Chapter 2 Analysis

Introduction

The project life cycle begins with the analysis phase where proper research is done on every parameter that can or may directly or indirectly could affect the project and product after release. It is the part where every direction of project is identified and the measurements are taken to make the project go smoothly while development and after release. Analysis helps us to determine user and system requirements along with how it can be achieved. Analysis helps us determine how feasible is the system to develop, weather cost is feasible or not, what features are to be integrated in the system and what users want from the system.

2.1 Information Gathering

The first step of analysis is information gathering which helps us communicate with the users, what they want from the system and how important it is in the current context.

Different techniques can be used to gather information. Some of the techniques used during analysis are described below: -

a) Observation

Observation is the basic technique that was used to perform analysis for the need of problem reporting platform in this case. Observation showed that the many common problems around the locality are not solved and people are getting affected because of government's negligence.

Observation around the locality showed problems like broken roads and drainage problems. It also showed unmanaged traffic problems and electricity problems.

b) Interview

Interview is the most common technique used to gather information from the real-life scenario. From interview with the people in current context they suggested that the common problems around people's day today life have not been solved due to the political stability and current state of government. So people desired a system that can be used to pressurize government to take actions of the problems and solve it. Interview also helped determine that people want system to directly complain about the problem on the click of button.

c) Questionnaire

Questionnaire is the tool used to gather information from people which consist of series of questions which people are supposed to answer. The collected sets of data can be used later to decide user requirements and system requirements of the project.

Some of the questions that were asked in the questionnaire are listed below

i) What do you think you can do to report problems around you to the government?
ii) If you were to develop problem reporting platform what are the features that you want to integrate in the application?
iii) How often do you use mobile?
iv) What you think will mobile application or desktop application will be better to report a problem?

v) Will you do a survey on your mobile if that will help fix problems like road and drainage around your locality?
vi) Will you stop if you are on the way to report a problem where the report will be strong enough to change the locality?
vii) Will you do a survey if nothing of your personal data is collected to improve locality with your help?

2.2 Feasibility Study

Feasibility study is the beginning of the design stage where all the logical elements that could affect the system are analyzed and required measures are taken to make the project success. Feasibility study studies about expertise level required, qualitative and quantitative assessments of resources and estimation of time and cost along with the identification of critical problem.

Some of the factors studied in this project to identify the feasibility of system are described below:

a) Legal Feasibility Study
Legal Feasibility Study helps us determine if any legal implications can or may arise against the project. It helps project maintain all the legal and ethical requirements that a project must have. The problem reporting platform does not need any legal requirements but some ethical requirements are maintained in the project like no personal data is collected from users' mobile devices and all the collected data are open to public which anyone can view at any required time.
b) Economic Feasibility Study
Economic feasibility study deals with the cost that will be required during the development and implementation of the project in the society. Since this project does not need hiring other member to work on project, the tools used are open source which are free of cost. The only cost required are to establish a database and domain for web UI which is not much. Neither money is required to collect data as normal people will be the one submitting data to the database. So, this project is economically feasible to develop and implement.
c) Social Feasibility Study
Social Feasibility Study helps us determine if the application will be acceptable in the society or not, can it grab user attention or not and can any user file a lawsuit against the project or not and similar factors. The information gathering result shows that problem reporting platform is what users want and in a current context there are many problems in the society. So, this shows that this project is socially feasible for implementation.
d) Technical Feasibility Study
Technical Feasibility Study studies the technical factors of the project like system require, user knowledge required, the cost to availability ratio etc. The system required is android which most people now a days have and user with any level of knowledge can do the survey within the application. The tools required for development are open source which are freely available on internet and the cost required to establish web UI is not much. So technically everything required are available which shows this project is technically feasible to develop.
e) Operational Feasibility Study
Operational feasibility study deals with the factors like process used, evaluation of end product, implementation adapt to the need etc. It helps us determine if the product can operate

smoothly in the changing environment and what methods are to be taken to improve the system.

2.3 Analysis Methodology

Structured systems analysis and design method (SSADM) is the type of analysis methodology used for this project. SSADM is a rigorous document-led approach for the development of information systems.

SSADM is basically waterfall method for the design of information systems. The use of this kind of methodology helps generate a well-documented and accurate information systems.

2.4 Requirement Specifications

The requirement specification is whole description of the product that is going to be developed. The requirement specification is basically documentation of what users wanted and what the system is. Requirement specification must fulfill all the business need or what features users want.

Requirement specification is the agreement between customer and development team which describes their needs of the application and what the application will have to fulfill their needs. The ultimate goal of the requirement specification is to reduce the problem of redesigning of the system later after if customers wanted to change the requirements.

There are basically two types of requirement specifications. They are listed and described below

a) Functional Requirements

Functional requirements are defined as the tasks that product must do to fulfill the business requirement of the user of functional requirement are the functions of the product with which clients' daily activity requirements are met.

It includes interface requirements, business requirement, compliance requirement, and security requirement.

The functional requirements of the problem reporting platform are explained below in table.

ID	Title	Description	Rational	Dependencies
Fr 1	Admin Registration	A user must be signed up as admin at initial which has control over the whole database and product.	For username and password for admin.	None
Fr 2	Admin Login	Login for admin so that he can manage the whole application when needed.	For privacy and easiness for management and control.	Fr 1
Fr 3	User Registration	Registration for the user so that he can make complains by filling an survey.	For reporting problems.	None
Fr 4	User Login	Login for user after which users can start taking surveys.	For username and password to login as user.	Fr 3

Fr 5	Manage Users	For user management by the admin who can modify add or delete the user.	For management and privacy reasons.	Fr1, Fr2, Fr3, Fr4
Fr 6	Making Complaints	For reporting the exact problem by taking photos and submitting problem with description, picture and GPS location.	For reporting a problem.	Fr3, Fr4
Fr 7	Manage Records	For managing the data sent by users which contains problems details.	For generation of web UI, graphs, data visualization, and manage the database.	Fr1, Fr3, Fr 6
FR 8	View My Records	To view the complaints made by any user.	For visualizing data for one user.	Fr 3,Fr 4
FR 9	Reward User	For rewarding the user who makes continuous complaints.	To promote more users.	Fr3,Fr6

b) Non-Functional Requirements

Those type of requirements that are to be included in the project but does not have any functions within the application but would be better if application contained those features can be said as non-functional requirements. nonfunctional requirements are any requirements that cannot be categorized in the functional data or process requirements.

Non functional requirements are the factors that define the usability and performance parameters of the systems. Some of the non-functional requirements are described below: -

i) User Interface

Easy user interface is the non-functional requirement of any project. User interface does not have any functional requirement but a project would be better if it has easy user interface for the user which user can use to interact with the application.

ii) Security

Data submitted by the user must be kept secure and the application should not gather any kind of user personal data. The application or system is better if it has good security measures applied within the system so that the data and information are not misused for any other use.

iii) Availability

Availability of application is the non-functional requirement since it is not functional requirement but a application is better if it is available every time to the user if they want at any time of the day. A properly available system can gather and connect with more users if it is available every time for the users.

iv) Reliability

Reliability of the application can be described as its non-functional requirement because it is not described as functional requirement but a system will gain more trust with user if it is reliable for storing and retrieving the information from inside and outside the system.

2.5 Prioritization

The business requirement of the organization must be prioritized before the development of the application which helps us meet the cost and time factor of the project. If requirements are prioritized then the user can have at least the basic working application that will fulfill the basic business requirement even if all the aims are not achieved by the end of the deadline because of problems during the development.

For prioritization of the aims and features that a application will have MOSCOW prioritization technique is used. MOSCOW helps us identify must have, should have, could have and wont have features of the application that will pave the road for the upcoming development and implementation stages of the project.

Using MOSCOW prioritization the features are differentiated as below:-

I) Must Have Features

The features that fulfills the basic business requirement of project are must have features. Some of the must have features are listed below:-

i) Registration for one system admin with full control of application.
ii) Registration for users.
iii) Get GPS location of the Phone.
iv) Make Complaints about phone
v) Web UI to visualize the data.
vi) Rewards for user to promote users for making complaints.
vii) Store data locally if not online.
viii) Unregistered Users can make a complaint but cannot view unless not registered.
ix) In app user guide.

II) Should Have Features

Some of the should have features are

i) Easy User interface.
ii) Encryption system to secure data over network.

III) Could Have Features

Some Could Have features are

i) Online help.
ii) Rewards to user who makes maximum complaints.

IV) Wont Have Features

Some of the wont have features are listed below:-

i) Include multimedia content with complaints.

2.6 System Requirements

System requirements are the environment parameters that help what system can run the application. A application must be made to run on the existing system of the organization so that organization should not use more cost just developing the platform to run the newly purchased application.

System requirements include hardware and software requirements required to run the application. Some of the hardware requirements are listed below:-

i) Android system with processor higher or equivalent to 1 GHz.
ii) RAM equal or more than 512 MB.
iii) Storage required 10MB on phone.
iv) Screen resolution at least 1280*720 px.

Some of the software requirements are listed below:-

i) Operating System Android above or equal than lollipop or Android 5.0.
ii) Browser to view the data visualization done with Web UI.
iii) Back end database software-MySql.

2.7 System Architecture

In the current application scenario application should gather information from user, send to the database and retrieve from the database. So the system is a client server system so three tire architecture is used to develop this application.

This type of architecture consists of presentation tire, application tire and data tier. The components of system architecture are described below:-

i) Presentation Tier
Presentation tier is the top level that displays information related to services and functionality included inside the system. This system interacts with remaining component by sending data to the other system over the network..
ii) Application Tier
Application tier is the middle tier or main logic tier of the information system. This tier controls application usability and functionality by performing detailed analyzation and processing of data and information.
iii) Data tier
Data tier is the storage area for the data and information. It houses database servers where data are stored and shown in the presentation tier.

The diagram below shows the three tier architecture

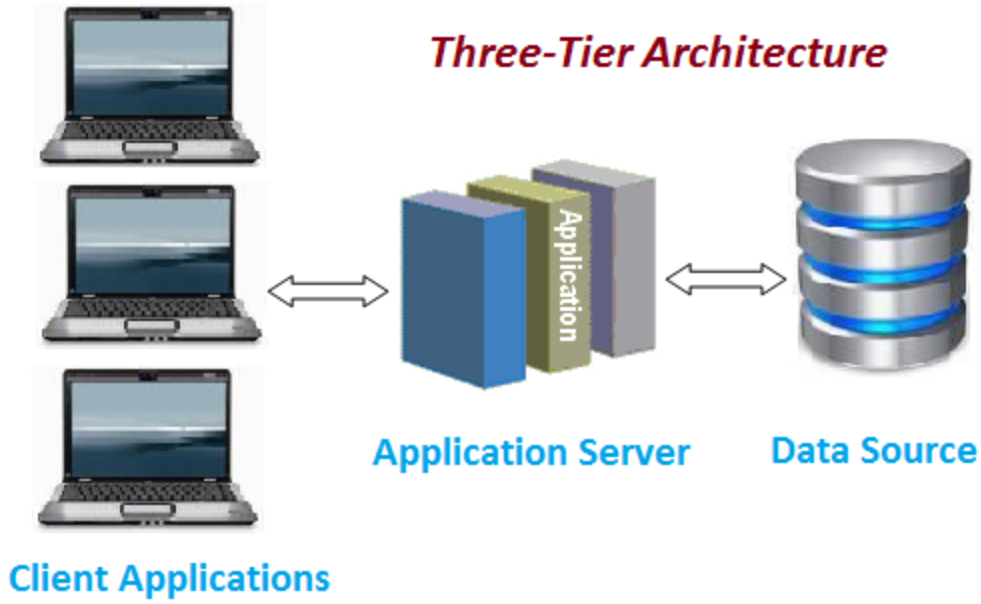

Figure 1 Three Tier application architecture

The main advantage of using three tier architecture is that each component can be developed individually. Because the programming for a tier can be changed or relocated without affecting the other tiers, the 3-tier model makes it easier for an enterprise or software packager to continually evolve an application as new needs and opportunities arise.

2.8 Natural Language Analysis

Natural Language Analysis is the technique used to find out how real life scenario gets embedded in the system by analyzing the scenario written in high level language. This process includes identification of nouns verbs and adjectives from the scenario which are possible class, method and attribute name for the application.

A non profit organization “Help for All” has been conducting awareness and helping people in need. The company is changing the focus for pressuring the government and help people solve the problems around the community. Due to political stability simple problems are neglected by the government.

Now this organization wants to user to collect the information lying around the locality like water, drainage, road etc. and has told to make a survey by user through their phone. Once the user is registered he/she can make complaints and view the records. After logging in the system does not need any internet connection for making complains or made are stored locally on the phone which gets uploaded the device is online. GPS location is noted and image is taken clearly showing the problem and some other information like road type and problem details. Also web to visualize data is necessary as per the requirement of organization.

After analyzing the scenario the possible classes are described below in table

Nouns	Verbs	Adjectives
Help for All, company, government, complaints, user, unregistered user, login, dashboard, registration	Add, edit, delete, remove, search, view	Name, address, phone, email, problem name, problem description, quality

2.9 Use Case

Use case diagram is the static diagram that show the human interaction with the system and what process they can perform as a certain type of user. It defines roles of system and user to achieve certain type of goal.

The main components of use case diagram are

I) Actor
Actor is the main user of the system who interact with the system to achieve certain type of goal. A system does not perform on its own without the actor.

II) System
System is the application which actor uses to fulfill their business requirements. System performs task along with user.

III) Use Cases
Use cases are the exact activity that a actor does to achieve the goal where system works with the actor to meet a certain type of need with the help of application.

The use case diagram for this current scenario is shown below

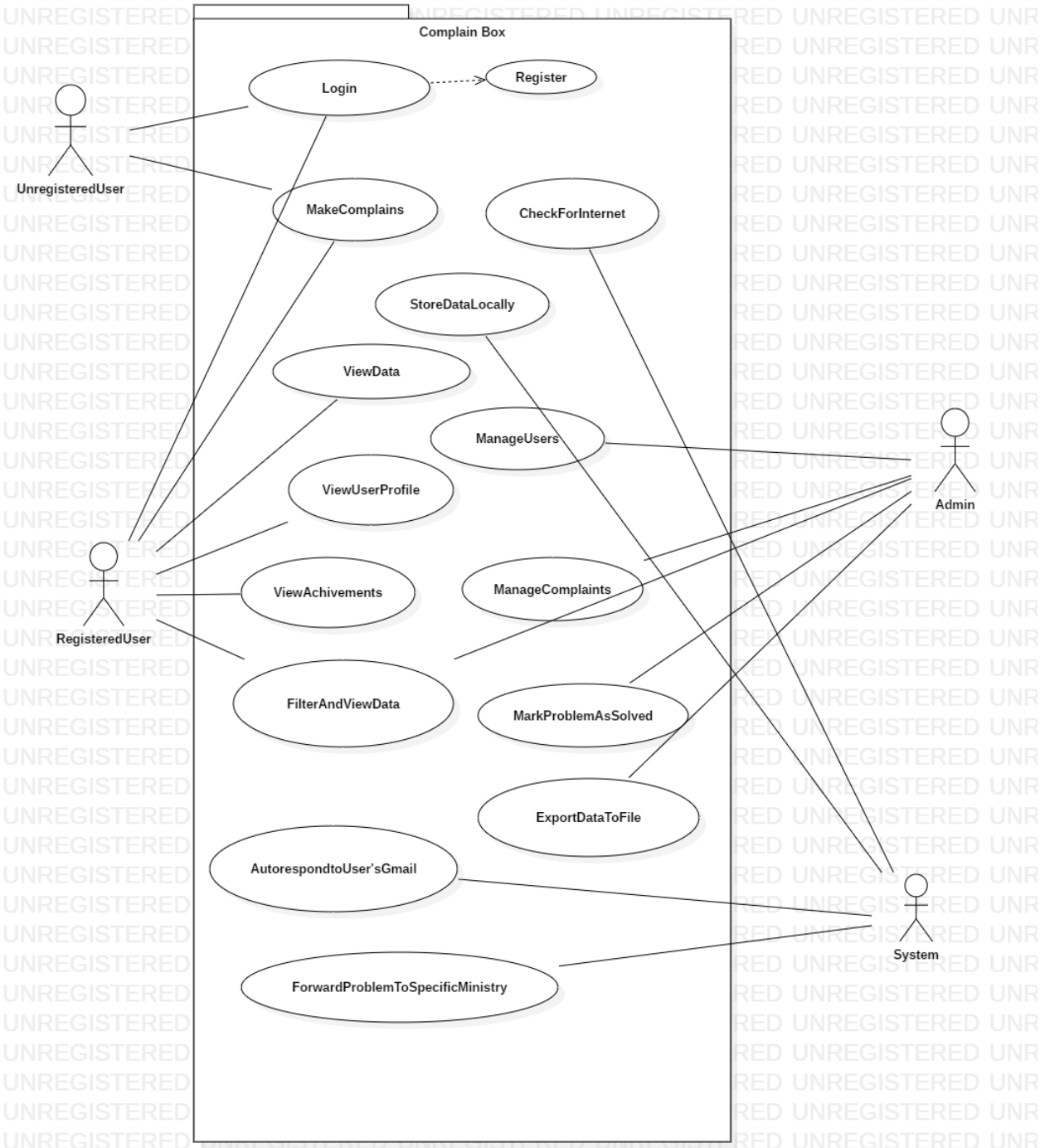

Figure 2 Use Case Diagram

2.10 Class Diagram

Class diagram is the static diagram that shows the structure of the system by showing the system's classes, methods and attributes and their relationship with each other.

The class diagram works as the basic building block of the system which shows the main elements, interactions in the system and the classes to be programmed.

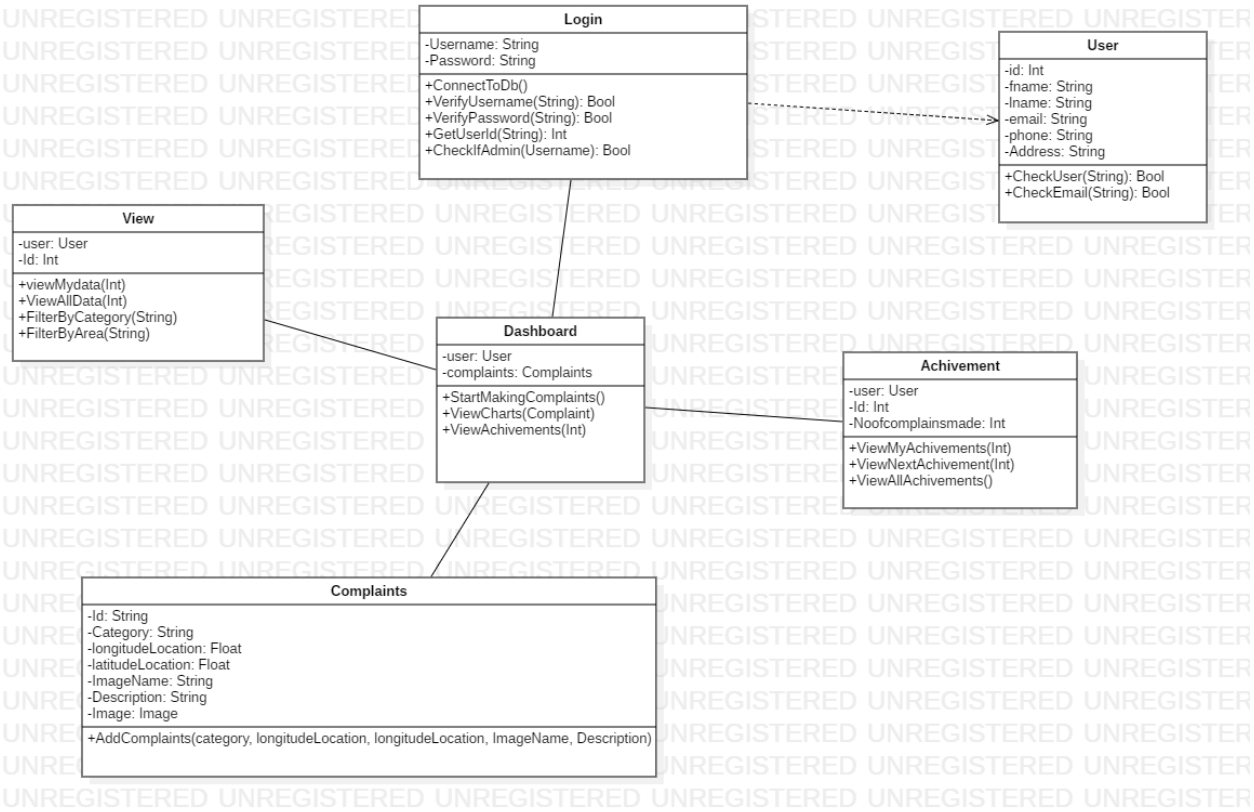

Figure 3 Class Diagram

Chapter 3 Design

Introduction

Design can be defined as the stage of software development where system is designed on the basis of user requirement and feasibility study. If the system is feasible enough to make then the solution must be presented considering the requirements, features wanted by the user are modelled which will help for code development in another stage.

The design stage is the most important part of software development where the logical system derived as a result of analysis is converted into physical diagram. Design is also a detailed description of what is needed to solve the original user requirement which are business activity of client.

In order to understand system from different view and create a solution different types of design pattern are used. The types of design pattern are

3.1 Structural Modelling

Structural Modelling is the type of application modelling which deals with the class and object composition. Structural class creation pattern uses inheritance to compose the interface for the user which user uses to complete the business activity.

Structural model of the application can be shown using different kinds of diagrams and pictures. For representing my structural model class diagram and flowcharts are used.

3.1.1 Class Diagram

Class diagram is the structural model of the program that shows the structure of the system by showing system's classes along with the attributes and operations and their relationship with the objects that are instantiated the runtime of the program.

Class diagram is the main building block of object oriented modelling that later can be used to translating objects into actual code that gets run on the computer to perform the business activity.

Different notations are used to construct class diagram. The notations used in designing the class diagrams are explained below:-

a) Class Visibility
   Class visibility is shown by writing characters in front of class name.
   To show the visibility of class different signs are used in the diagram. Some notations used are
   i) +
   + sign represents the class is public which means that class can be called from within the namespace.
   ii) –
   - Sign represents that the class is private which means the variables and methods are not accessible to other class.
   iii) #
   # sign represents the class is protected which means a class is accessible within the solution inside a single namespace.

b) Class
   A class contains code to manipulate data according to user requirement. The class in class diagram is shown as following inside a rectangular box with name on top row attributes on second and methods in third.

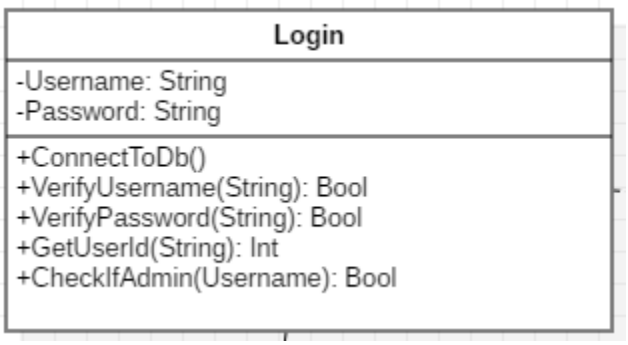

Figure 4 Example class diagram

c) Class Relationship

One class may be associated with one other to perform specific functionality with in the application. The relation of two class can be of different types and some are explained below

I) Inheritance or Generalization

If one class inherits the methods and attributes of another class then relationship between them can be said as inheritance or generalization. A class can inherit properties of windows form and act like one windows form .

Generalization on class diagram is shown as below:-

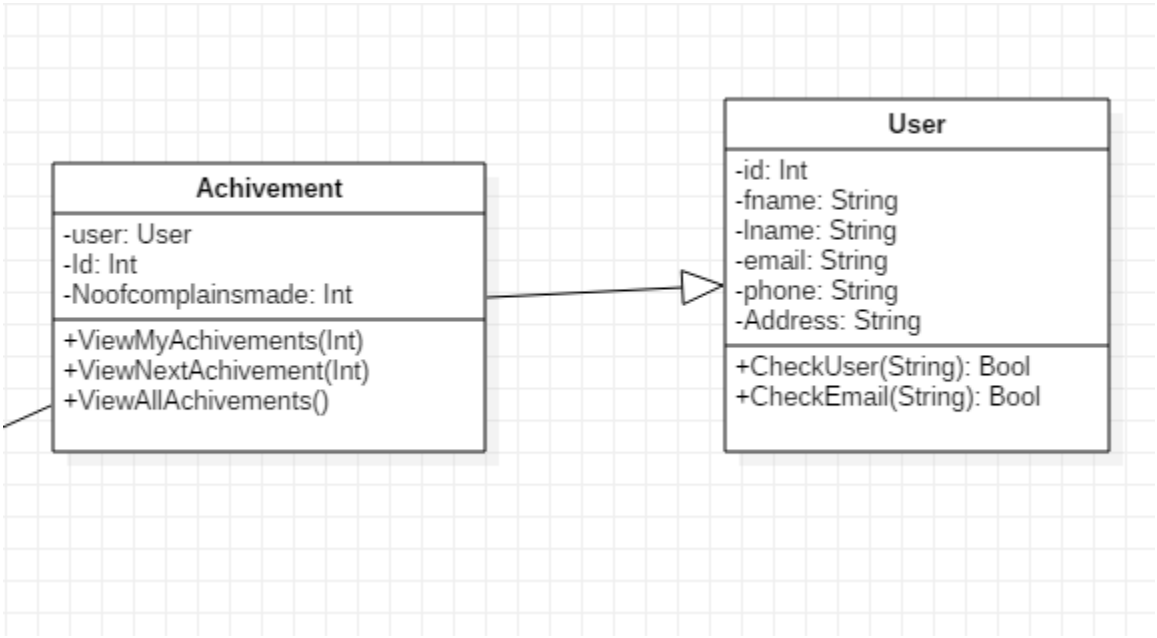

Figure 5 Inheritance Class Diagram symbols

II) Association

Association can be defined as the relation between two classes where one object wants another object to perform a service for it.

Association on class diagram is shown as below

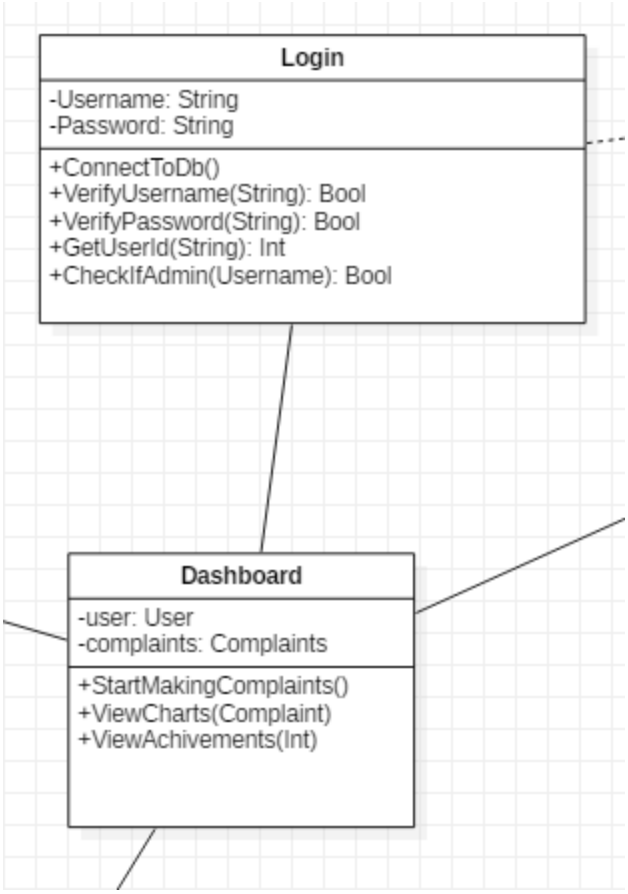

Figure 6 Example association example

III) Aggregation

The relationship between two classes is said to be aggregation relation if one class owns another class and owner class is not destroyed as object even if the owned class is destroyed. The aggregation function on class diagram is shown with unfilled arrowhead. The aggregation relationship can be shown as below:-

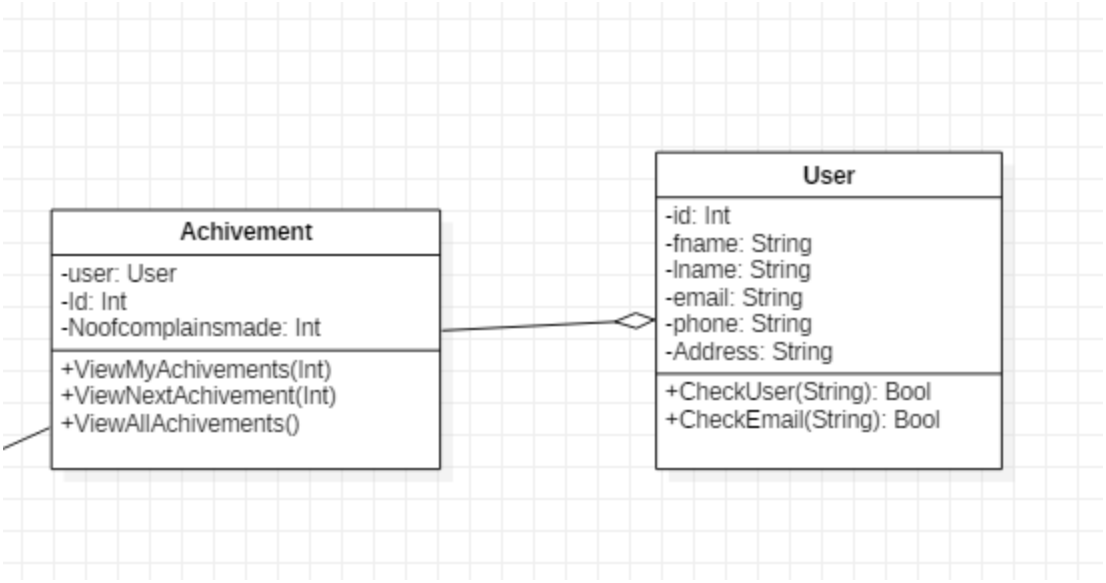

Figure 7 Aggregation Example

IV) Composition

The composition of two class can be said as composition relation if one class owns other where child class can't exist if owner class does not exist.

The composition in class diagram as shown below:-

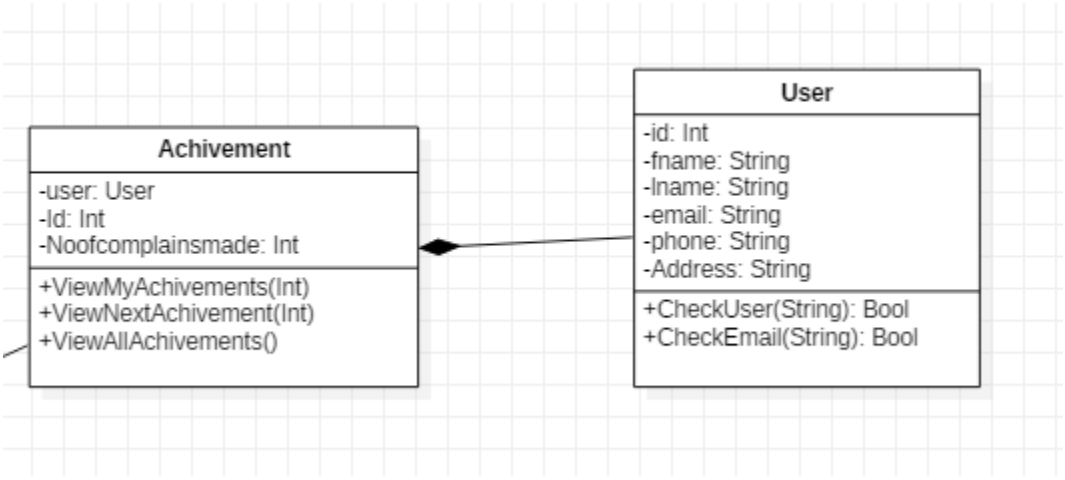

Figure 8 Composition example

The final class diagram of the application is shown as below:-

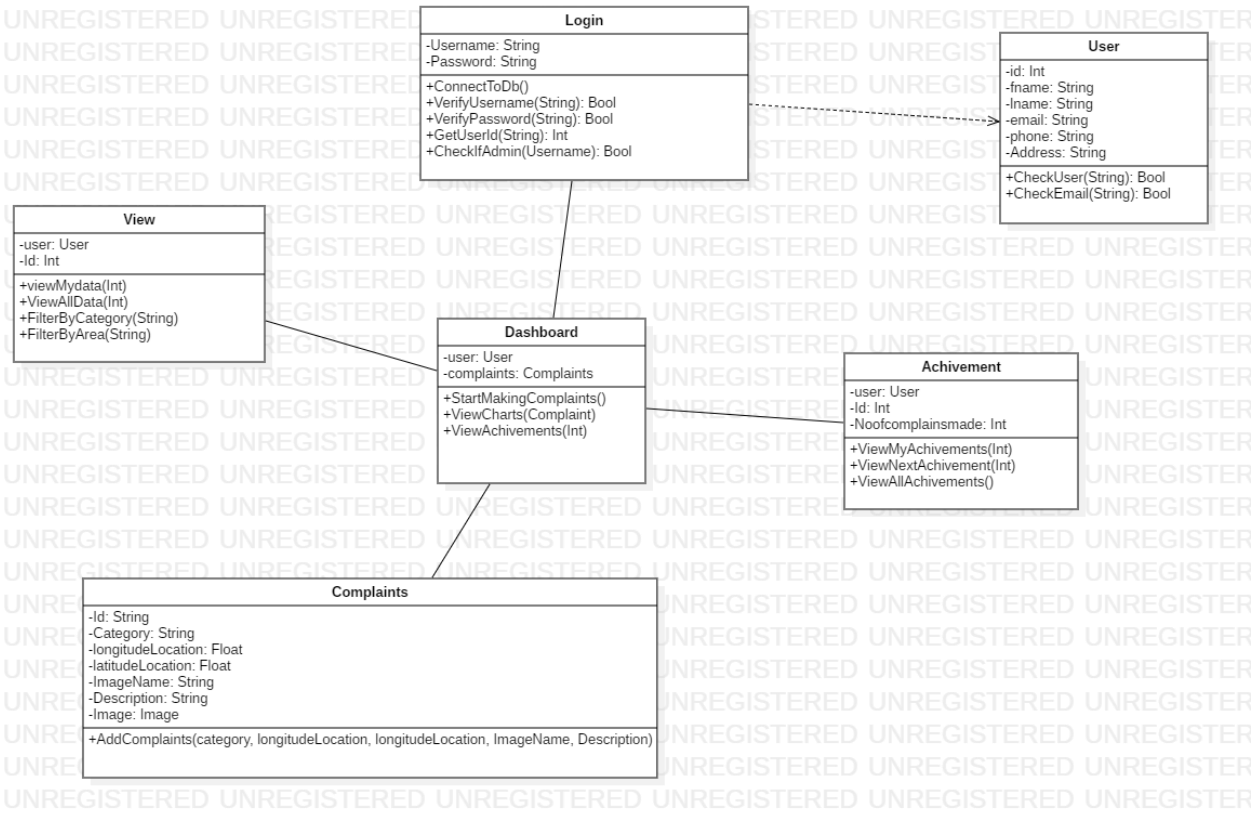

Figure 9 Final Class Diagram

3.2 Behavioral Modelling

Behavioral modelling is such type of application modelling which deals with the representation of behaviors of the application with the help of diagrams to achieve a business requirement functionality which are shown on the use case. A behavioral model shows the interaction between the objects using different types of symbols and diagrams.

Behavioral modelling of the application can be done by using different types of models. Some of the modelling techniques used are explained as below:-

3.2.1 Activity Diagram

Activity diagram can be defined as the type of UML diagram which explains the dynamic aspects of the system where the flow of the program is explained from one activity to another activity.

Activity diagram shows the flow of activity from one activity to another activity. Different notations used in creating the activity diagram are explained below:-

i) Start Point

Start point shows the start of the activity within the application. Start point on UML diagram is represented by a filled circle with attached arrow which shows the direction of flow. Symbol used to explain start point is shown below

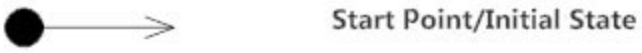

Figure 10 Start point symbol

ii) Activity

Activity is the action state of the system where it follows the direction on basis of what is to be done to the data. Activity in UML diagram is shown using a rounded edged rectangle. The activity is shown below:-

Figure 11 Activity Symbol

iii) Action Flow

Action flow shows the transition of the process from one to another. Action flow is represented with arrowed line. Example is shown below in diagram

Figure 12 Action flow diagram

iv) Object Flow

Object flow can be defined as the creation and modification of the objects by the activity to perform an action. Object flow is shown in image below:-

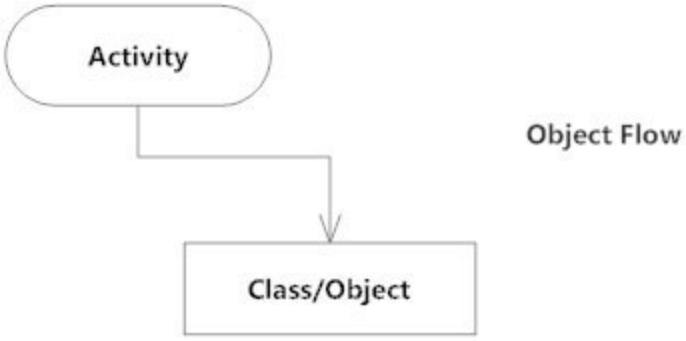

Figure 13 Object flow diagram

v) Decisions and branching

During the flow of the program if the system needs to perform some action which drives them to different actions on the basis of results. Decision in UML is shown using diamond and arrows from the diamonds which leads them to different directions. Decision is shown in image below

Figure 14 Descision symbol

vi) Merge Event

Merge event is the condition where two parallel process meet with one another and they have the flow of activity from one to another. Merge event is represented by different action flow meeting at a point. The image of merge event is shown below:-

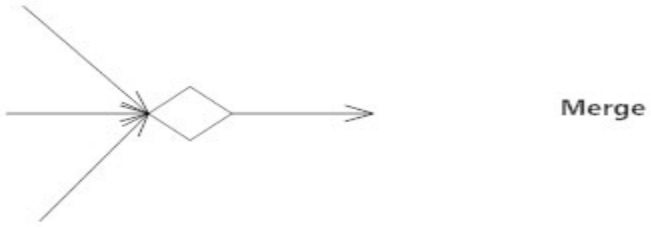

Figure 15 Merge symbol

vii) Join and Fork Node

Fork node splits one activity into many flows to different directions whereas join node assembles different activities to one single activity.

Join and fork nodes are also called synchronization because nodes assembles and disassembles single activity to many or vice versa.
Join and fork nodes are shown in the image below:-

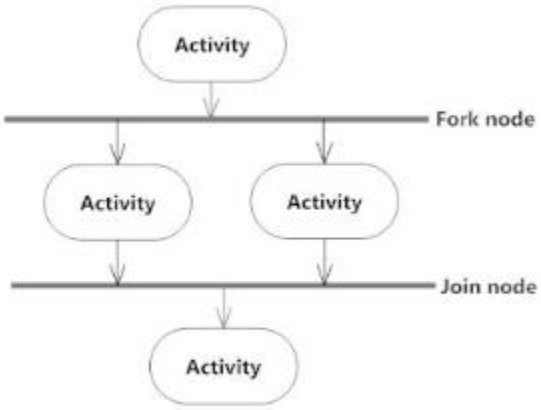

Figure 16 Join and fork symbol

viii) Swim Lanes

Swim lanes shows the job sharing and responsibilities for sub process of a business process. Swim lanes show the accessibility of one class with the other where messages are send and received from one to another.
The figure of swim lane is shown below

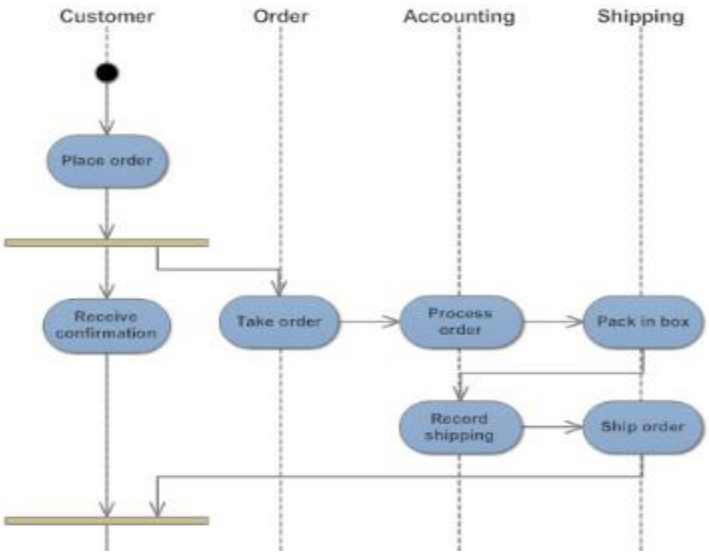

Figure 17 Swim lanes symbol in class diagram

ix) Send and receive signals

Send and receive signals are used to show message sharing functionality within the application. The signal must be received in order to move to the next activity. Symbols used to show message flowing from one to another are shown as below:-

Figure 18 Sent and Receive signal

x) Final State of end point

Final state is the end of the activity which is shown with a circle inside another circle filled. The final state is shown in the image below.

Figure 19 End point symbol

The actual activity diagram for the application is shown and explained below

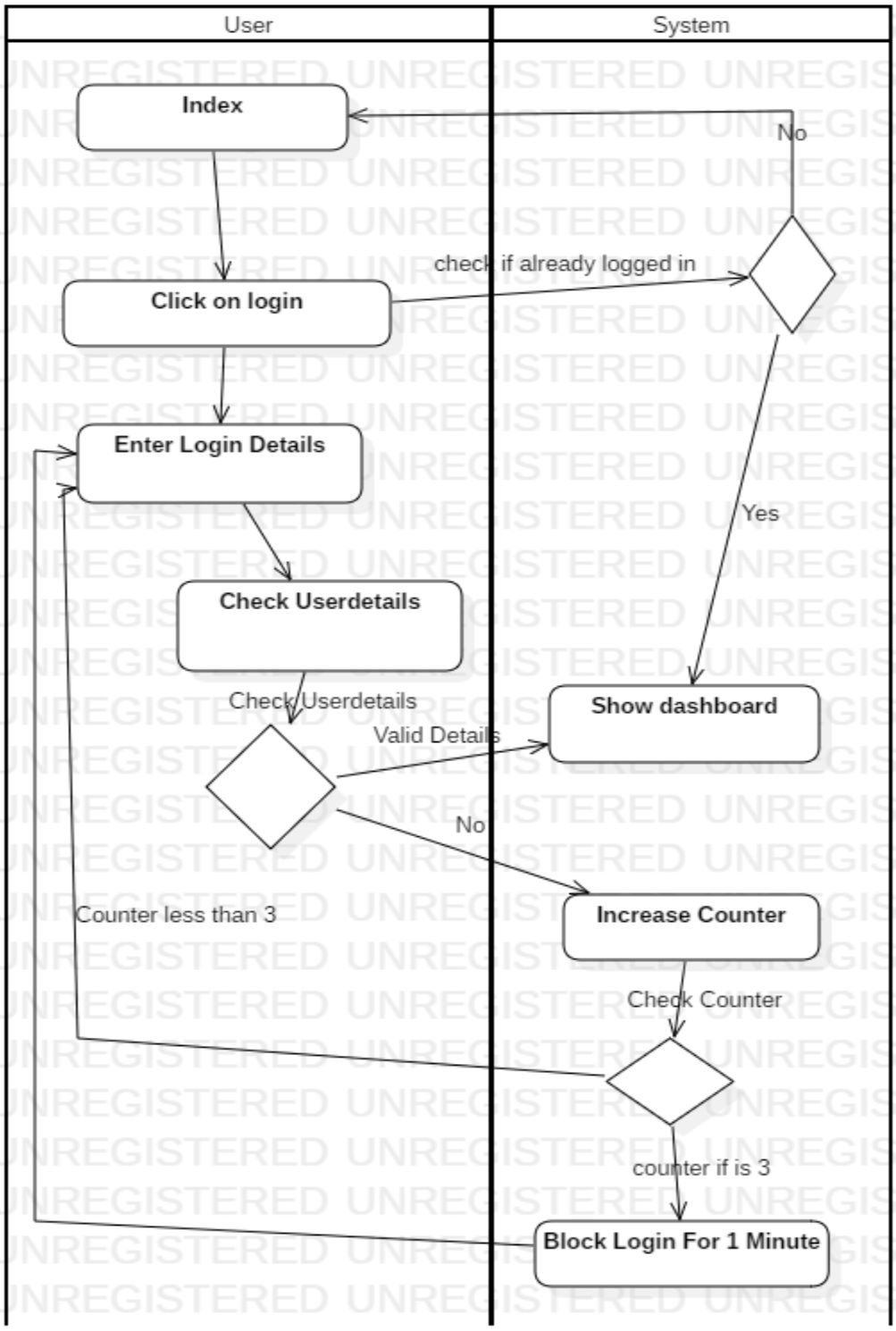

Figure 20 Activity diagram i

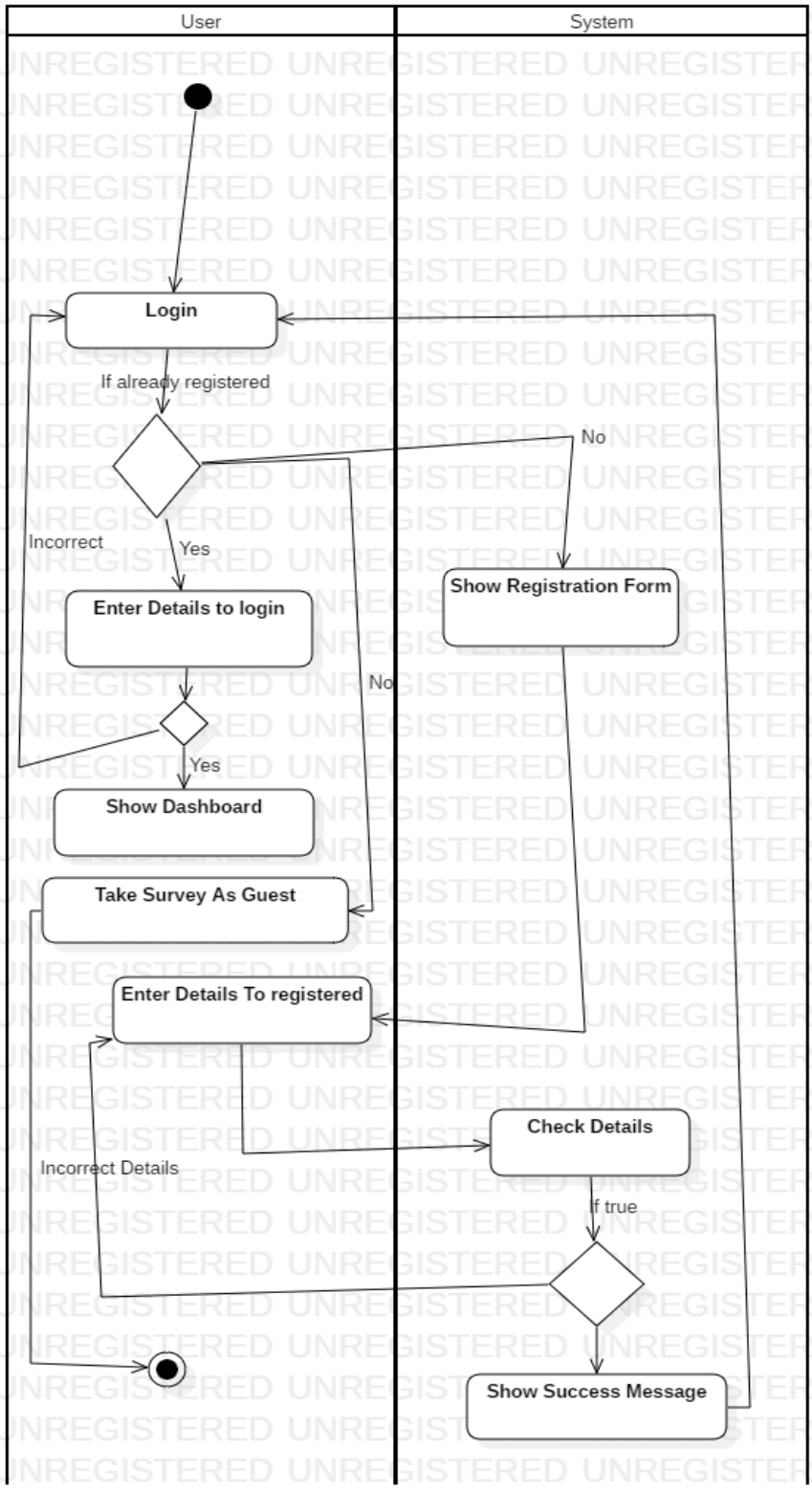

Figure 21 Activity diagram 2

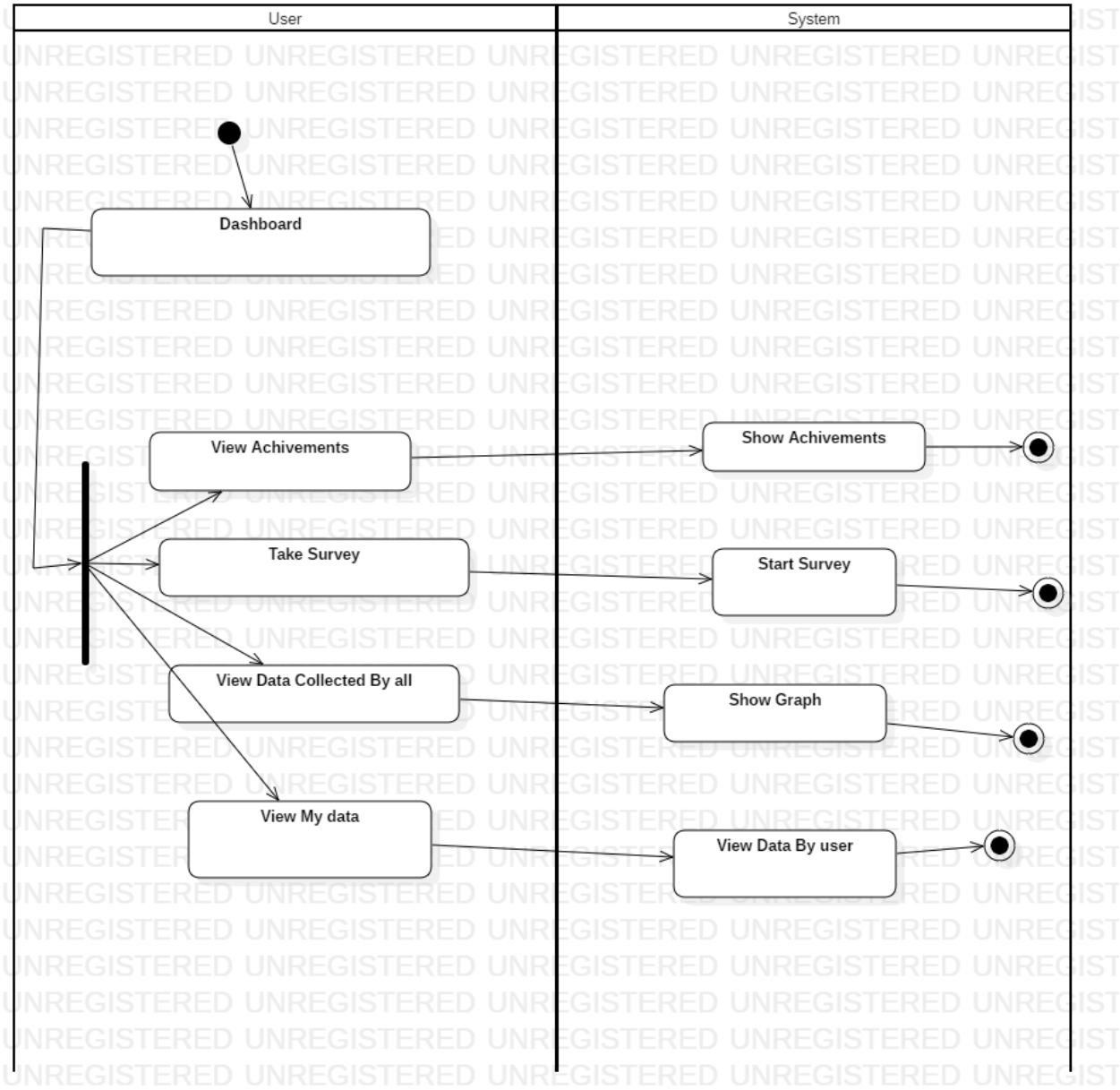

Figure 22 Activity diagram iii

3.2.2 Sequence diagram

Sequence diagram is behavioral diagram of the application which shows the interaction between the objects in sequential order in which the program flows. The order in which objects interact with each other to fulfill a functionality that is being integrated in the application is a sequence or order of execution. Sequence diagram describes the order of execution in the application.

Different symbols are used in the diagram to show flow of program. Some of the symbols used in the sequence diagram are described below: -

i) Actor

Actors are the users of the application who gives instructions to the system to perform the business activity. Actor interact with objects and interface of the system. Actors are outside the scope of application. Many users can be added in the sequence diagram to represent different level of users.

Actor is a stickman in the sequence diagram. The symbol of actor is shown below

Figure 23 Actor symbol

ii) Lifeline

Lifeline are drawn as a box with dashed line from the center of class which represents either role or object instantiation that participates in the sequence being modelled.

Lifeline shows the accessibility of class and message sharing from one class to the other.

The lifeline symbol is shown in image below

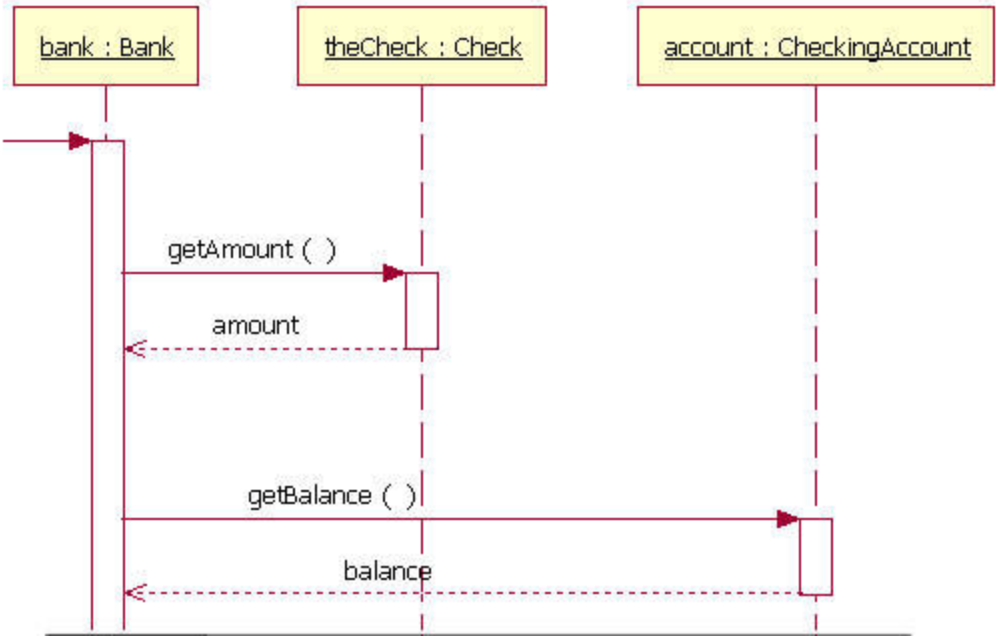

Figure 24 Lifeline symbol

iii) Messages

The application consist of different classes where they have to share message internally to fulfill the functionality. The messages appear in sequential order on the lifeline. Messages are communication protocols between two or more classes. Some types of messages are

- Synchronous Message
  Synchronous messages are those type of messages where reply must be got from other classes before executing another activity. The symbol used in synchronous message is shown in image below:-

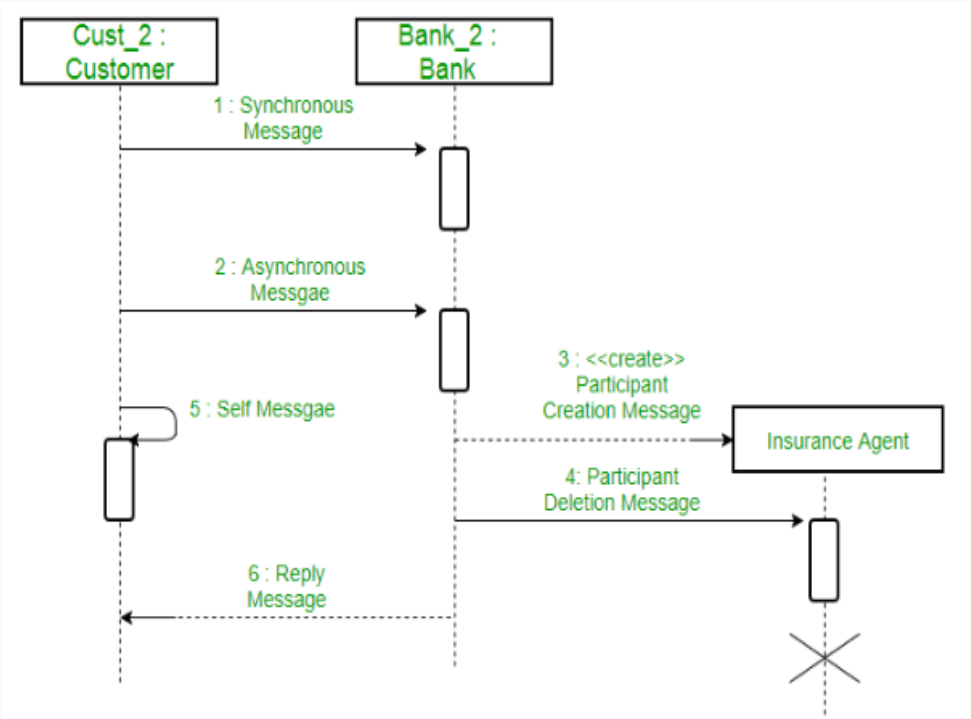

Figure 25 Message Flow symbol

- Asynchronous Message
  Asynchronous message is the type of messages shared between two classes where the reply is not to be waited before executing another activity.

  Lined arrowhead is used to show this type of message in sequence diagram.

  Following picture shows asynchronous message

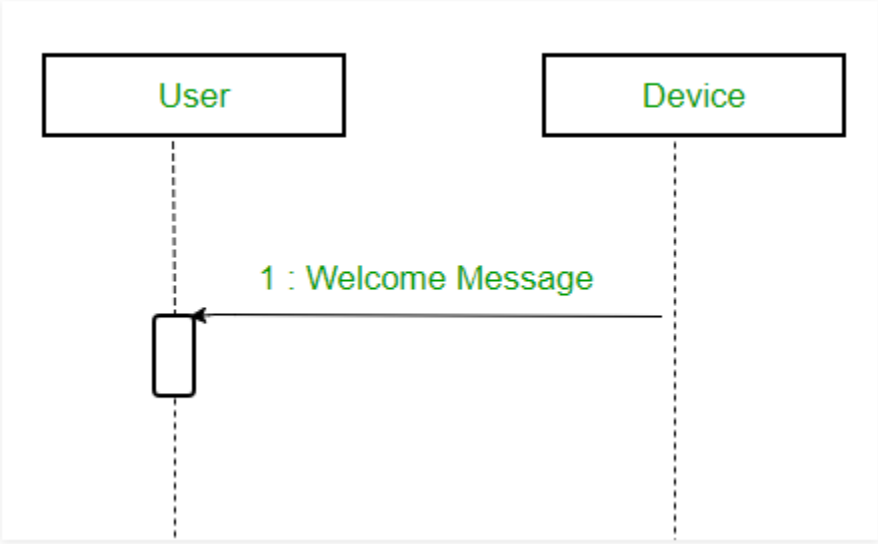

Figure 26 Asynchronous Message results

- Delete Message
  Delete messages are sent to delete the objects instantiated during the execution of the program. Delete message deletes the objects in the memory if occurrence exists.
  Delete message Is shown using dashed line terminating at 'x'. Delete message is shown in the figure below-

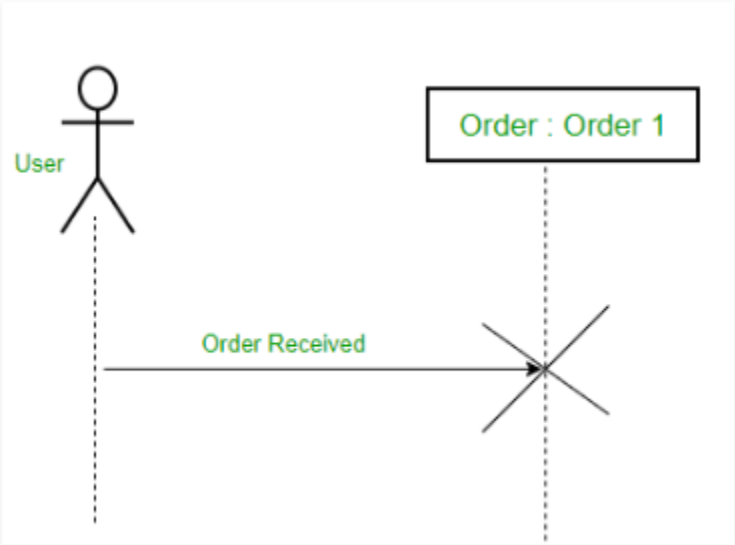

Figure 27 Delete Symbol

- Create Message
  Create message is used to instantiate the object of class in the program. Create message allocates space in the memory for the instantiated object. Create message is shown in the sequence diagram with dashed line with arrowhead and create labelled above the line. The symbol of create message is shown below:-

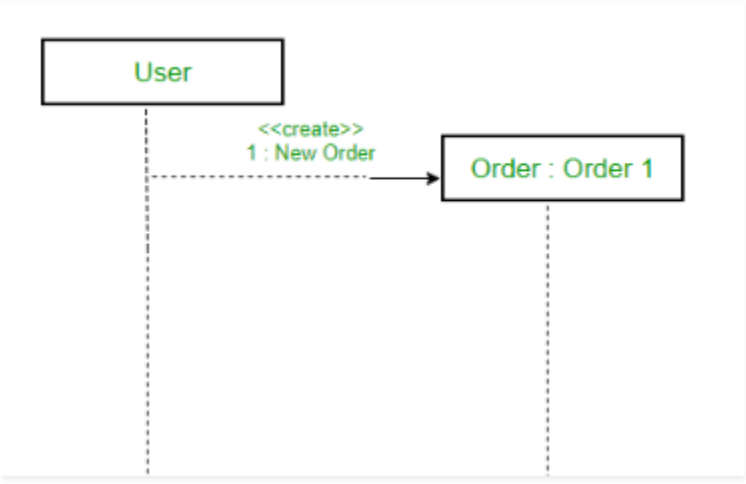

Figure 28 Create message symbol

- Reply Messages

Reply messages are the messages sent from the class to another in response of initial message sent. Reply messages are shown using the dotted line with arrowhead from one swim lane to another.
The symbol of reply message is shown below

- Lost Messages
  A lost message is used on such cases where the recipient is not known to the system. It is represented using an arrow directed towards an end point from lifeline.
  The symbol is used to represent lost message is shown below in the diagram

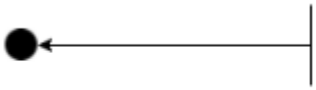

Figure 29 Lost Message Symbol

iv) Guards

In order to model conditions guards are used in the sequence diagram. Guards play an important role in letting software developers know the constrains attached to the system or particular activity.
The guards are shown in image below:-

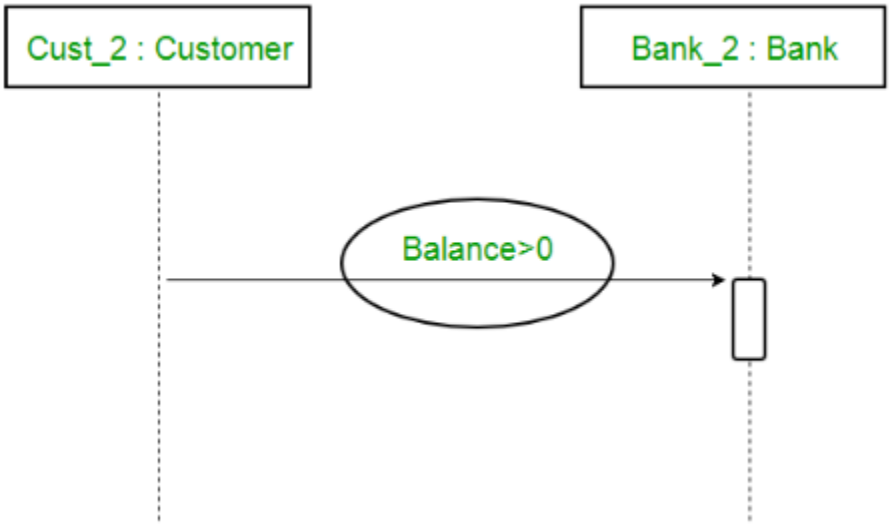

Figure 30 Guards Example Symbol

The sequence diagram for the system is shown and explained below:-

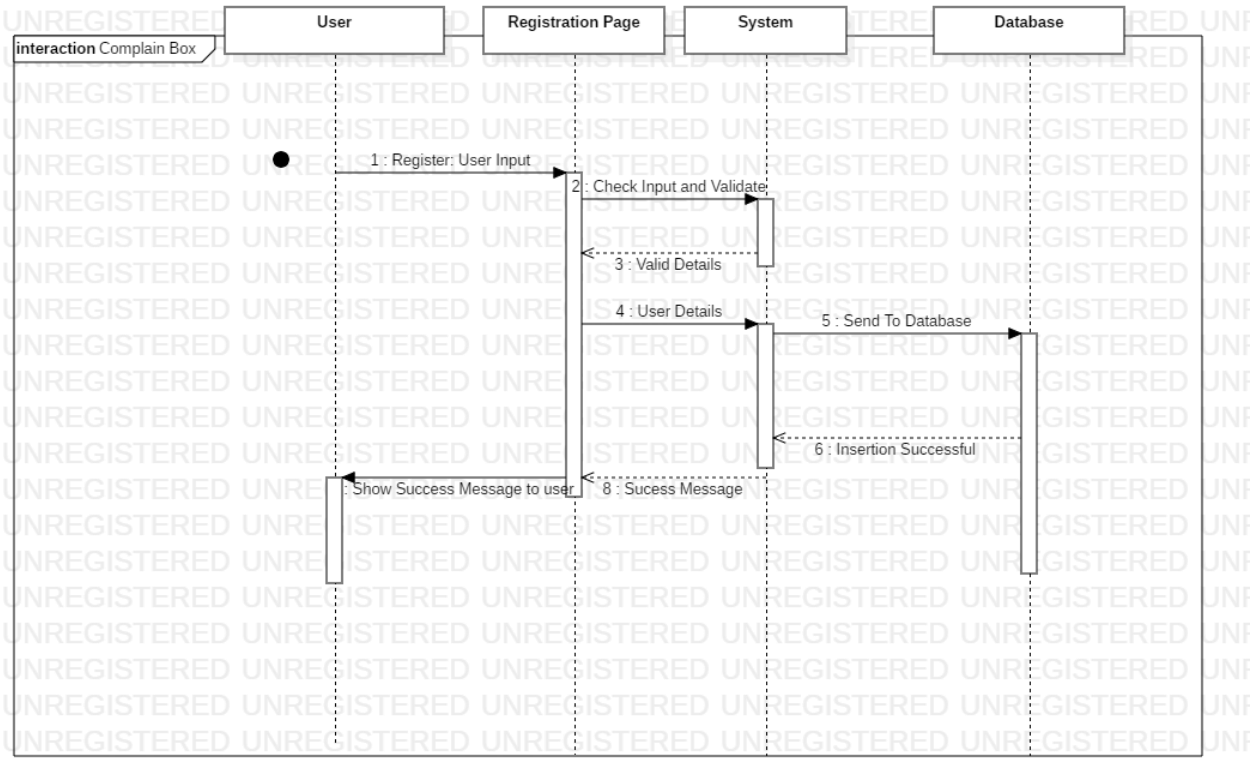

Figure 31 Sequence Diagram

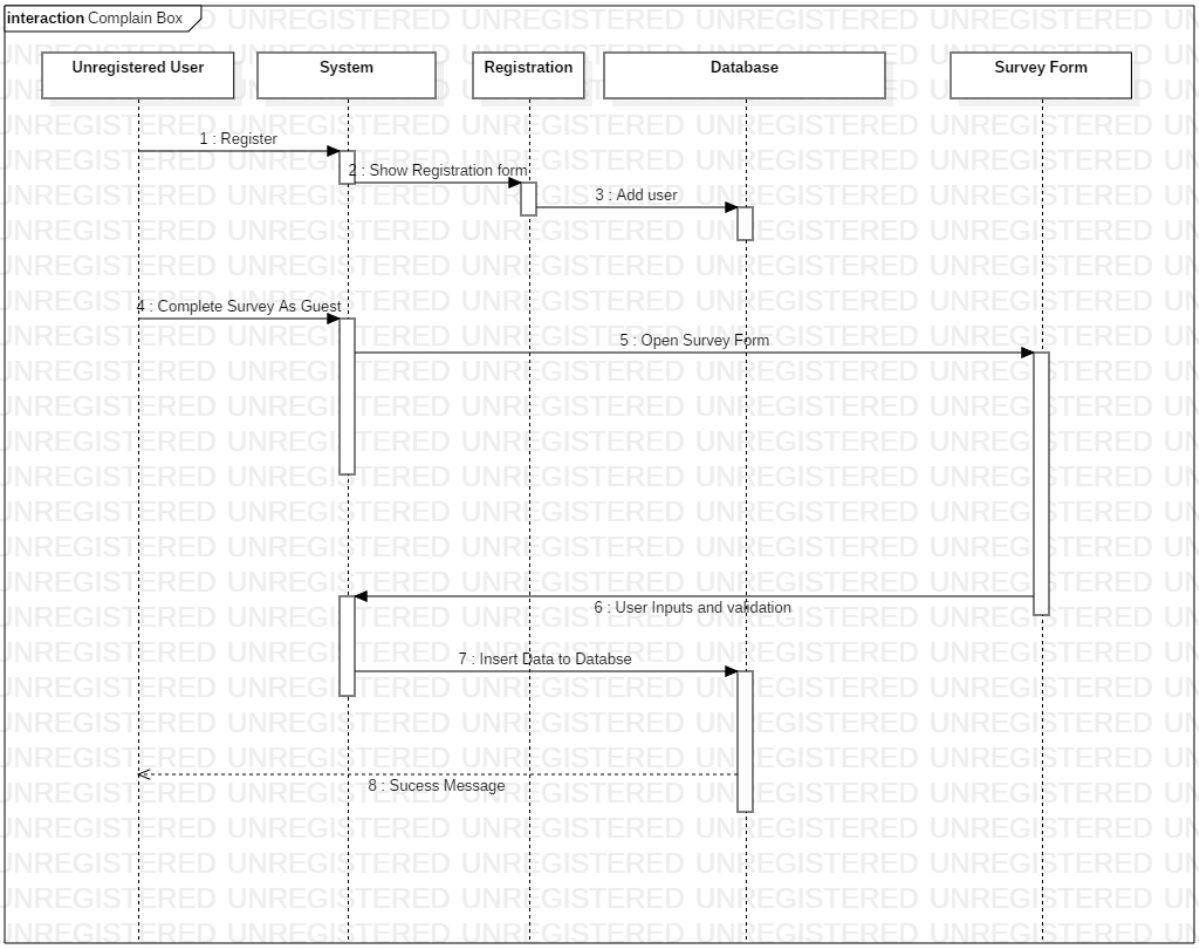

Figure 32 Sequence diagram ii

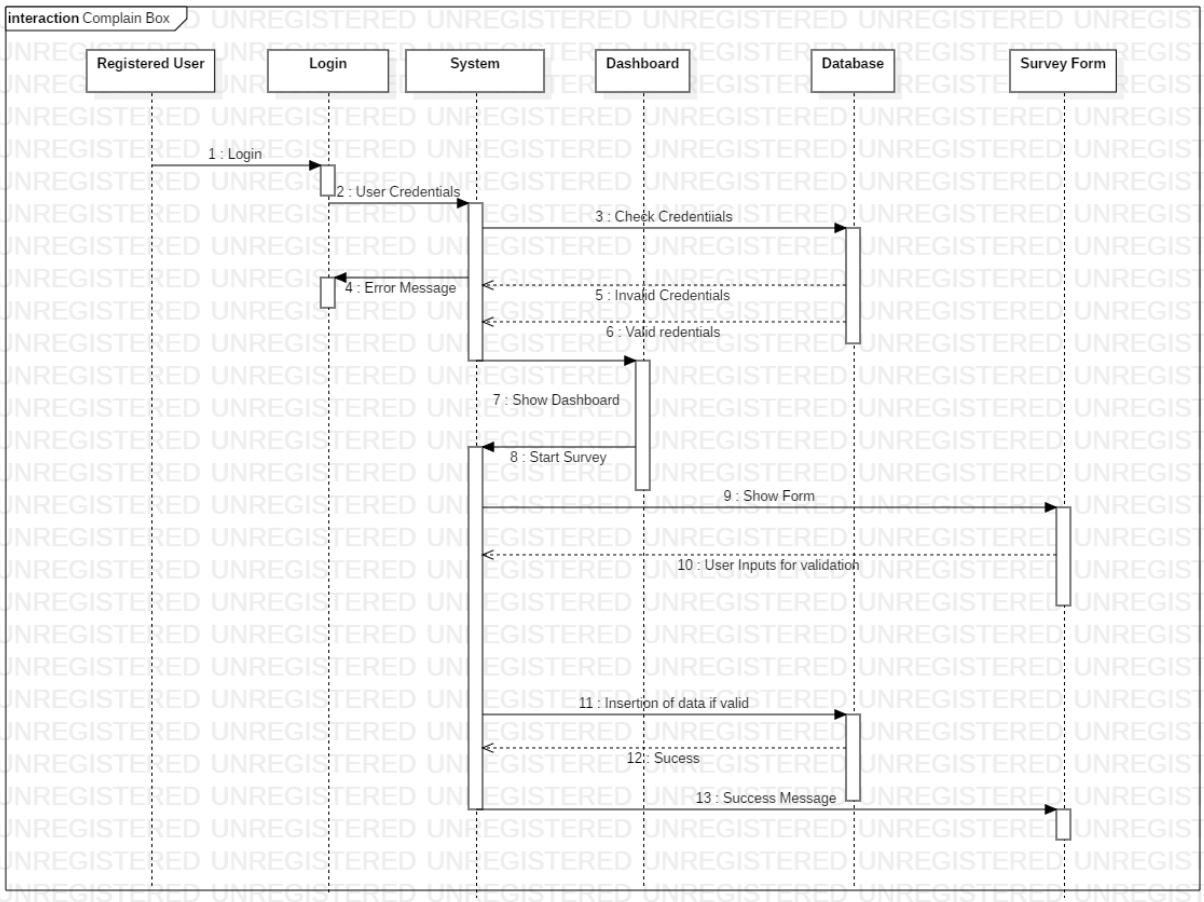

Figure 33 Sequence diagram iii

3.2 Database Modelling

Database modelling is a technique of modelling the database that determines logical structure of database and fundamentally determines in which manner data can be store organized and manipulated.

The database for this application is modelled using object-oriented manner so that objects can be managed easily and data are stored in the database.

The object-oriented database model is best known post-relational database model since it incorporates tables but is not limited to tables.

3.3.1 Data Dictionary

Data dictionary can be defined as the metadata of the tables in the database. Data dictionary contains the properties of the tables in the database.

Data dictionary provides detailed information about the business data such as standard definitions of data elements their meanings and allowable values.

The database of this system is modelled with object-oriented approach so that objects can be managed properly.

The data dictionary of the database is shown in table below

The database contains four tables for the storage of data. The tables in database are shown below

Table	Action	Rows	Type	Collation	Size	Overhead
tbl_electricitycomplaints	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
tbl_roadcomplaints	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
tbl_user	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
tbl_watercomplaints	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-

Figure 34 database structure

The data dictionary of tbl_user is shown below in diagram

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	fname	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
3	lname	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	address	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
5	mail_add	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
6	phone	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
7	username	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
8	password	varchar(50)	latin1_swedish_ci		No	None			Change Drop More

Figure 35 User table data dictionary

The data dictionary for tbl_roadcomplaints is shown below in image

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	user_id	int(11)			No	None			Change Drop More
3	road_type	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
4	longitute	double			No	None			Change Drop More
5	latitude	double			No	None			Change Drop More
6	problem	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
7	description	varchar(1000)	latin1_swedish_ci		No	None			Change Drop More

Figure 36 Data dictionary for table roads

The data dictionary of the table tbl_electricitycomplaints is shown below:-

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	user_id	int(11)			No	None			Change Drop More
3	pole_type	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	road_type	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
5	longitute	double			No	None			Change Drop More
6	latitute	double			No	None			Change Drop More
7	problem	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
8	problem_description	varchar(1000)	latin1_swedish_ci		No	None			Change Drop More

Figure 37 Data Dictionary for electricity complaints table

The data dictionary for tbl_watercomplaints is shown in table below:-

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	user_id	int(11)			No	None			Change Drop More
3	water_source_type	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	longitute	double			No	None			Change Drop More
5	latitude	double			No	None			Change Drop More
6	problem	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
7	description	varchar(1000)	latin1_swedish_ci		No	None			Change Drop More

Figure 38 Water Complaints data dictionary

3.3.2 ER Diagram

ER diagram can be defined as the structural diagram of the database showing tables and attributes to use in database creation, management and maintenance.

ER diagram helps later in producing the real database for the company or the organization.

ER diagram of the system above is shown in image below:-

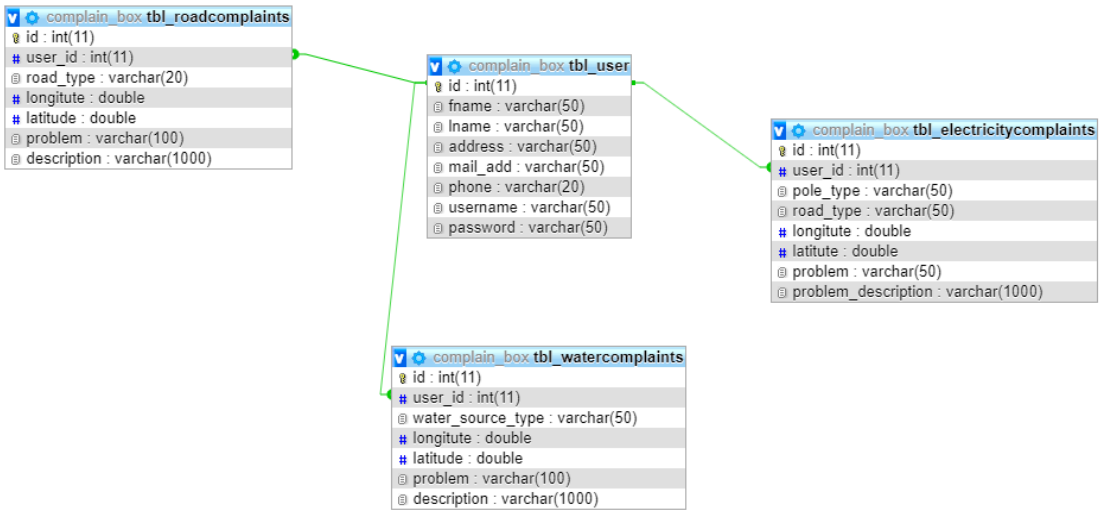

Figure 39 ER Diagram

3.3 UI Modelling

UI modelling is the technique of designing user interface for the user so that user can get the best easy possible solution to interact with the system. UI plays important role in the usability of the application.

Good interface is a non-functional requirement of the application so that user can have best experience with the system. UI of the system must be kept as easy as possible for maximum usability within the application.

3.4.1 Prototyping

Prototyping is the technique for creating the UI of the system being developed on other medium which can be kept as reference for the development of the application.

Prototypes are rough diagrams that can be changed for modelling the application.

The prototypes developed for this application are shown below:-

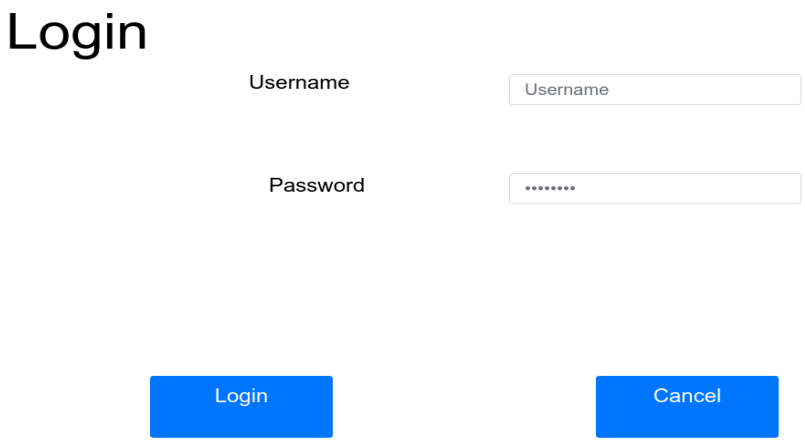

Figure 40 UI modelling

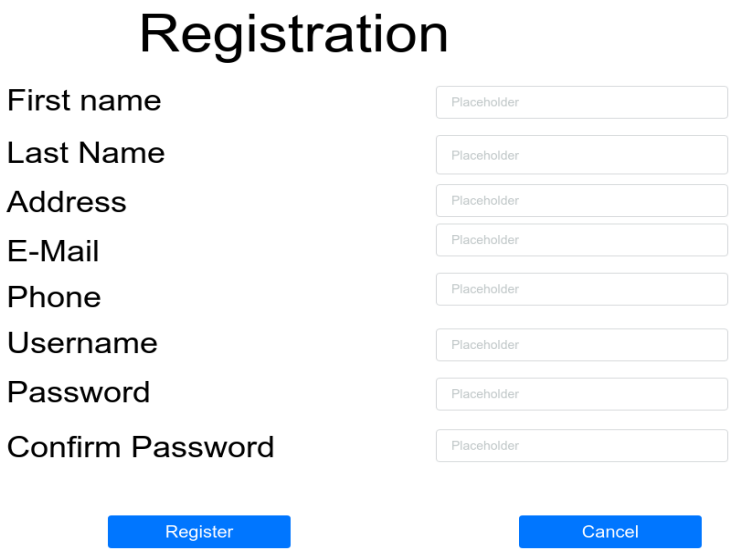

Figure 41 UI modelling ii

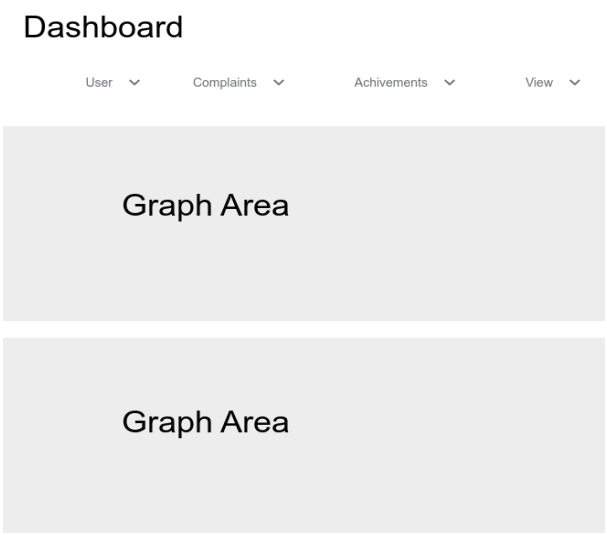

Figure 42 UI modelling iii

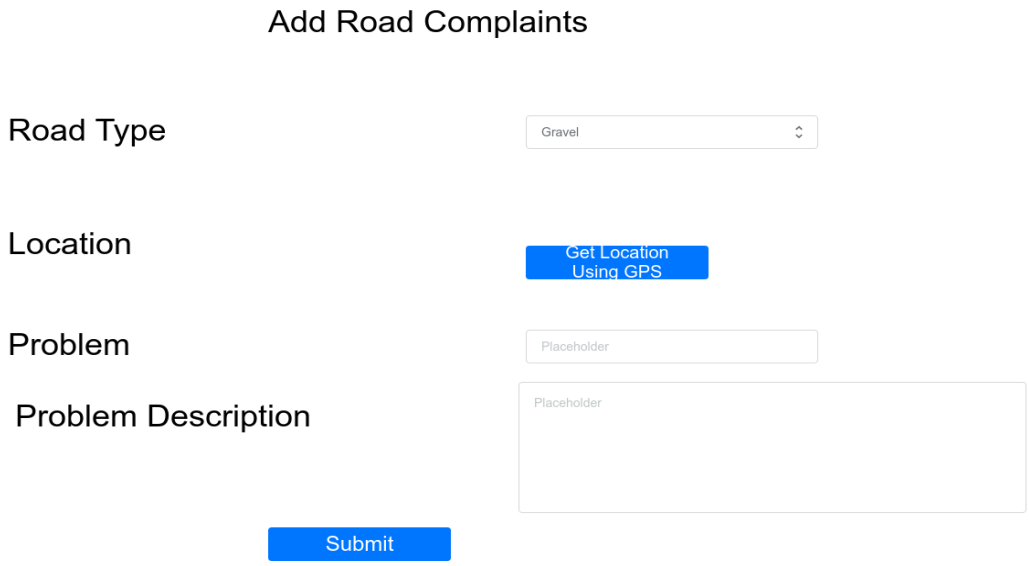

Figure 43 UI modelling iv

Chapter 4 Implementation

Implementation

Implementation is the deployment phase of design where a design is converted into a code. Implementation is carrying out the finished design structure which will later be converted into working product which fulfills the business requirements of client. Coding is the thing done during implementation. Implementation also includes development of functional and non-functional requirements of clients.

Programming Language

Android is the most used operating system of the world and almost every smartphone has android with them. Android is also open source and anybody can develop a application for them as per their requirements. As android, being the most used operating system in the world, I wrote the application use Android Studio IDE and Java as a programming language. Java is the most popular language running over billions of devices around the world. Some of the reasons for developing android application are

i) Has large userbase
ii) Has dedicated marketplace if I want to sell the product.
iii) Official support by Google as Android is owned by Google.
iv) Capable for performing almost any task on low requirement devices like android.
v) Easy to develop using IDE and readily available information on internet.

Developing Environment

Integrated Developing Environment (IDE)

For developing and testing the application Android Studio is used as IDE for java. Android Studio is easy to use and efficient for running android applications with the help of virtual device. We also can set the virtual device and its property and also can decide what version of OS does it run. As the application uses MYSQL database to store and retrieve information, WampServer is used to store and host MYSQL locally on the device. The web API that communicates with the database and returns the data to android application are stored in WWW directory of the Apache Server on drive. Sublime text is the text editor used to edit web API for the application.

Deployment Strategy

Different version the code is pushed on the GITHUB directory and also the application can be deployed using Google Play Store, official store for google. As android being huge platform, getting user informed about the application is going to be difficult. Once the application gets promoted, more downloads can be expected and the data on database increases as user submits the data using the application. The

achievements obtained by users are stored and users are encouraged to make more and more data submissions.

Risk Management

Risks are crucial factors that are to be considered before executing any project. There are as many risks as possible number may factor the runtime of the program.

Risks are to be analyzed and managed to effectively avoid risks. Identifying and avoiding such risks enhance smooth execution of the project.

The risk management table is shown below with respect to occurrence probability and impact.

Risk	Likelihood	Consequence	Impact	Action
Changes in requirements	3	4	12	Requirements are gathered as correctly as possible.
Architecture inflexibility	2	4	8	Making the system flexible with flexibility and scalability as possible
Natural Calamities	1	4	4	To keep backup in various places as possible. for example, using cloud storage.
Security Loopholes	3	4	12	Security should be given high priority. For example, using latest security tools.
Insufficient training	1	4	4	Adequate Training should be given to users.

User Training

Training user to be capable to submit the data is going to be difficult. Users with experience in using similar android applications can make the submission easily. As for disabled people accessibility options are used inside the application which delivers easy usage. As also for easy of use, easy to use UI is used and is tested for acceptance where everybody agreed that the developed UI is easy to use.

Also, for providing the information about the application inbuilt guide is made on how to use the application. Also, the user manual is made for the users which help them to learn on using the application easily.

The UI of the application with its code is shown in images below

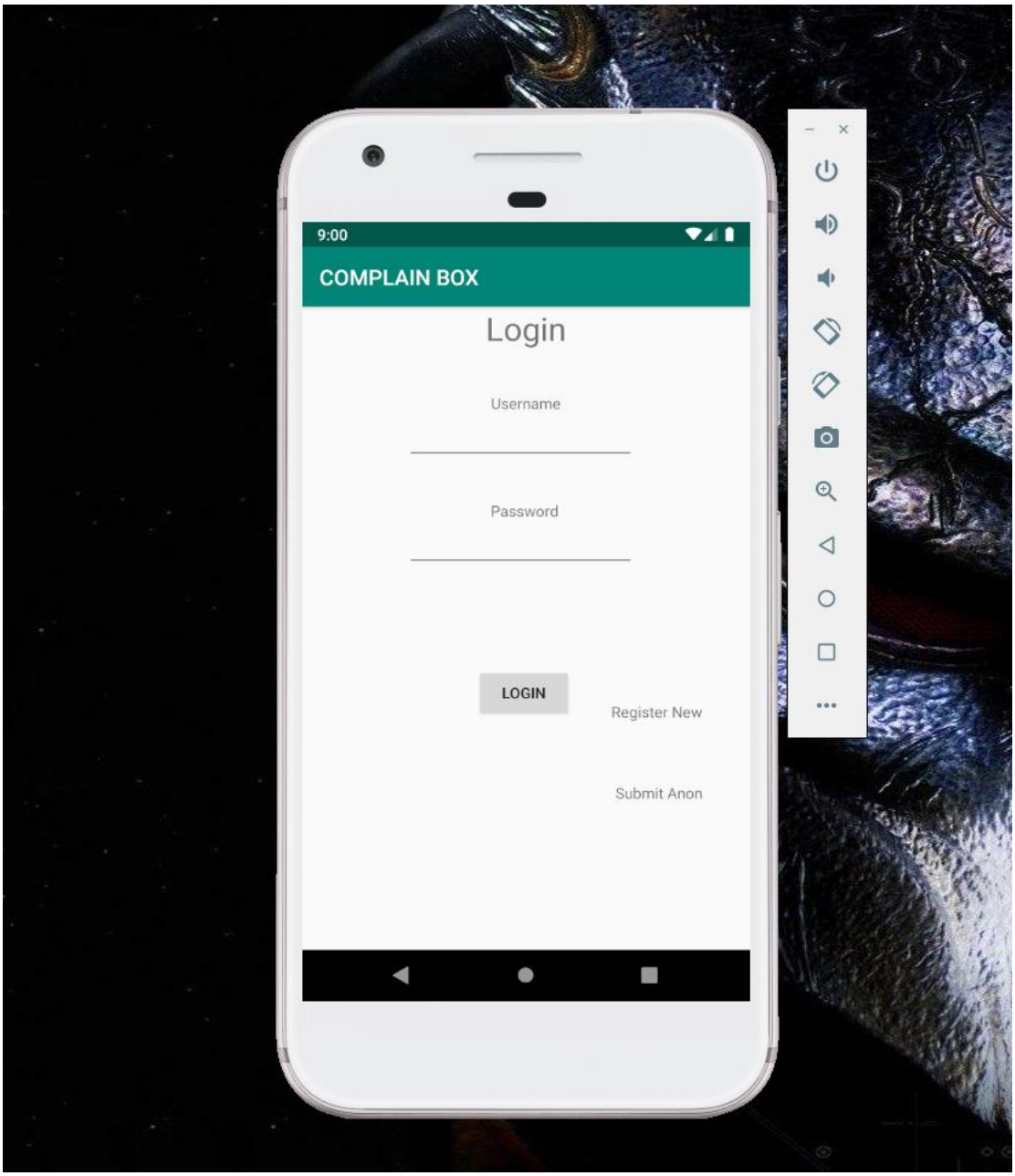

Figure 44 Login UI of application

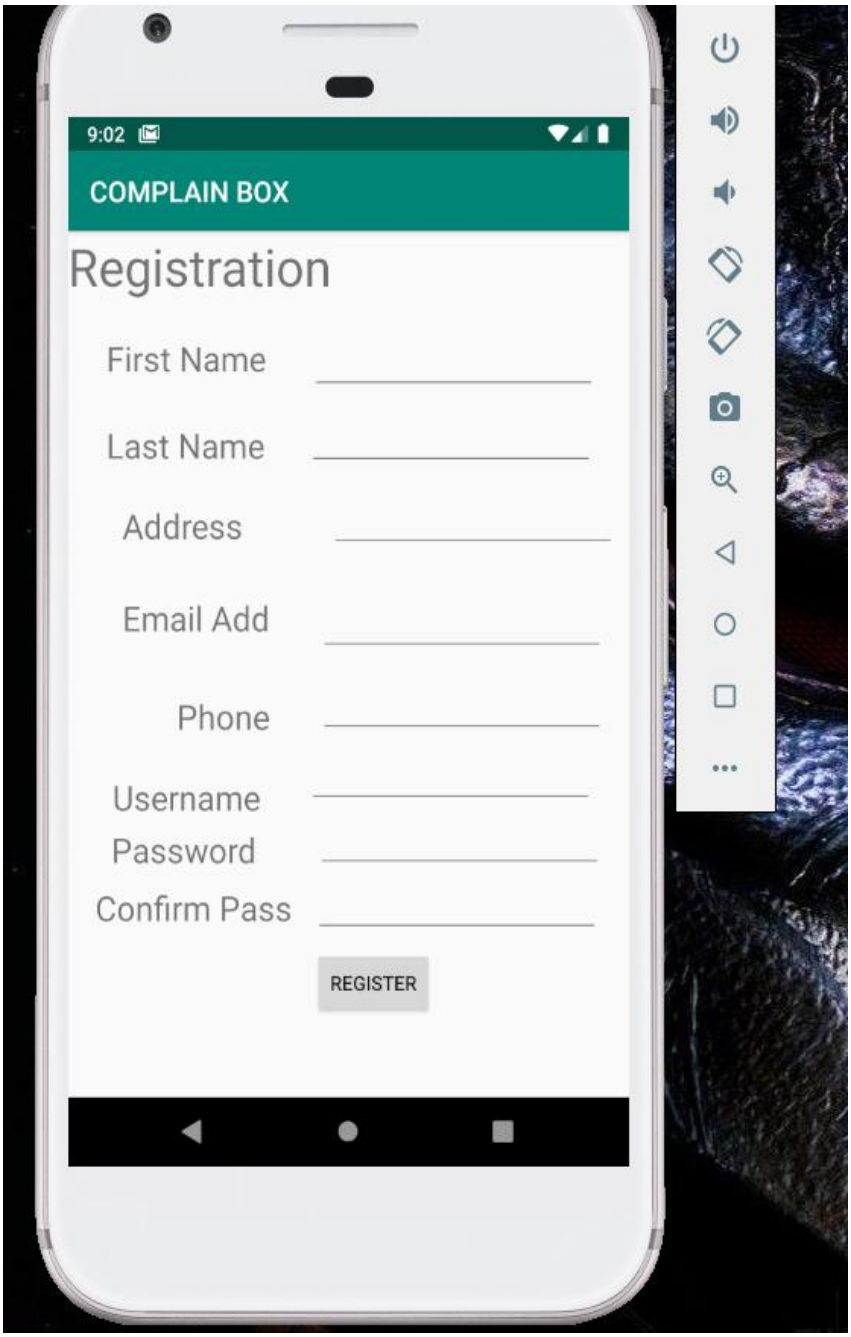

Figure 45 Registration UI

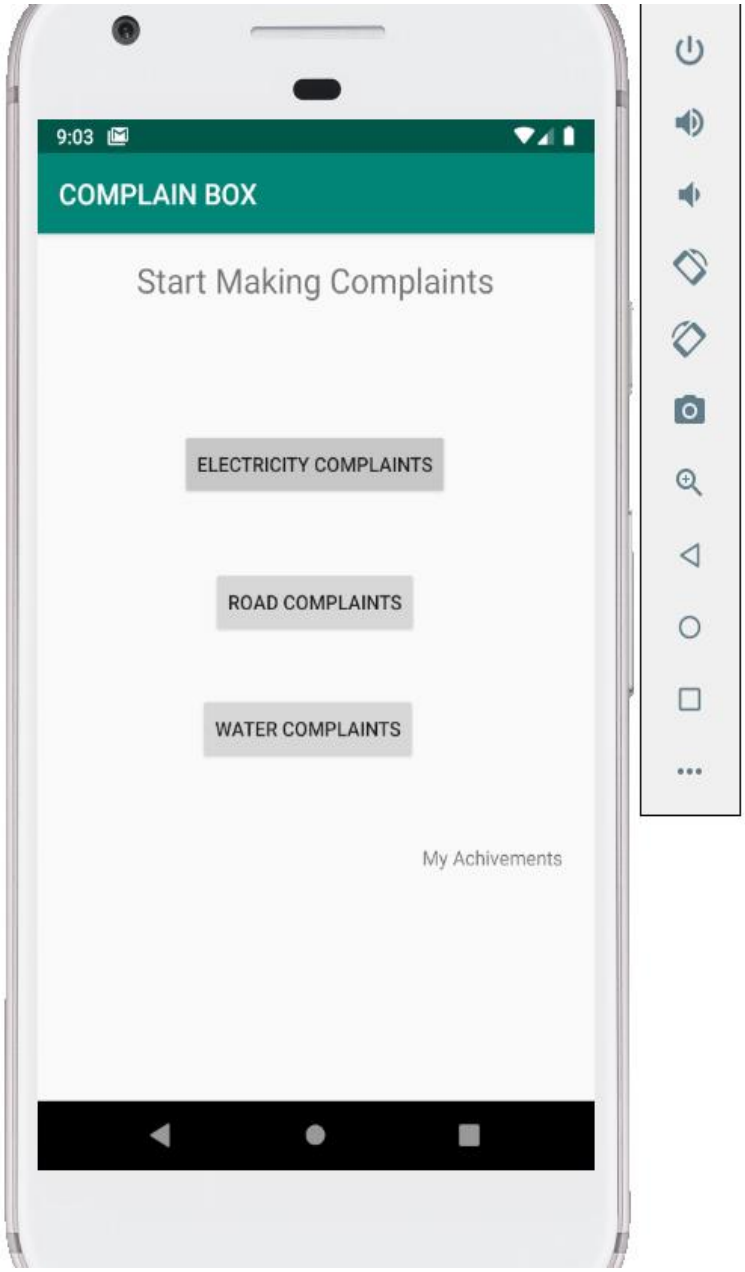

Figure 46 Dashboard UI

Figure 47 Get Location UI

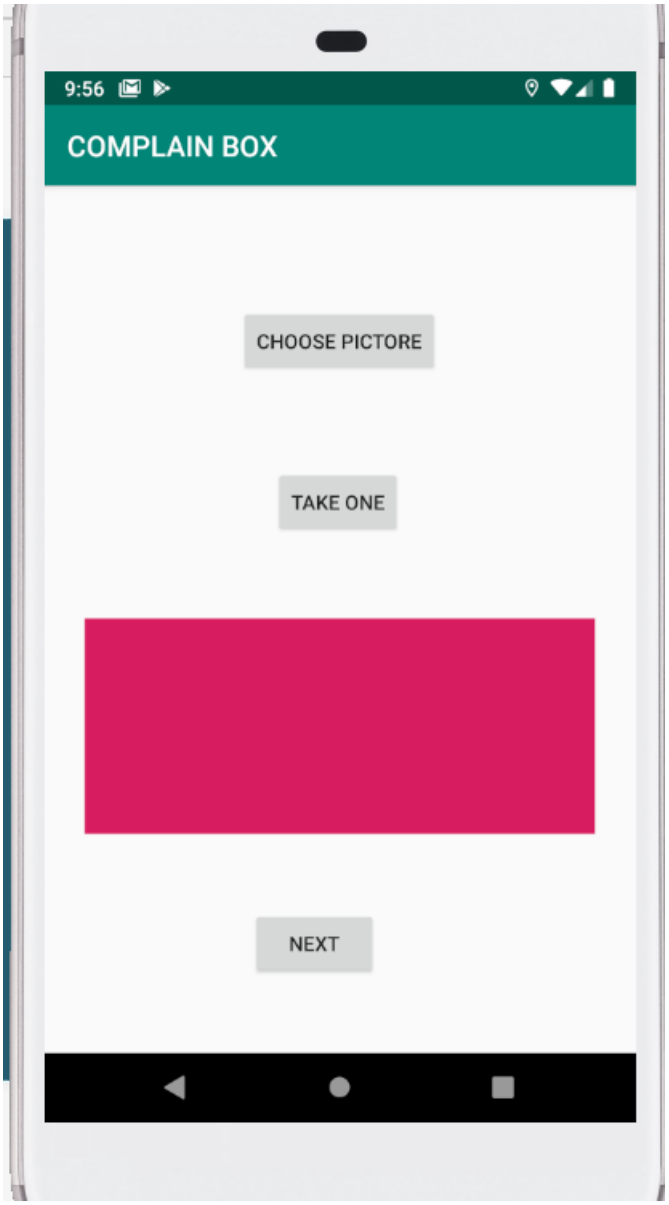

Figure 48 Choose image UI

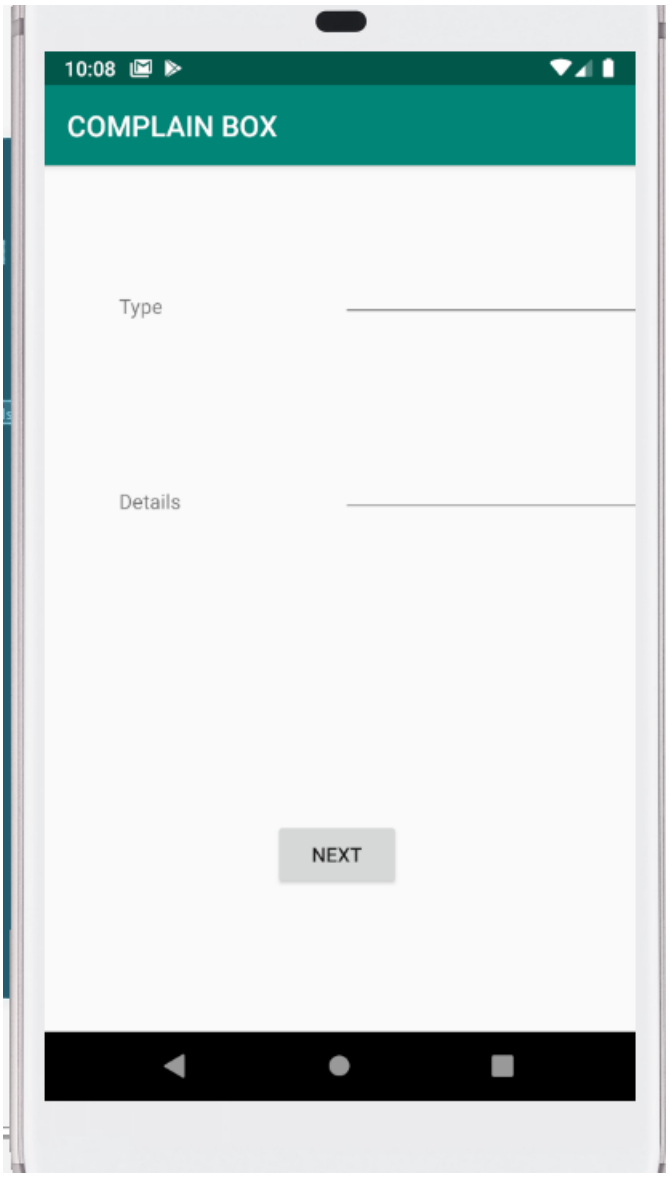

Figure 49 Insert Details UI

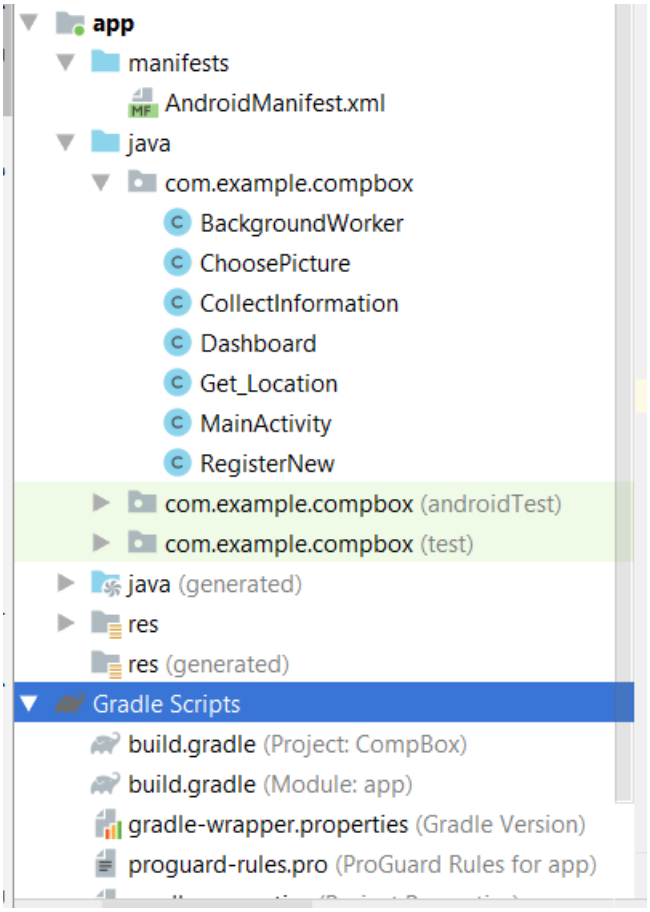

Figure 50 Program Structure and activities

AndroidManifest.xml

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

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.CAMERA" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="COMPLAIN BOX"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">
        <activity android:name=".CollectInformation"></activity>
        <activity android:name=".ChoosePicture" />
        <activity android:name=".Get_Location" />
        <activity android:name=".RegisterNew" />
        <activity android:name=".Dashboard" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Figure 51 Manifest file

Screenshot of class backgroundworker

```java
package com.example.compbox;

import ...

public class BackgroundWorker extends AsyncTask<String, Void, String> {


    Context context;
    AlertDialog ad;

    BackgroundWorker(Context ctx) {
        context = ctx;

    }

    @Override
    protected String doInBackground(String... params) {
        String type = params[0];


        if (type.equals("login")) {
            String login_url = "http://10.0.2.2/compbox/join.php";
            String Username = params[1];
            String Password = params[2];

            try {
                URL url = new URL(login_url);
                HttpURLConnection hr = (HttpURLConnection) url.openConnection();
                hr.setRequestMethod("POST");

                hr.setDoOutput(true);
                hr.setDoInput(true);
                OutputStream os = hr.getOutputStream();
                BufferedWriter br = new BufferedWriter(new OutputStreamWriter(os, charsetName: "UTF-8"));
                String PostData = URLEncoder.encode( s: "username", enc: "UTF-8") + "=" + URLEncoder.encode(Username, enc: "UTF-8") + "&"
                        + URLEncoder.encode( s: "password", enc: "UTF-8") + "=" + URLEncoder.encode(Password, enc: "UTF-8");
                br.write(PostData);
```

Figure 52 Backgroundworker code i

```
52                  br.write(PostData);
53
54                  br.flush();
55                  br.close();
56                  os.close();
57                  InputStream is = hr.getInputStream();
58                  BufferedReader buffRead = new BufferedReader(new InputStreamReader(is,  charsetName: "iso-8859-1"));
59                  String result = "";
60                  String line = "";
61                  while ((line = buffRead.readLine()) != "Null") {
62                      result += line;
63                      break;
64                  }
65                  buffRead.close();
66                  is.close();
67                  hr.disconnect();
68                  return result;
69              } catch (MalformedURLException e) {
70                  e.printStackTrace();
71              } catch (IOException e) {
72                  e.printStackTrace();
73              }
74
75
76          } else if (type.equals("register")) {
77              String reg_url = "http://10.0.2.2/compbox/registration.php";
78              String Fname = params[1];
79              String Lname = params[2];
80              String Address = params[3];
81              String Emailadd = params[4];
82              String Phonenum = params[5];
83              String Username = params[6];
84              String Password = params[7];
85              try {
86                  URL url = new URL(reg_url);
87                  HttpURLConnection hr = (HttpURLConnection) url.openConnection();
88                  hr.setRequestMethod("POST");
89
```

Figure 53 Backgroundworker code ii

```
85          try {
86              URL url = new URL(reg_url);
87              HttpURLConnection hr = (HttpURLConnection) url.openConnection();
88              hr.setRequestMethod("POST");
89
90              hr.setDoOutput(true);
91              hr.setDoInput(true);
92              OutputStream os = hr.getOutputStream();
93              BufferedWriter br = new BufferedWriter(new OutputStreamWriter(os, charsetName: "UTF-8"));
94              String PostData = URLEncoder.encode( s: "fname", enc: "UTF-8") + "=" + URLEncoder.encode(Fname, enc: "UTF-8") + "&"
95                      + URLEncoder.encode( s: "lname", enc: "UTF-8") + "=" + URLEncoder.encode(Lname, enc: "UTF-8") + "&"
96                      + URLEncoder.encode( s: "address", enc: "UTF-8") + "=" + URLEncoder.encode(Address, enc: "UTF-8") + "&"
.example.compbox (androidTest)
98                      + URLEncoder.encode( s: "email", enc: "UTF-8") + "=" + URLEncoder.encode(Emailadd, enc: "UTF-8") + "&"
99                      + URLEncoder.encode( s: "phone", enc: "UTF-8") + "=" + URLEncoder.encode(Phonenum, enc: "UTF-8") + "&"
100                     + URLEncoder.encode( s: "username", enc: "UTF-8") + "=" + URLEncoder.encode(Username, enc: "UTF-8") + "&"
101                     + URLEncoder.encode( s: "password", enc: "UTF-8") + "=" + URLEncoder.encode(Password, enc: "UTF-8");
102             br.write(PostData);
103
104
105             br.flush();
106             br.close();
107             os.close();
108
109             InputStream is = hr.getInputStream();
110             BufferedReader buffRead = new BufferedReader(new InputStreamReader(is, charsetName: "iso-8859-1"));
111             String result = "";
112             String line = "";
113             while ((line = buffRead.readLine()) != "Null") {
114                 result += line;
115                 break;
116
117             }
118             buffRead.close();
119             is.close();
120             hr.disconnect();
121             return result;
122         } catch (MalformedURLException e) {
```

Figure 54 Backgroundworker code iii

```
                return result;
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return null;
    }

    @Override
RegisterNew
    protected void onPreExecute() {
        ad = new AlertDialog.Builder(context).create();
        ad.setTitle("Login Status");
    }

    @Override
    protected void onPostExecute(String result) {
        if (result.equals("Login success")) {
            ad.setMessage(result);
            ad.show();
            Intent gotodash = new Intent(context.getApplicationContext(), Dashboard.class);
            context.startActivity(gotodash);
        } else if (result.equals("Login Failed")) {
            ad.setMessage(result);
            ad.show();
        } else if (result.equals("Register Success")) {
            ad.setMessage(result);
            ad.show();

        } else {
            ad.setMessage(result);
            ad.show();
        }
    }

    @Override
    protected void onProgressUpdate(Void... values) { super.onProgressUpdate(values); }
```

Figure 55 Backgroundworker code iv

Main Activity code

```java
package com.example.compbox;

import ...

public class MainActivity extends AppCompatActivity {

    EditText user, pass;
    Button loginbtn;
    // Context ctx;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        user = (EditText) findViewById(R.id.uname);
        pass = (EditText) findViewById(R.id.pass);
        loginbtn = (Button) findViewById(R.id.btnlogin);
    }

    public void loginmethod(View view) {

        String usernmae = user.getText().toString();
        String password = pass.getText().toString();
        String type = "login";

        BackgroundWorker bw = new BackgroundWorker( ctx: this);
        bw.execute(type, usernmae, password);


    }

    public void gotoreg(View view) {


        Intent gotodash = new Intent(this.getApplicationContext(), RegisterNew.class);
        this.startActivity(gotodash);
    }

```

Figure 56 Code main activity

```
35
36
37      }
38
39      public void gotoreg(View view) {
40
41
42          Intent gotodash = new Intent(this.getApplicationContext(), RegisterNew.class);
43          this.startActivity(gotodash);
44      }
45
46      public void submitanon(View view) {
47      }
48  }
49
```

Figure 57 Main activity ii

Registration Code

```java
package com.example.compbox;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class RegisterNew extends AppCompatActivity {
EditText Fname,Lname,address,Emailadd,Phone,Username,Pass,Conpass;
AlertDialog ad;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register_new);
        Fname=(EditText)findViewById(R.id.fname);
        Lname=(EditText)findViewById(R.id.lname);
        address=(EditText)findViewById(R.id.address);
        Emailadd=(EditText)findViewById(R.id.mailadd);
        Phone=(EditText)findViewById(R.id.phonenumber);
        Username=(EditText)findViewById(R.id.uname);
        Pass=(EditText)findViewById(R.id.password);
        Conpass=(EditText)findViewById(R.id.cpass);
    }

    public void registernew(View view) {

        String type="register";
        String firstname=Fname.getText().toString();
        String lastname=Lname.getText().toString();
        String Address=address.getText().toString();
        String emailaddress=Emailadd.getText().toString();
        String PhoneNumber=Phone.getText().toString();
        String Uname=Username.getText().toString();
        String Password=Pass.getText().toString();
        String ConPass=Conpass.getText().toString();

        if(Password.equals(ConPass)) {
```

Figure 58 Registration Code i

```
37          String ConPass=Conpass.getText().toString();
38
39          if(Password.equals(ConPass)) {
40
41
42              BackgroundWorker bw = new BackgroundWorker( ctx: this);
43              bw.execute(type, firstname, lastname,Address, emailaddress, PhoneNumber, Uname, Pas
44
45          }
46          else{
47              ad=new AlertDialog.Builder( context: this).create();
48              ad.setTitle("Alert");
49              ad.setMessage("Password Dont Match");
50              ad.show();
51
52          }
53      }
54  }
55
```

Figure 59 Registration ii

Location Manager Code

```java
package com.example.compbox;

import ...

public class Get_Location extends AppCompatActivity {
    private Button button;
    private TextView textView;
    private LocationManager locationManager;
    private LocationListener locationListener;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_get__location);
        button = (Button) findViewById(R.id.getlocation);
        textView = (TextView) findViewById(R.id.showlocation);
        locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
        locationListener = new LocationListener() {
            @Override
            public void onLocationChanged(Location location) {
                textView.append("\n" + location.getLatitude() + " " + location.getLongitude());
            }

            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {

            }

            @Override
            public void onProviderEnabled(String provider) {

            }

            @Override
            public void onProviderDisabled(String provider) {
                Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                startActivity(intent);
            }
```

Figure 60 Location Manager Code

```java
            }
        };
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && checkSelfPermission(Manifest.permis
                requestPermissions(new String[]{
                        Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.ACCESS_COARSE_LOCATION, Manifest.permission.INTERNET
                }, requestCode: 1);
                return;
            } else {
                configurebutton();
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
        switch (requestCode) {
            case 1:
                if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED)
                    configurebutton();
                return;
        }
    }

    private void configurebutton() {
        button.setOnClickListener((v) -> {
                locationManager.requestLocationUpdates( provider: "gps",  minTime: 5000,  minDistance: 1, locationListener);

        });

    }
}
```

Figure 61 Location Manager code ii

Chapter -5 Testing and Implementation

Testing and Integration

Testing is the main phase of the development cycle which determines the reliability and functionality of the application. During testing phase every functional and non-functional requirement are tested with the set of data that represent the real-life scenario of the organization. There are mainly two types of testing. They are

i) White Box testing

White box testing can be defined as the testing done by developer to check if the operations inside the application are working as intended. White box testing are done by developer which if success indicates the successful workings of methods and functionalities. There are mainly two types of white box testing

a) Unit Testing

Unit testing is the type of testing done to check the functionality of the methods which is done right after the method is created to check if the functionality of the method is correct. Unit testing is done to check every method before it goes to production where it gets real life scenario data. Unit testing helps in development of program that is reliable from the core. Performing unit tests also enable us to apply test driven development

b) Instrumented Testing

Instrumented testing is the type of testing where the UI of the system is checked by suppling set of data that user supposedly enters in the UI. Instrumented testing helps check if visual components are working as intended and perform the desired tasks.

ii) Blackbox testing

Blackbox testing is the type of Behavioral testing that is done to verify the behaviors of the application and to check if it is working as intended by developer for the users. During Blackbox testing user is supposed to check the activity of the application if it works right or not which is mostly invisible to the user using it from front end. Blackbox testing helps us identify missing or incorrect functions and also helps us find interface errors.

iii) Integration Testing

Integration testing is the type of testing where individual units or components of the system are combined and tested as group. Integration helps find us faults between two units or objects communicating with each other to perform a task combinedly.

iv) User Acceptance Training

User acceptance testing is the type of testing that helps us verify if users will actually like the UI of the product. User acceptance testing helps us identify ambiguous components of the system and modify it so every user can use it easily and comfortably.

v) Regression Testing

Regression testing is the type of testing where functional and non-functional features added later to fulfill the system requirements has not affected the code written before. Regression testing helps find out new updates has not affected program adversely.

Integration

Integration can be defined as the process of putting tested application in the live environment and making it available to end users. After testing it is essential to find out if product got by end users is what they want and fulfills every functional and non-functional requirements of the project. Tannings to the end user is also provided in the integration testing so that user can use the product efficiently and reliably.

Chapter 5

Maintenance

Maintenance is the final phase of DSDM where system is monitored for anomalies. Maintenance comes after testing and integration. Maintenance helps admin to identify the performance of software after deployment to correctly resolve issues that arises.

As maintaining software becomes costly after most of the business requirements are met the phases of SDLC would end. If maintenance is required for the product after maintenance the SDLC returns back to planning phase to modify the product and keeps on going forever.

Maintenance helps improve the project to fit for what users actually desire. Maintenance phase also can be used to get feedbacks from the user which can be used to improve the product to its best.

Conclusion

Hence the specifications of the project are met to develop the complain Box, an android application that can be used to report problems people face day to day in the life. Every phase of development lifecycle are done properly which helps make us a reliable product. In first phase the viability of application is analyzed. The second phase if the project is viable helps design the structure of application which later can be used to code the program. Implementation or coding is the third phase where every functional and non-functional requirement are met. The fifth stage is testing where product is tested through to test if it works as intended. The last phase is maintenance where product is monitored and user suggestions are taken which helps improve the product.

The project developed above ‘Complain Box’ fully meets the requirements and tasks asked to perform. The data collected from Complain Box can be used to make pressure groups to the government.