## <th>Complaint Id</th>

## <th> Description</th>

## <th>User Name</th>

## <th>User Mobile</th>

## {/\* <th>Proof</th> \*/}

## <th>Action</th>

## <th>Add Comments</th>

## </tr>

## {complaint &&

## complaint.length > 0 &&

## complaint.map((data, index) => {

## return (

## <tr key={data.\_id}>

## <td width="150px">{data.complaint\_regid}</td>

## <td width="150px">{data.complaint\_description}</td>

## <td width="150px">Rubin Siby</td>

## <td width="150px">9048317092</td>

## {/\* <td width="100px"></td> \*/}

## <td width="230px">

## <div class="input\_field cmpselect\_option">

## <select

## // onChange={(e) =>

## // handleChange(

## // e,

## // data.\_id,

## // data.registrationNo,

## // data.description,

## // data.status

## // )

## // }

## >

## <option>Update Status</option>

## <option value="resolved">Resolved</option>

## <option value="processing">Processing</option>

## <option value="rejected">Rejected</option>

## {/\* {department &&

## department.length > 0 &&

## department.map((p) => {

## return (

## <option value={p.registrationNo}>

## {p.departmentname}

## </option>

## );

## })} \*/}

## </select>

## <div class="select\_arrow"></div>

## </div>

## </td>

## <td width="230px">

## <div class="input\_field cmpselect\_option">

## <textarea className="officer-comments"></textarea>

## </div>

## </td>

## <td>

## {" "}

## <button

## className="officerComplaintUpdate"

## // onClick={(e) => deactivateUser(e, data.\_id)}

## >

## Update

## </button>

## </td>

## </tr>

## );

## })}

## </table>

## </div>

## </div>

## </div>

## );

## }

## export default OfficerComplaintList;

**Admin**

**AssignComplaint**

import React, { useState, useEffect } from "react";

import axios from "../../../../axios";

import "./Complaints.css";

function Complaints() {

const [department, setDepartmentList] = useState(null);

const [complaints, setComplaints] = useState(null);

async function handleChange(e, id, regid, description, status) {

console.log(id, regid, description);

const data = {

complaint\_department: e.target.value,

complaint\_id: id,

complaint\_regid: regid,

complaint\_description: description,

complaint\_status: status,

};

let response = await axios.post(`admin/assign-complaint`, data);

if (response.status === 200 && response.data.verify === "success") {

getComplaints();

}

}

async function getComplaints() {

let response = await axios.get(`admin/getComplaint`);

if (response.status === 200) {

console.log(response);

setComplaints(response.data.complaint);

}

}

async function getDepartments() {

let response = await axios.get(`admin/get-departments`);

if (response.status === 200) {

setDepartmentList(response.data.departments);

}

}

useEffect(() => {

getComplaints();

getDepartments();

}, []);

return (

<div className="complaintList">

<div className="complaintTitle">complaints</div>

<div class="complaintDatas">

<div class="complaintData">

<table>

<tr>

<th style={{ padding: "20px" }}>Sl.No</th>

<th>Registration Id</th>

<th>Department</th>

<th>Description</th>

<th>Date</th>

<th>Action</th>

</tr>

{complaints &&

complaints.length > 0 &&

complaints.map((data, index) => {

console.log("length", data.length);

return (

<tr key={data.\_id}>

<td>{index}</td>

<td width="200px">{data.registrationNo}</td>

<td width="200px">{data.department}</td>

<td width="200px">{data.description}</td>

<td width="200px">{data.date}</td>

<td width="230px">

<div class="input\_field cmpselect\_option">

<select

onChange={(e) =>

handleChange(

e,

data.\_id,

data.registrationNo,

data.description,

data.status

)

}

>

<option>Select Deparment</option>

{department &&

department.length > 0 &&

department.map((p) => {

return (

<option value={p.registrationNo}>

{p.departmentname}

</option>

);

})}

</select>

<div class="select\_arrow"></div>

</div>

</td>

</tr>

);

})}

</table>

</div>

</div>

</div>

);

}

export default Complaints;

**Assign Complaint-Back-end**

router.post("/assign-complaint", function (req, res, next) {

console.log("bruuuu");

adminOperations.assignComplaint(req.body).then((response) => {

try {

if (response === "success") {

res.status(200).json({ verify: response });

}

} catch (error) {

res.status(401).json({ message: error });

}

});

});

assignComplaint: (data) => {

return new Promise(async (resolve, reject) => {

try {

let status;

console.log(data.complaint\_department);

const OfficerExist = await officerSchema.findOne({

userId: data.complaint\_department,

});

if (OfficerExist) {

data.complaint\_status = "Assigned";

await officerSchema

.updateOne(

{ \_id: OfficerExist.\_id },

{ $push: { complaints: data } }

)

.then(async () => {

console.log(data.complaint\_id);

await complaintSchema.updateOne(

{ \_id: data.complaint\_id },

{ $set: { status: "Assigned", officer: OfficerExist.\_id } }

);

})

.then(() => {

let status = "success";

resolve(status);

});

} else {

let status = "failed";

resolve(status);

}

} catch (error) {

console.log((err) => err.message);

}

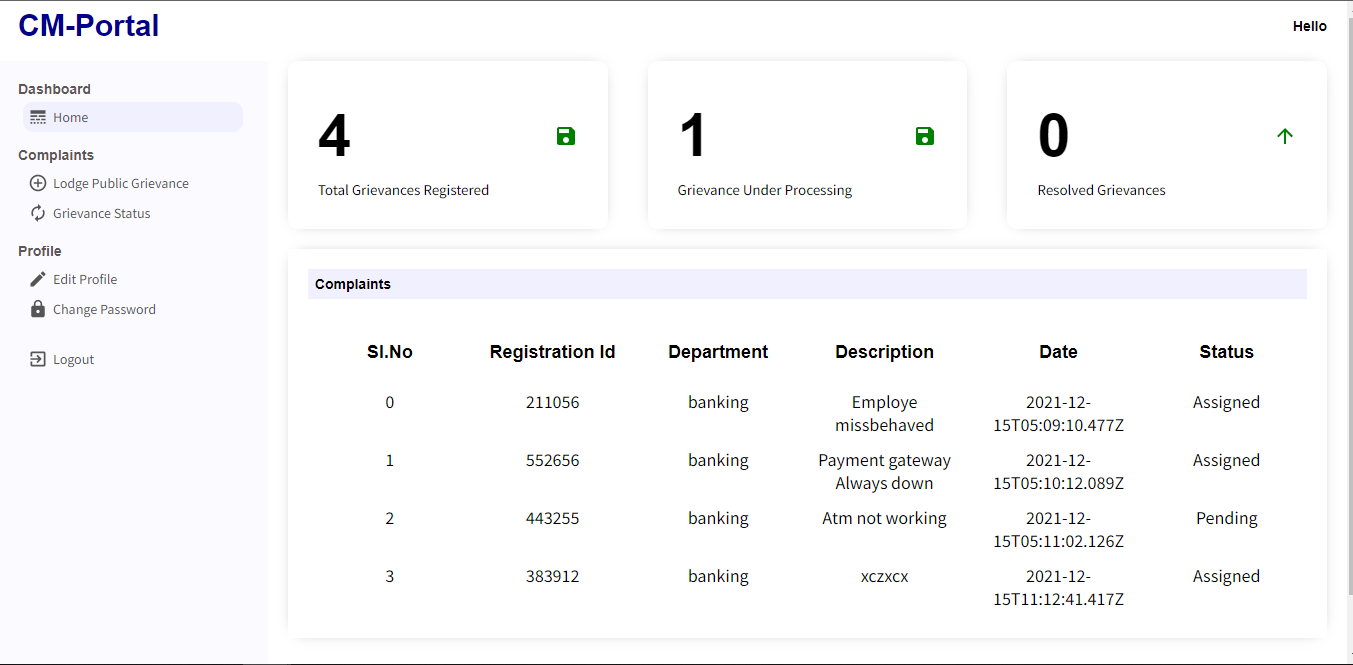
});

},

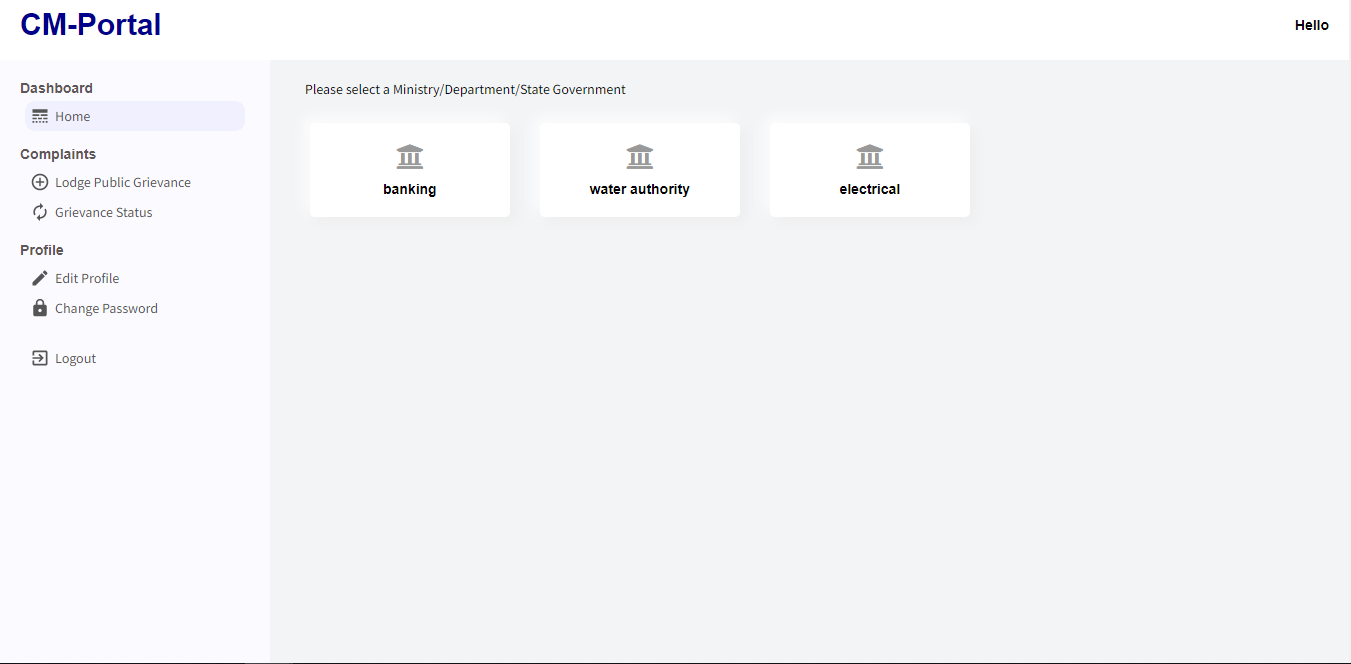
**9.2 SCREEN SHOTS**

**USER PAGES**

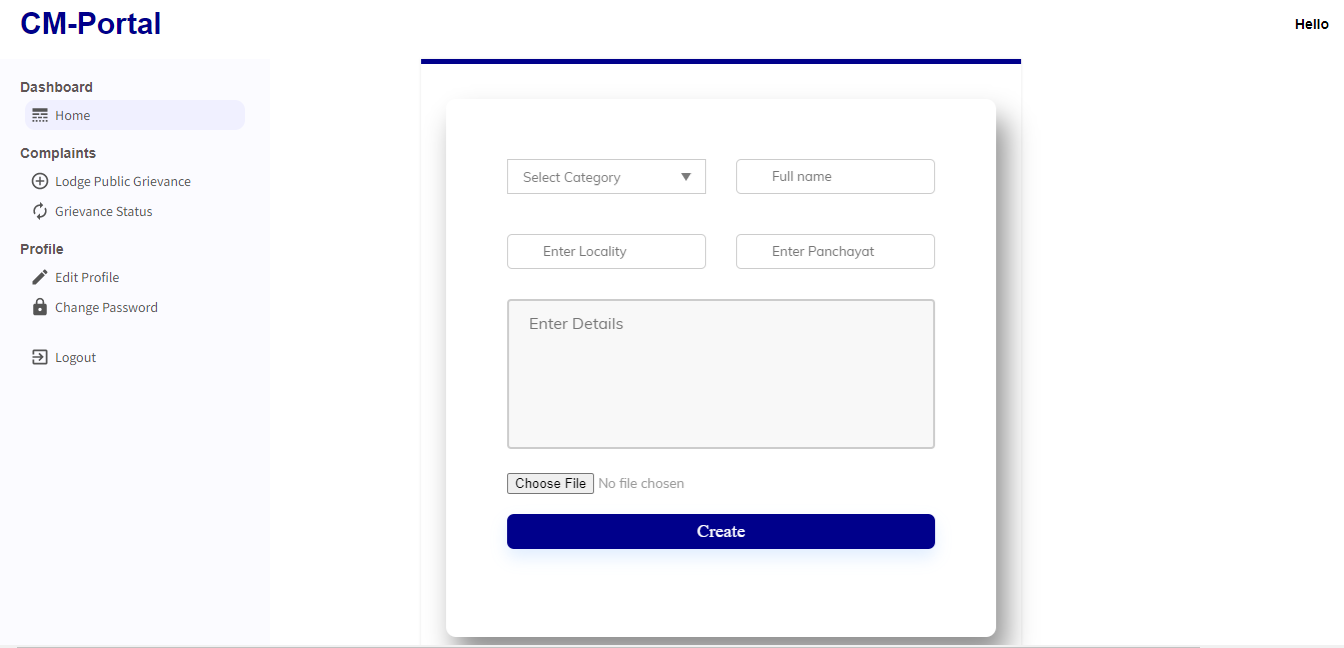
**User Dashboard page**



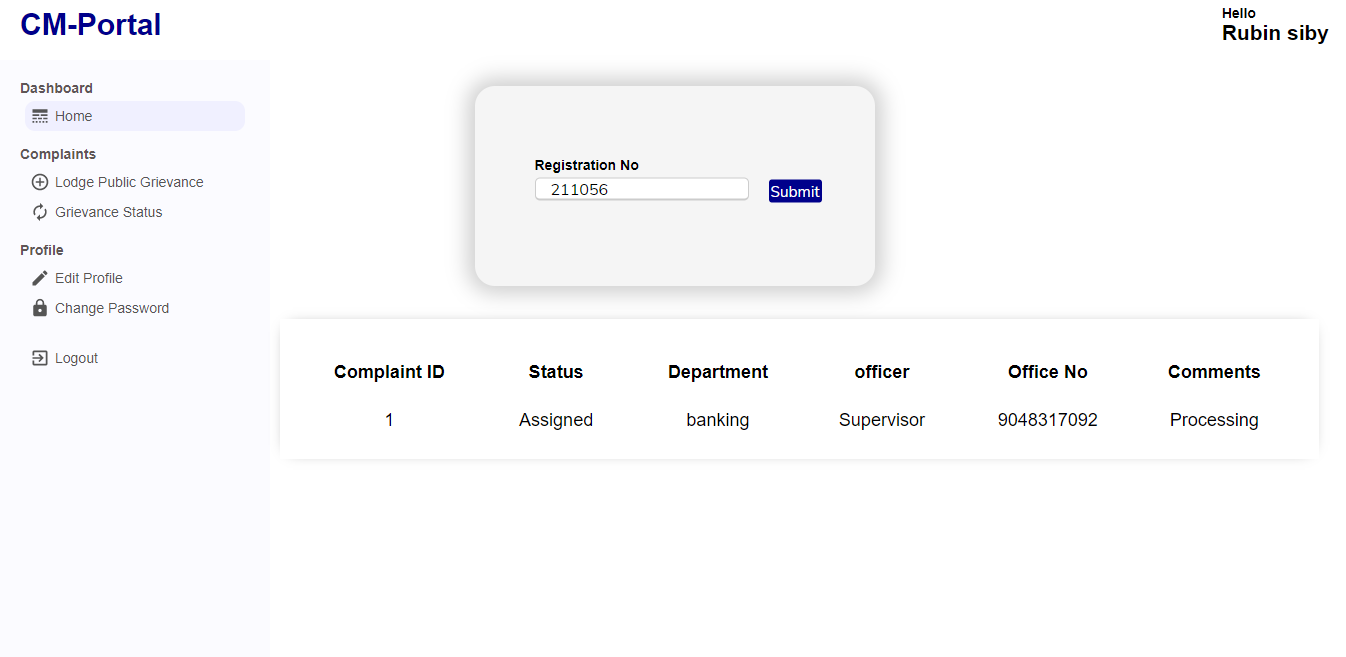
### User Complaint Category page



**Lodge Complaint Page**

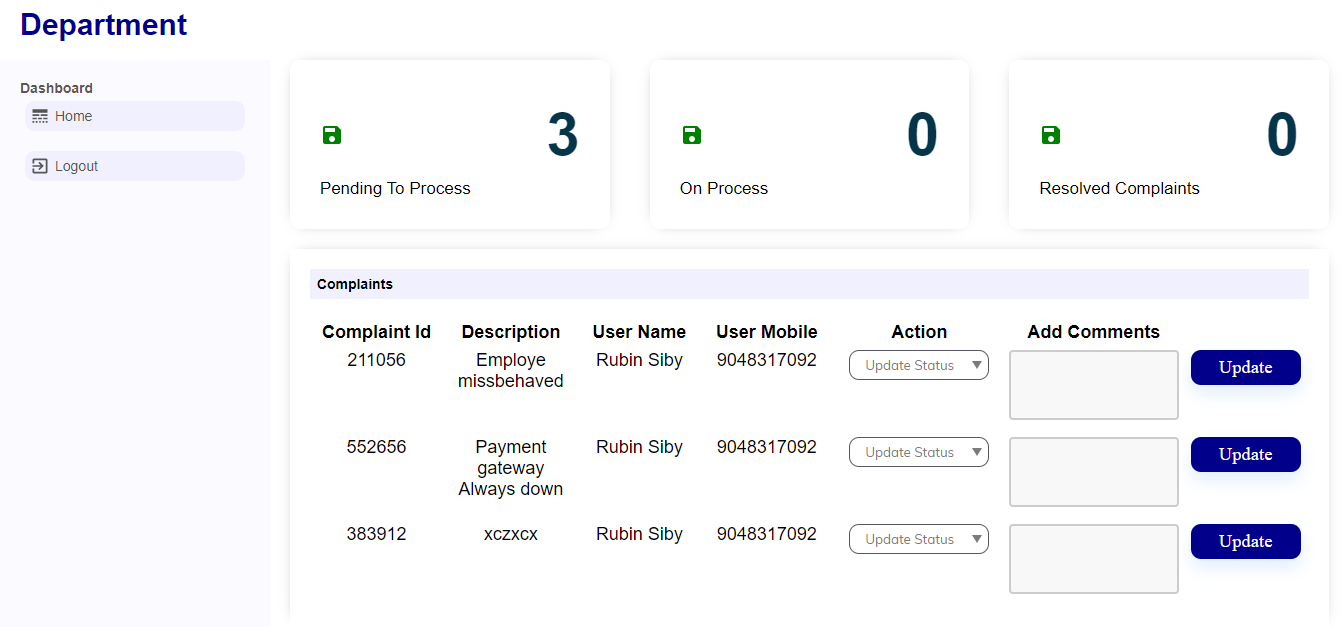


**Check Complaint Status Page**

****

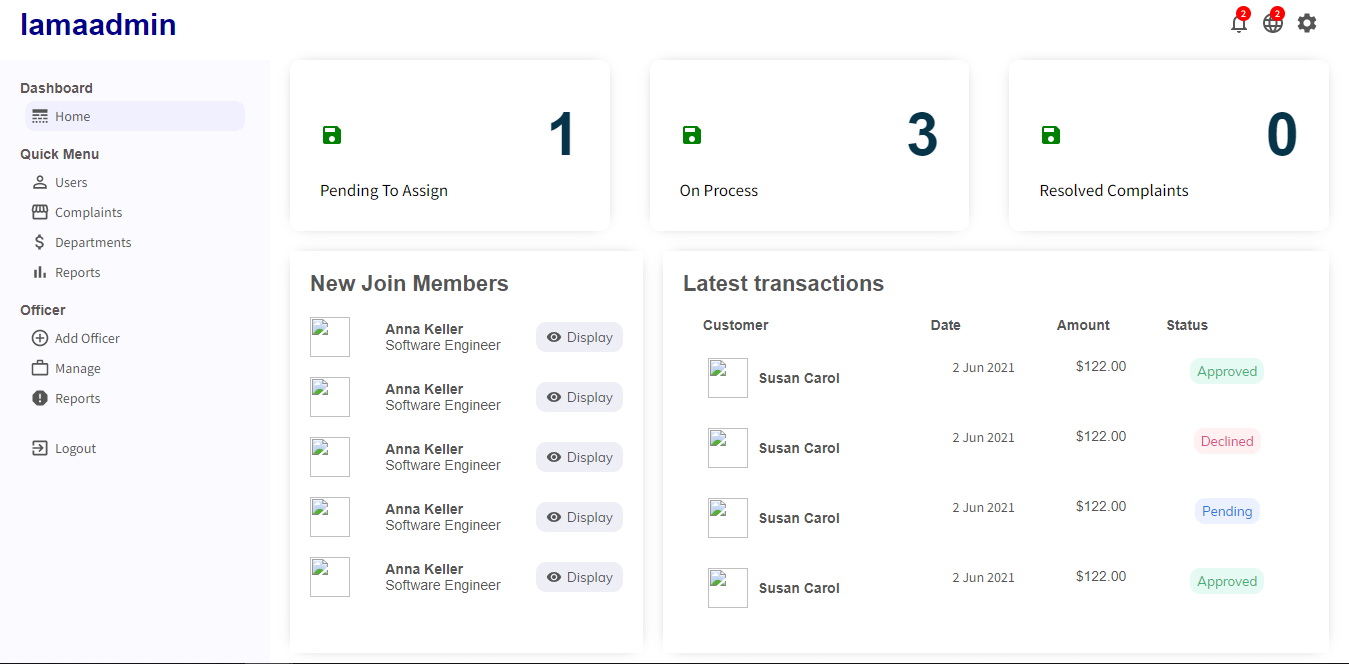
**OFFICER PAGES**

### Officer Dashboard Page

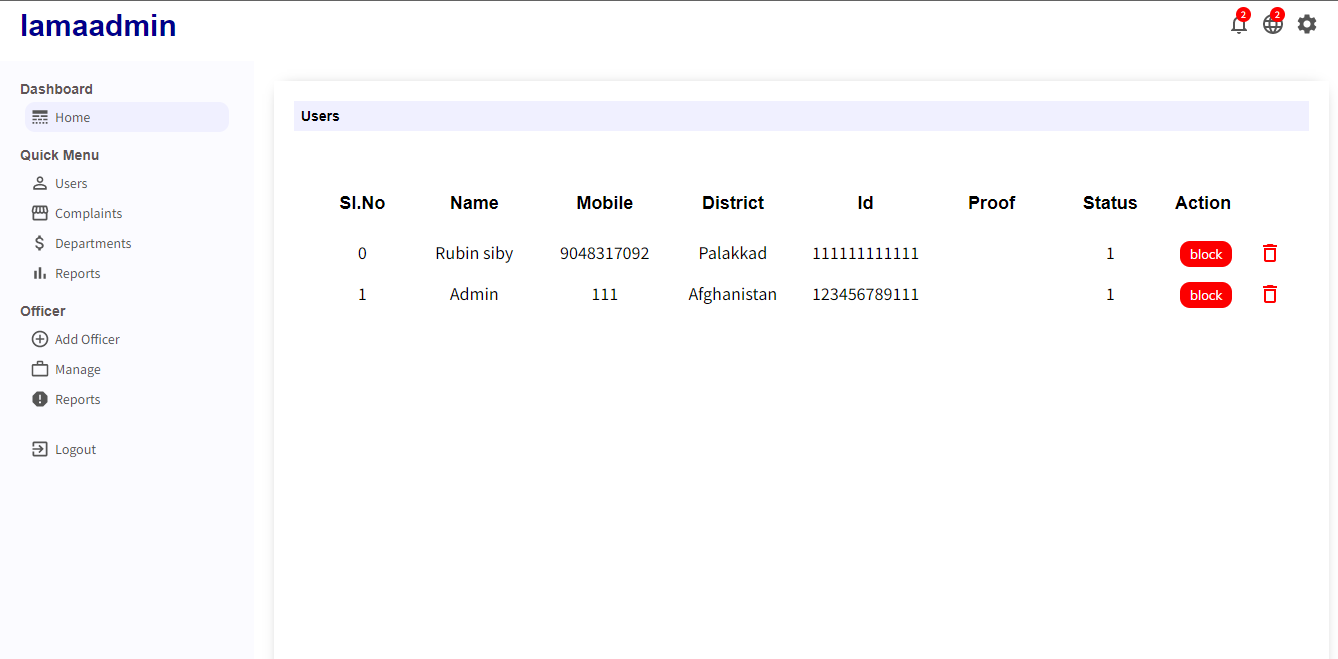
****

### ADMIN PAGES

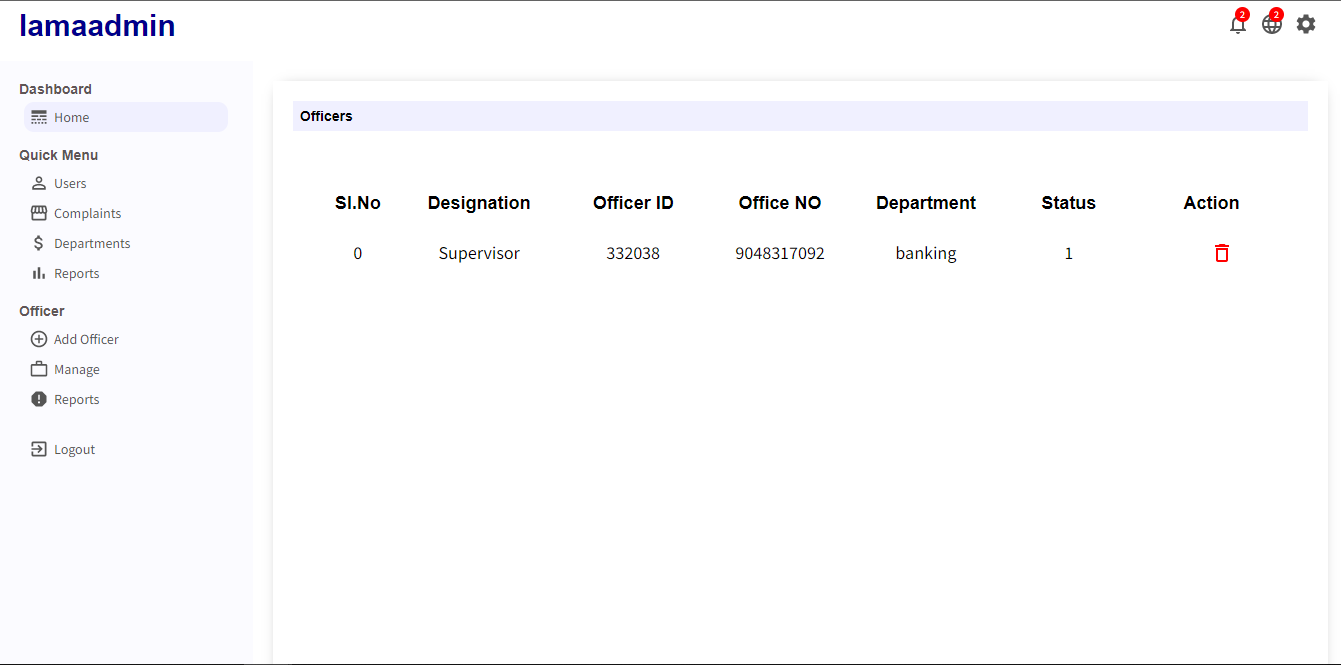
**Admin Dashboard Page**



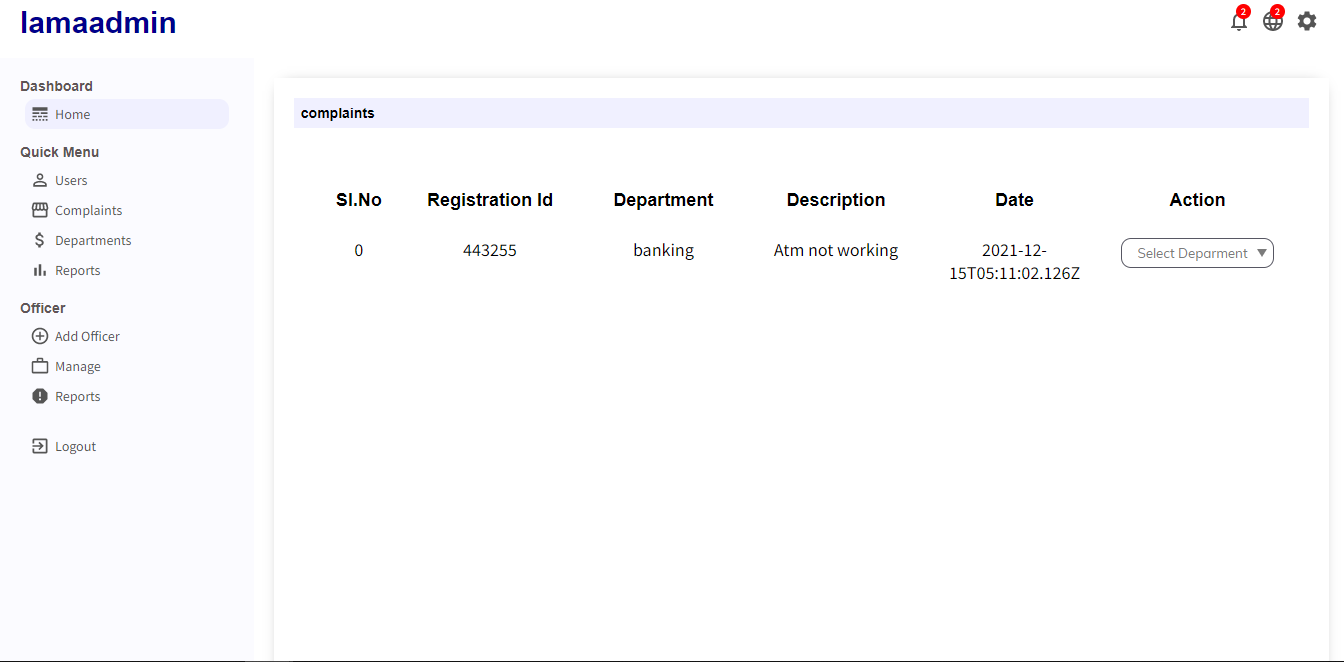
### View all registered customers



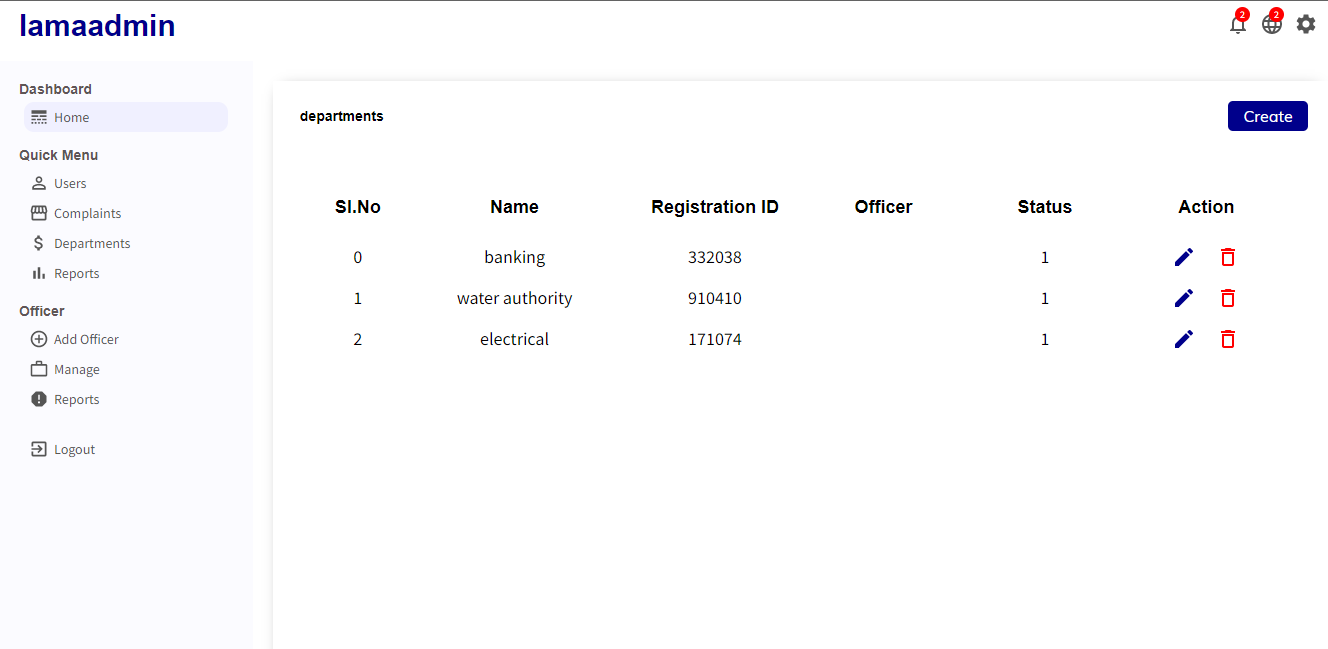
**Manage all created Officers**

****

**Assign Complaints**



### Manage & Create Departments

****

### Create Officers

