

Society Management System

A Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Chaitanya Vijay Mane

Roll Number – 513

Under the esteemed guidance of

Prof. Randeep Singh Ghai

Assistant Professor



DEPARTMENT OF INFORMATION TECHNOLOGY

GURU NANAK KHALSA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

(Autonomous)

MATUNGA, MUMBAI – 400 019

MAHARASHTRA

AY 2019 – 2020

GURU NANAK KHALSA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

(Autonomous)

MATUNGA, MUMBAI, MAHARASHTRA – 400 019

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project entitled, "**Society Management System**", is bonafied work of **Chaitanya Vijay Mane** bearing Seat No: **513** submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide

Coordinator

External Examiner

Date:

College Seal

Abstract / Synopsis

Society Management System

Introduction

A housing society management and billing project that effectively manages and handles all the functioning of a cooperative housing society.

This system application can store the data of various flat owners and their family members along with their images.

The system also maintains and calculates the society maintenance as well as parking, cultural funds, emergency funds and other charges and adds them automatically in individual flat bill.

The system needs an administrator to input various flat owner data and billing amounts into it. The rest of the work is done by the system on its own. The system consists of automatic bill generation facility.

It calculates various associated costs, adds them up and provides a bill accordingly. System has to send email notification for maintenance details and water bill due date.

Secretary can create event and add flat owners to this event. Also he can be able to modify event or cancel event. Event notification should be sent to email.

Features:-

1.Secretary Account :- Login & Secretary can view and show society member details.

2.Member's Login :- Login & Member of the society can view maintenance and summary of the bill / bill information & add/update is own information.

3.Society Bill Amounts :- The secretary calculates various billing data including, maintenance, water , parking, event fund for flat owner.

4.Upload Documents :- The society member can browse as well as upload their documents in the Image format.

5.Flat/User/Bill Details :- Secretary can view all the details that are been inserted by the society member's.

6.Feedback/complaints : Member of the society can post feedback / complaints to the Secretary.

7.Send Alert Notification's : This alert notifications will be send to all the society member's.(Ambulance, Fire Brigade, Police, Less Water Supply).

8.Notice Boards : Secretary can send an notification's to all the society member's.(Events - Festivals, Nationals Days, Birthday's , Various Programs Activities).

Software Requirements:-

- 1.Windows 7,Windows 8,Windows 10.
- 2.Android Studio 4.0
- 3.Java 8 Library Language(API).

Hardware Components:-

- 1.Processor - i3 & i5
- 2.Hard Disk - 10GB
- 3.Memory - 4GB/8GB RAM

Advantages:-

- 1.It helps the society secretary to handle and manage flat owners data.
- 2.It helps them manage society funds.
- 3.It brings transparency and efficiency in the working of housing societies.

Disadvantages:-

- 1.This system can only handle single society.
- 2.This system does not include bank payment, dd, cheque status.

Applications:-

- 1.This application can only be used in housing societies.
- 2.This application mainly featured for day to day notifications.

ACKNOWLEDGEMENT

I would like to express my thanks to the people who have helped me most throughout my project. I am grateful to my Prof. **Randeep Singh Ghai** for nonstop support for the project. I can't say thank you enough for him tremendous support and help.

I owe my deep gratitude to our HOD of Information Technology Department **Mrs. Jasbir Kaur** who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

At last but not the least I want to thank all of my friends who helped/treasured me out in completing the project, where they all exchanged their own interesting ideas, thoughts and made this possible to complete my project with all accurate information. I wish to thank my parents for their personal support or attention who inspired/encouraged me to go my own way.

DECLARATION

I hereby declare that the project entitled, “**Society Management System**” done at **Guru Nanak Khalsa College**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Chaitanya Vijay Mane

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Background

1.2 Objectives

1.3 Purpose, Scope, and Applicability

1.3.1 Purpose

1.3.2 Scope

1.3.3 Applicability

CHAPTER 2: SURVEY OF TECHNOLOGIES

2.1 Introduction / Background / Context

2.2 Existing System

2.3 Existing Solutions in the Market

2.4 Proposed System

TABLE OF CONTENTS

CHAPTER 3: REQUIREMENTS/ANALYSIS

3.1 Problem Definition

3.2 Requirements Specification

3.3 Planning and Scheduling

3.3.1 GANTT Chart

3.3.2 WBS Chart

3.4 Software and Hardware Requirements

3.5 Preliminary Product Description

3.6 Conceptual Models

3.6.1 Class Diagrams

CHAPTER 4: SYSTEM DESIGN

4.1 Basic Modules

4.2 Procedural Design

4.2.1 Use Case Diagram

4.2.2 Activity Diagram

4.3 User Interface Design

4.4 Test Cases Design

TABLE OF CONTENTS

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation Approaches

5.2 Code Details

5.3 Testing Approach

5.3.1 Unit Testing

5.3.2 Integration

CHAPTER 6: RESULTS AND DISCUSSION

CHAPTER 7: CONCLUSIONS

CHAPTER 8: REFERENCES

Software and Broad Areas of Application

Front End Interaction-	Android Application
Back End Database-	Google Firebase
Languages used-	XML , Java
Tools used-	Android Studio
Technologies used-	Microsoft Windows 10

CHAPTER 1 : INTRODUCTION

1.1 Background

This system is developed to manage day-to-day activities of any co-operative housing society. Generally, in society all the work is manually.

Contact no of members are noted on the papers. As all work is done on paper so it is very difficult to manage and keep track of all the work expenses in the society.

This society co operative society management system will computerize all day to day operation in the society. There is no automated system for doing all the things that generally happen in society, so that members can come to know what is happening in society.

This system is also Raise invoices against all flats or select flats based on area or based on a fixed amount.

1.2 Objectives

The purpose behind a cooperative society is to provide goods and services to the consumers at a reasonable cost with the aim to encourage them towards productivity, savings, investments and equal distribution of surplus.

Co-operative societies have a lot of objectives:-

- 1.To provide services and not run to earn any profit.
- 2.To help those who are in need, not compete
- 3.To undertake all the activities with keeping in mind the aim which is, the benefit of others.

1.3 Purpose , Scope and Applicability

1.3.1 Purpose:-

The Managing Committee of a housing co-operative is its nerve centre, steering the society towards a wholesome, efficient and harmonious existence.

Whether it is financial management, member grievances or day-to-day affairs, the MC has to run like a well-oiled machine to create the best living conditions for the residents.

1.3.2 Scope:-

The information regarding adopting procedure for solving the complaints of the citizens and members of the society and similarly to take decision on that is given in this manual.

In this, the procedure to take decision on various subject on the society level according to the co-operative Act, Rules, Bye-Laws and Government orders/Directives issued on the Government level from time to time.

1.3.3 Applicability:-

Applicability of GST on housing societies GST is applicable to the societies as the society is offering services to its members in respect of collection of common charges and payment to contractors/ different agencies.

The service it was offering was earlier falling in the category of club or association service.

CHAPTER 2 : SURVEY OF TECHNOLOGIES

2.1 Introduction

This system is developed to manage day-to-day activities of any co-operative housing society. Generally , in society all the work is manually. Contact no of members are noted on the papers.

As all work is done on paper so it is very difficult to manage and keep track of all the work expenses in the society. This society co operative society management system will computerize all day to day operation in the society.

This System also keep the details of service providers who provide different services like Security, Housekeeping, Pest Control, Equipment Maintenance etc. There is no automated system for doing all the things that generally happen in society, so that members can come to know what is happening in society.

This system is also Raise invoices against all flats or select flats based on area or based on a fixed amount. Members receive email and SMS whenever an invoice is raised against their flat. This system of maintaining a society is made in such a way, so that the most common problem faced in residential societies are solved.

Using this system Advance payments or partial payments can be easily noted and tracked against any flat. They require the co-ordination among the respective management societies coupled with the vendors which provide these services so that the appropriate convenience can be provided.

Using this system admin can select and send email/SMS reminders to defaulters on the payments to be made.

2.2 Existing System

In current housing society management system all society works are done manually. Society chairman keep all the society expenses report on paper file.

If some time this data is needed then retrieving this data is very slow and tiresome work. Society expenses and earning data is stored in various registers so the linking between it becomes difficult.

Voting is conducted in the society for various designations like secretary, treasurer, chairman, etc. members need to be present on the site for voting.

Due to some reasons some members cannot be present & cast vote. This problem need to be solved so that all society member can take part in the voting. Generating bill and sending it to each society member is also a tiresome work.

Sometime members did not get bills and it is lost. Payment collection is also big task in society. Watchman has to go each flat to collect the payments. In current system all work is done manually so each work takes more time.

Sending notice or other message to society members is also a big task. So an system is needed that can solved all above problems.

2.3 Existing solution in the market

Survey in the Market:-

With the increase in growth of real estate development, it becomes necessary for the builders or manager of the societies to maintain the resident's details and their contribution towards the society.

It would be impossible to carry out all the process of maintaining society data on paper. Hence there emerges a need of automating these operations on a data processor.

Therefore, we have engendered suitable software that automates this process of managing data. Usually, the data that all housing society requires are every resident's detail, their maintenance reports and bill calculation, and their additional charges.

Our software incorporates all these functions along with some additional features. Housing Society System is a software developed by our experts that serves the best way of handling society data by automating all the accounting activities.

This software will be under control of secretary of the society. He can maintain records of all the members' information like personal details, automobile and parking details, maintenance charges, membership offers and many more.

The purpose of the software is to generate bill reports for each resident according to the service usage by family members.

One of the Solution in the Market:-

CO-OP HOUSING SOCIETY APPLICATION-

A co-op is a multi-unit property owned by a company, usually a Limited-Liability Company (LLC). Individuals who purchase a unit in the building become shareholders in the holding company, with an exclusive leasehold on their particular unit.

That means your purchase is financed with a home loan instead of a residential mortgage since you're buying shares in a company instead of real property.

The property is managed by a co-op board comprised of a limited number of shareholders in the building, who make decisions for the benefit of the corporation (other residents). Legally, it's a different arrangement from a condominium, in which case each unit has a different owner.

In a co-op, the corporation usually assumes maintenance responsibilities that fall outside individuals' units, including property management and maintenance staff, mowing and snow removal, insurance, taxes, mortgage (if any), repairs, and major systems like HVAC, electric, and plumbing.

2.4 Proposed System

Our proposed system is cloud based and has automated functionality for generating monthly maintenance and member can view their bill status on their account.

Housing society management system allows societies to create user accounts for each member in two ways. The main functionality of our project is that, there is an online bill payment system, online voting system for different society positions like Secretary, Chairman, Treasurer Etc.

When member account is created, the system sends an automated email to the member whose account is created. This System allows members to login with their own account and get updated with society happenings. User can also track payments made against each bill. He can generate report of all payments any time.

Using the housing society management system, treasurer can publish expense statements to notice board and to owners at the click of a button. This software concentrates mainly on generation of bills whereas our system consists of various other features of society which are exact replica of the real happenings in the society.

On successful payment of dues, members receive instant e-receipts which can be saved or printed for future reference. Members can view/download receipts for all past payments online at any time. Notices can be posted to everyone in the complex.

All notices will be visible on the online Notice Board till the expiry date of the notice. Administrators can send instant messages by Email to all members. Admin can easily generate meeting notice, agenda, minutes of the meeting, note the attendances and more online.

CHAPTER 3 : REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

Housing Problem. one of the social problems bred by capitalism, manifested as a particular form of housing need. With the growth of the urban population and the transformation of a dwelling into a commodity, there is a sharp deterioration in the working people's living conditions and a huge rise in apartment rent.

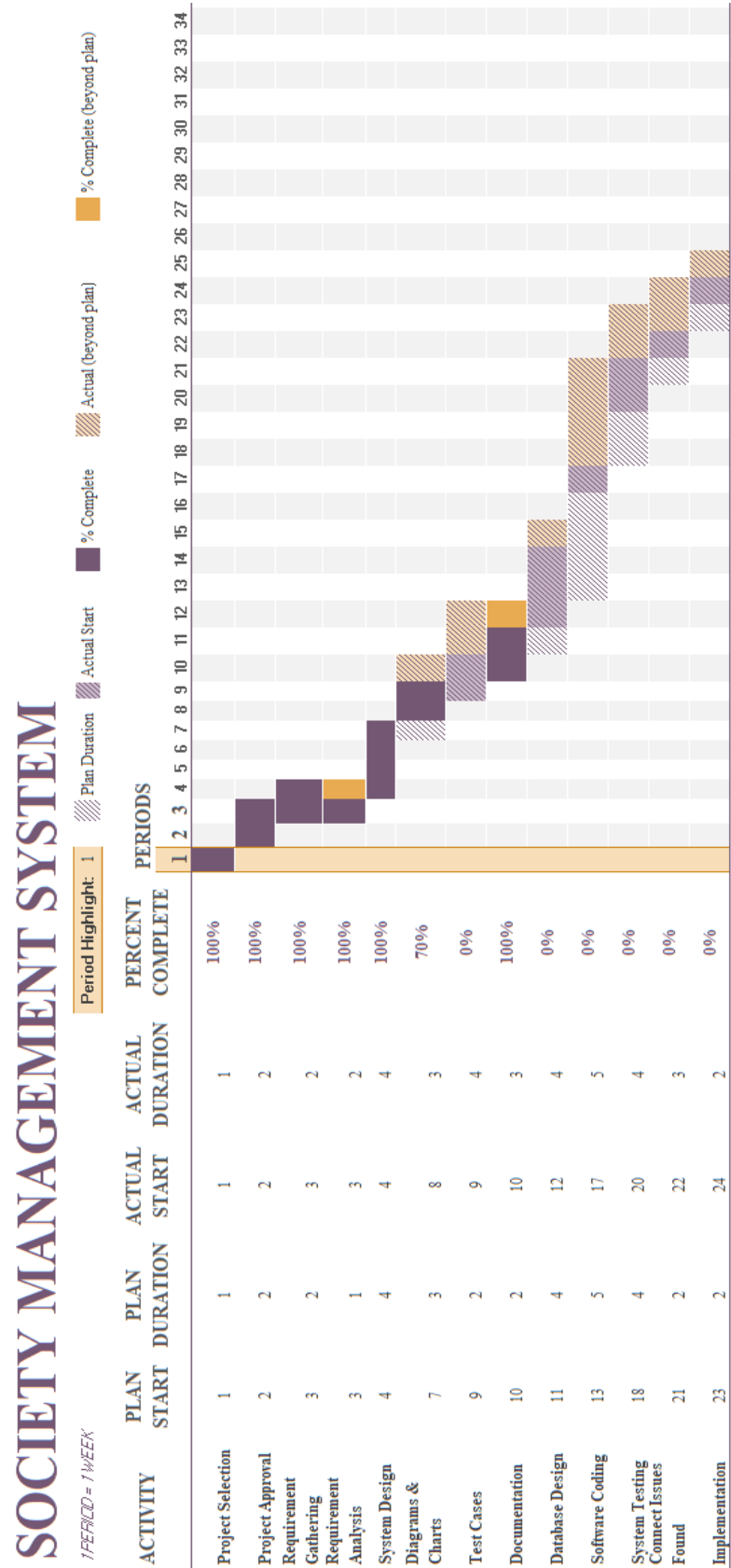
The Housing Problem At one time a home was defined as a place where a family resided, but as American society changed, so did the definition of home. A home is now considered a place where one or more people live, a private place to which they have legal right and where strangers may be excluded.

3.2 Requirements Specification

The basic requirements for Registration of Co-operative Housing Society normally is not known to the flat purchasers. It is here that apart from the statutory obligations cast upon the builder, the builder as a friend, philosopher and guide of promoters helps in forming a Co-operative Housing Society. It is also compulsory to make amendments in it as per requirement. Thereafter necessity felt to form Housing Federation. The provision of Federation of Societies has been made in the Maharashtra Co-operative Society Act 1960 and Rule 1961. ... special general body meeting of the housing society. To give guidance for preparing of income-expenditure statement, reconciliation statement.

3.3 Planning and Scheduling

3.3.1 GANTT Chart:-



Description of GANTT Chart:-

A Gantt chart makes it simple to create, view and monitor project activities and tasks over a given time. Typically, each activity, process or task in a Gantt chart is represented by a horizontal bar scaled parallel to a calendar and/or dates.

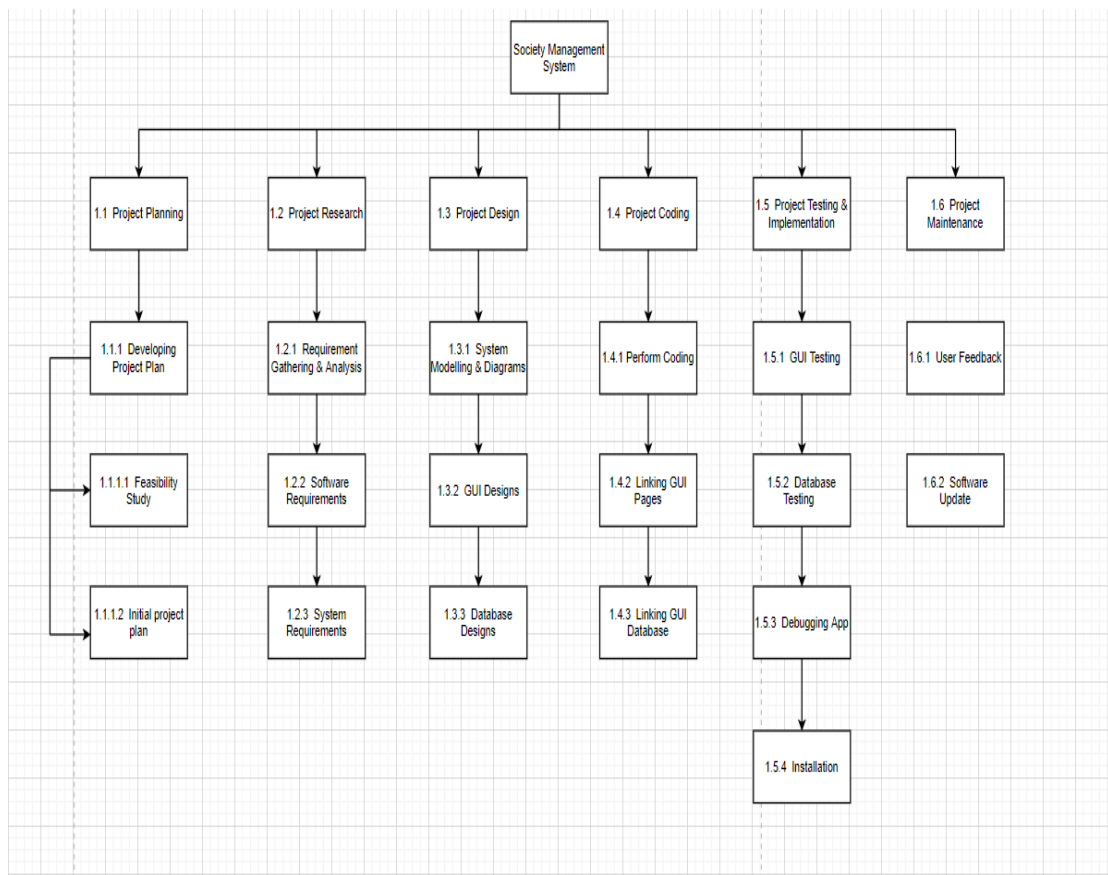
The start and end of each bar represent the start and the end of that particular activity. Gantt charts help in quickly understanding the different tasks in a project, their schedule (start and end date) and any overlapping tasks or activities.

A Gantt chart is constructed with a horizontal axis representing the total time span of the project, broken down into increments (for example, days, weeks, or months) and a vertical axis representing the tasks that make up the project (for example, if the project is outfitting your computer with new software, the major tasks involved might be: conduct research, choose software, install software).

Horizontal bars of varying lengths represent the sequences, timing, and time span for each task. Using the same example, you would put "conduct research" at the top of the vertical axis and draw a bar on the graph that represents the amount of time you expect to spend on the research, and then enter the other tasks below the first one and representative bars at the points in time when you expect to undertake them.

The bar spans may overlap, as, for example, you may conduct research and choose software during the same time span. As the project progresses, secondary bars, arrowheads, or darkened bars may be added to indicate completed tasks, or the portions of tasks that have been completed. A vertical line is used to represent the report date.

3.3.2 WBS Chart:-



Description of WBS Chart:-

The Work Breakdown Structure (WBS) is developed to establish a common understanding of project scope. It is a hierarchical description of the work that must be done to complete the deliverables of a project. Each descending level in the WBS represents an increasingly detailed description of the project deliverables.

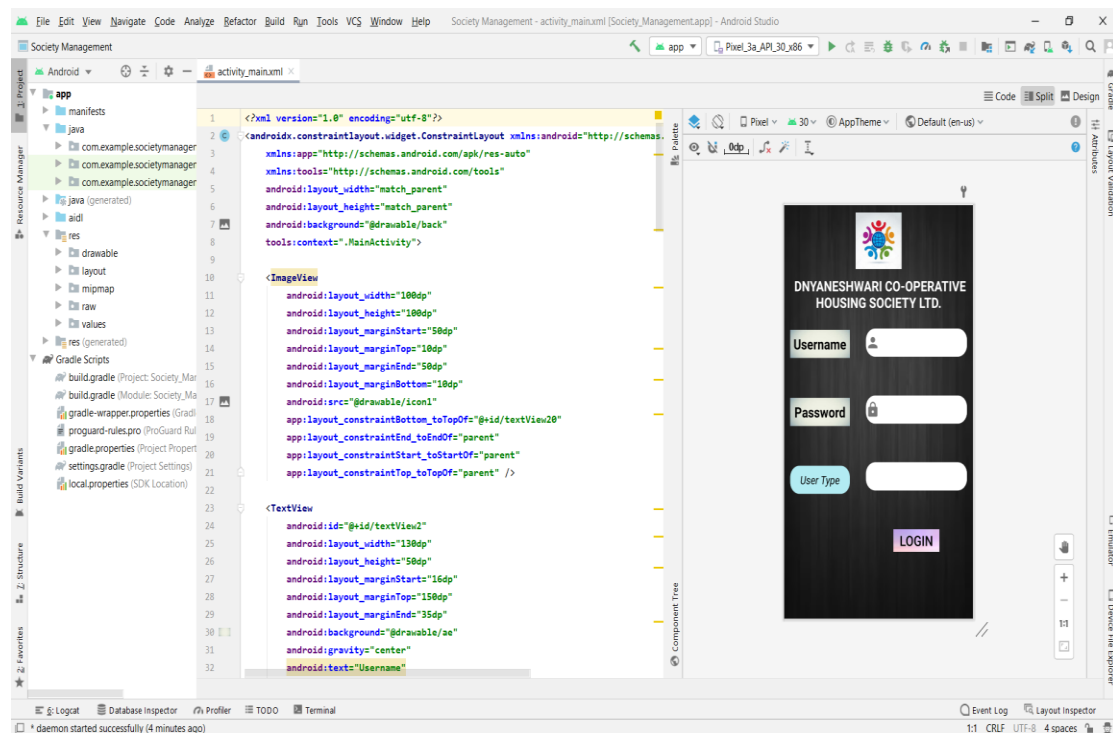
The first two levels of the WBS (the root node and Level 2) define a set of planned outcomes that collectively and exclusively represent 100% of the project scope. At each subsequent level, the children of a parent node collectively and exclusively represent 100% of the scope of their parent node.

3.4 Software and Hardware Requirements

Software Requirements :-

Android Studio-

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.



Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug. Create rich and connected apps. Android Studio is simply where you will write, edit and save your projects and the files that comprise said projects.

Hardware Requirements :-

Windows 10-

For using an Android emulator, you'll most likely need the 64-bit version of Windows 10. Most android emulators require this as a minimum and don't work with 32-bit Windows.

Processor-

Most android emulators require an Intel processor, but some also have support for AMD processors. The Intel processor should have support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality.

Memory-

You'll need at least 4 GB RAM to use an Android emulator. For some emulators, the minimum memory requirement may be higher. It's important to note that 2 GB of disk storage would not make up for memory as that's a requirement.

Storage-

Every emulator has different storage requirements, but you should keep at least 5 GB in mind. You'll need more of it once you start using the emulator and download all your games.

Games will also get stored on the hard drive, so make sure you have sufficient space. It's always better to have more space than required.

Screen Resolution-

Assuming you're downloading an Android emulator for playing games, you would want a high-definition screen resolution. However, for most emulators, the minimum screen resolution is 1280 x 800 pixels. This is one of those things that can enhance the quality of graphics.

Graphics Driver-

Make sure to update your graphics driver on Windows 10, as that's recommended for using Android emulators. An old graphics driver may not fully support or optimize graphics for Android games.

Wrap Up-

Meeting the minimum system requirements is necessary to install Android apps on PC. While the minimum requirements will allow you to install it, you better brace for some lag. For absolute best performance, go with the recommended requirements or better.

The good thing is most Windows 10 PC's today meet these minimum requirements, so you will not have to make hardware upgrades for simply downloading an Android emulator. Even if you have to, it will be worth it.

3.5 Preliminary Product Description

The Society Management System is a smart phone application. This application is free from bugging problem and does not work slow on heavy load.

Free from hacking and other problem causing factor, this software is used to store the information about the vehicle, persons, entry-exit, bank account and many other thing.

The basic concept of this society system, we create global smart based application in Android Studio with Java language to manage Society with House and Owner of house detail.

We provide platform to register society to our system, and there are many houses according to each society. Each house allocated to owner of house, all the owner are members of our project.

All members can make online complain related home to society management. We have developed the system for society member they can make complain form any where any time and we resolve the Complain as soon as possible.

In this system people can easily find address of the house by providing member name. In this society helping system there are two types one is Admin and Owner of house.

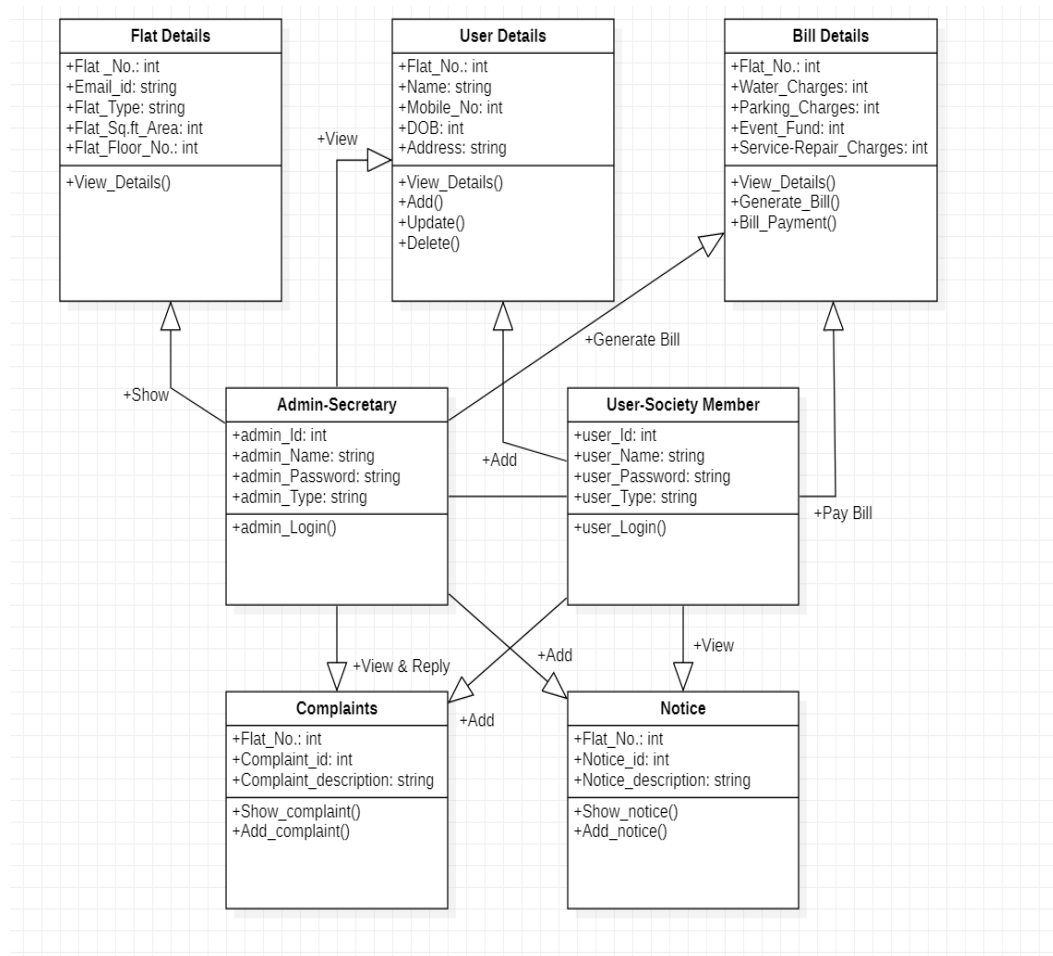
Applications:

This application can only be used in housing societies.

This application mainly featured for day to day notifications.

3.6 Conceptual Models

3.6.1 Class Diagram:-



Description of Class Diagram :-

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object - oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

CHAPTER 4 : SYSTEM DESIGN

4.1 Basic Modules

Splash Screen:-

The splash screen is the first screen that appears in front of the user when they interact with your application so to make an impactful impression you can create an Animated Splash Screen.

Login:-

Admin as well as User will be able to login through unique username and password.

Home Page:-

Admin and User will see their own dashboard page in which they can check flat details, user details, bill details as well as complaints and notices.

Society bill amounts:-

The secretary calculates various billing data including, maintenance, water , parking, event fund for flat owner.

Flat/User/Bill Details :

Secretary can view all the details that are been inserted by the society member's.

Feedback/complaints :

Member of the society can post feedback / complaints to the Secretary.

Send Alert Notification's :

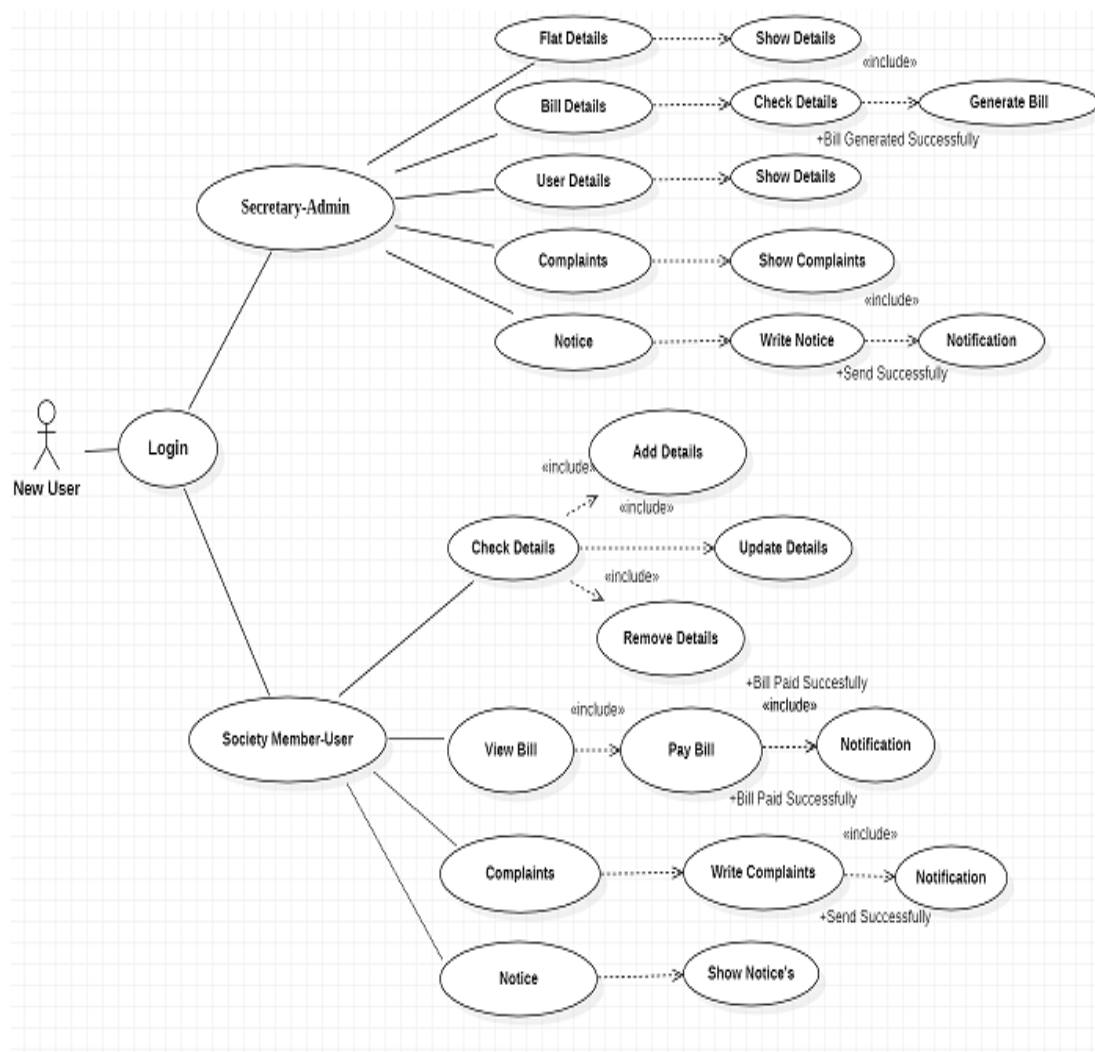
This alert notifications will be send to all the society member's.

Notice Boards :

Secretary can send an notification's to all the society member's.

4.2 Procedural Design

4.2.1 Use Case Diagram:-

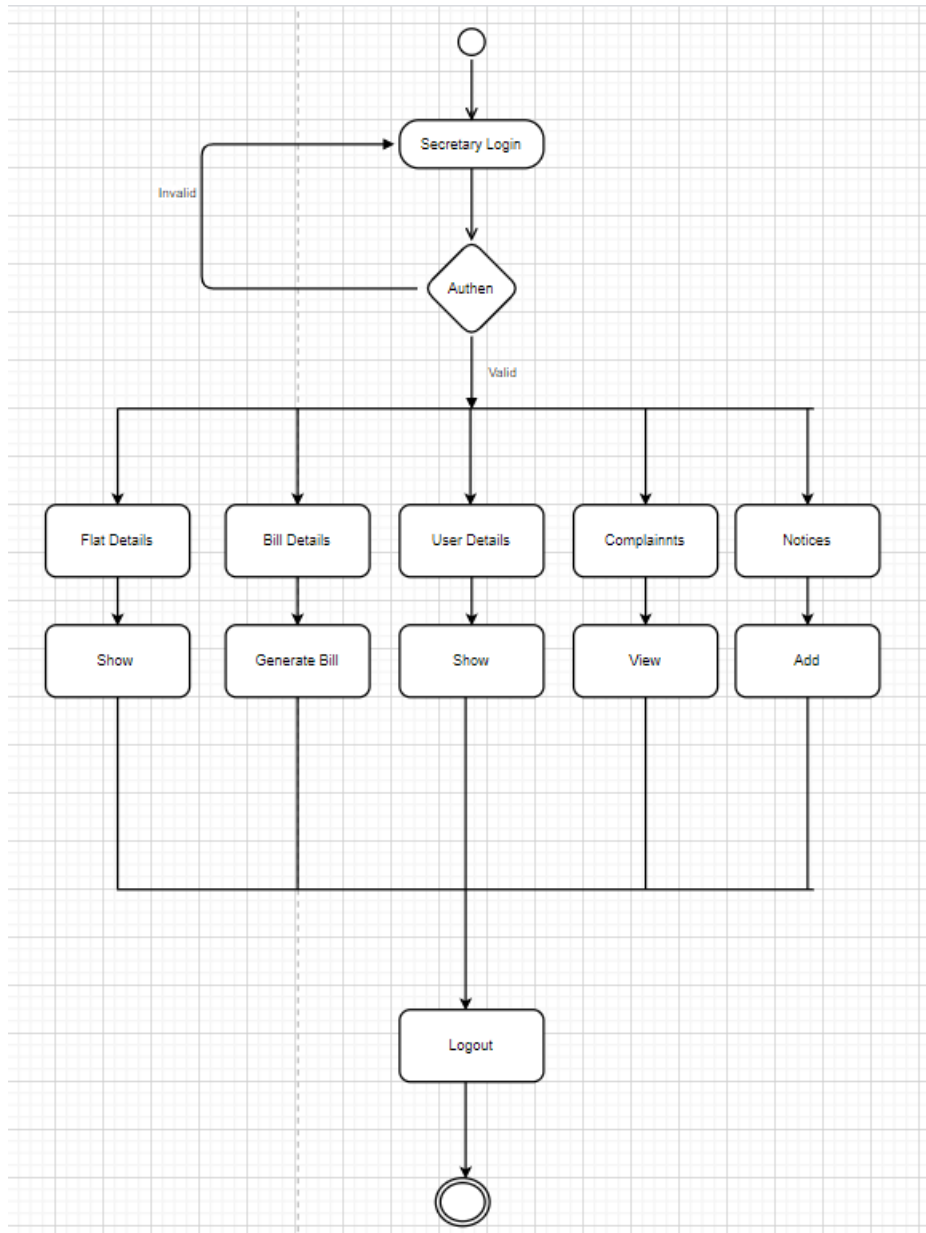


Description of Use Case Diagram :-

Use case diagrams referred as a Behavior model or diagram. It simply describes and displays the relation or interaction between the users or customers and providers of application service or the system. It describes different actions that a system performs in collaboration to achieve something with one or more users of the system. Use case diagram is used a lot nowadays to manage the system.

4.2.2 Activity Diagram:-

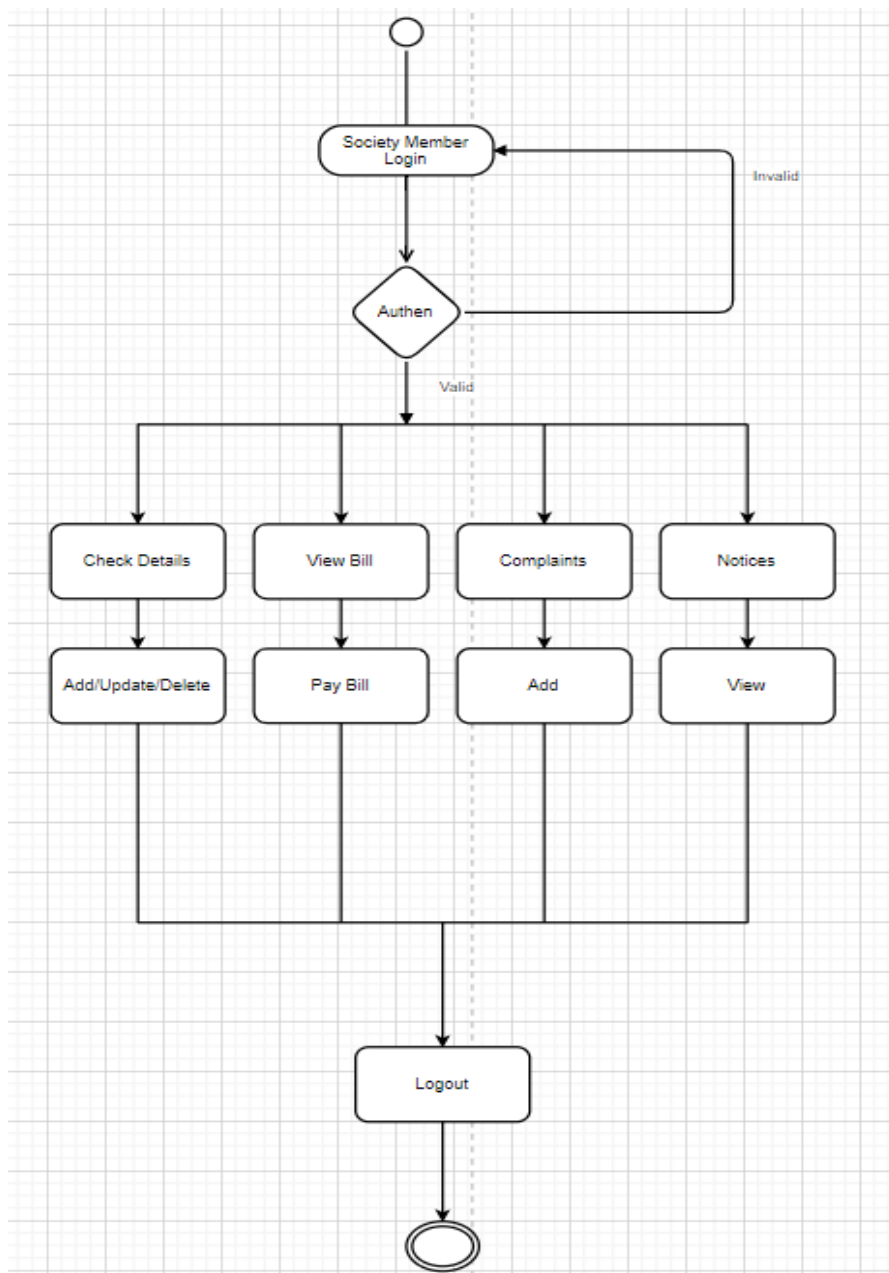
Admin-



Description of Activity Diagram(Admin)-

The admin login is done through authentication. It captures the dynamic behavior of the system. Other five diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

User-



Description of Activity Diagram(Admin)-

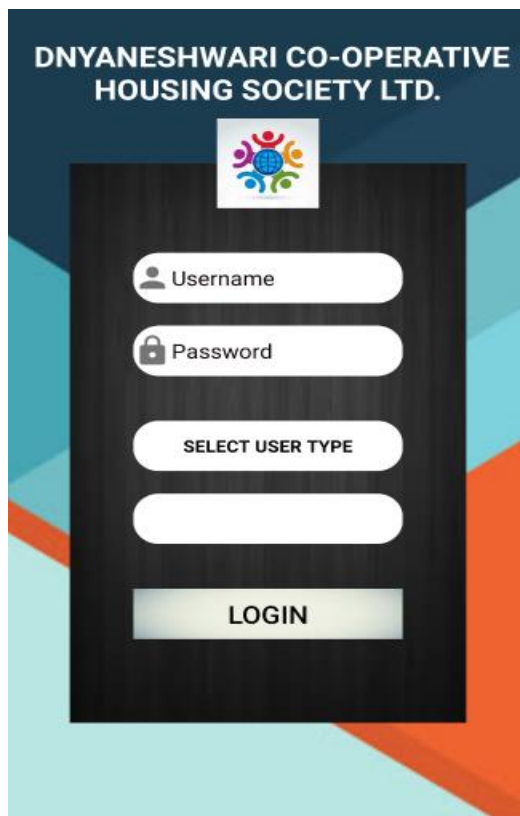
The user login is done through authentication. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

4.3 User Interface Design

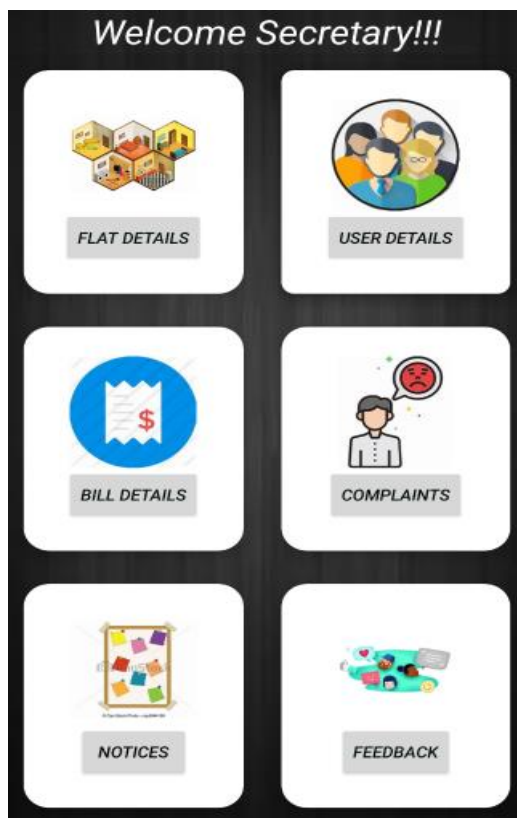
1. Animated Splash Screen-



2. Login Screen-



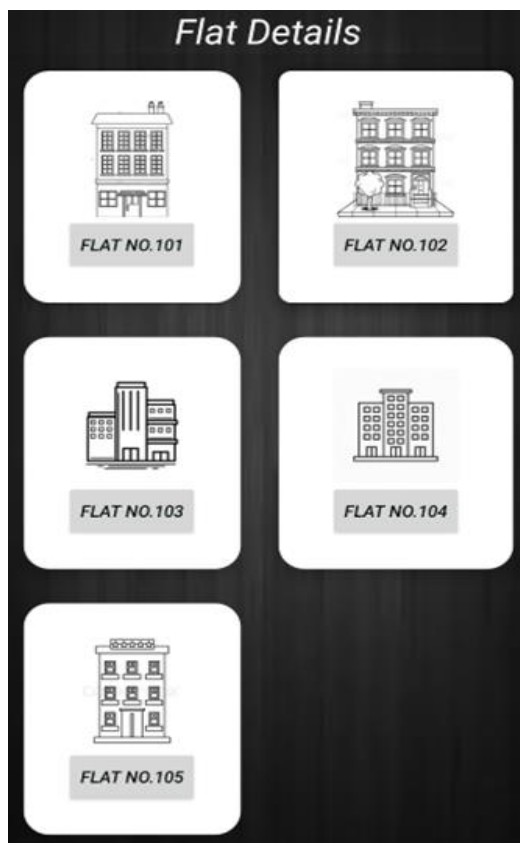
3.Admin Page-



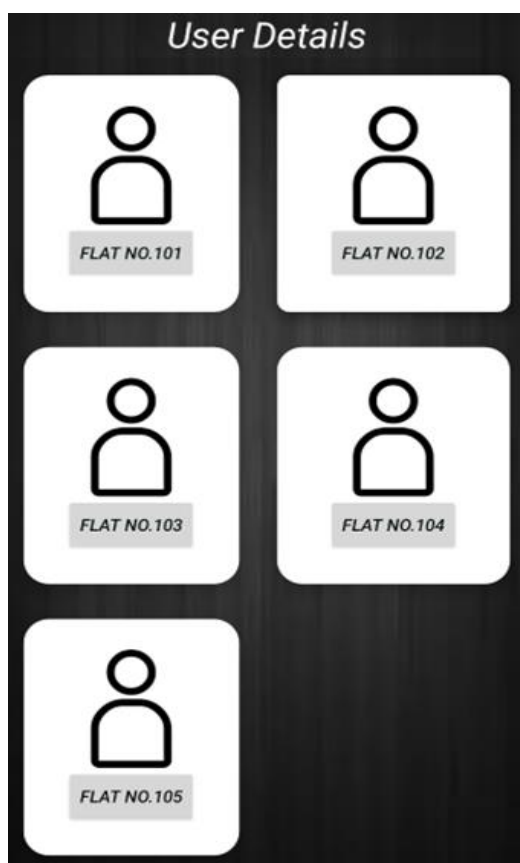
4.User Page-



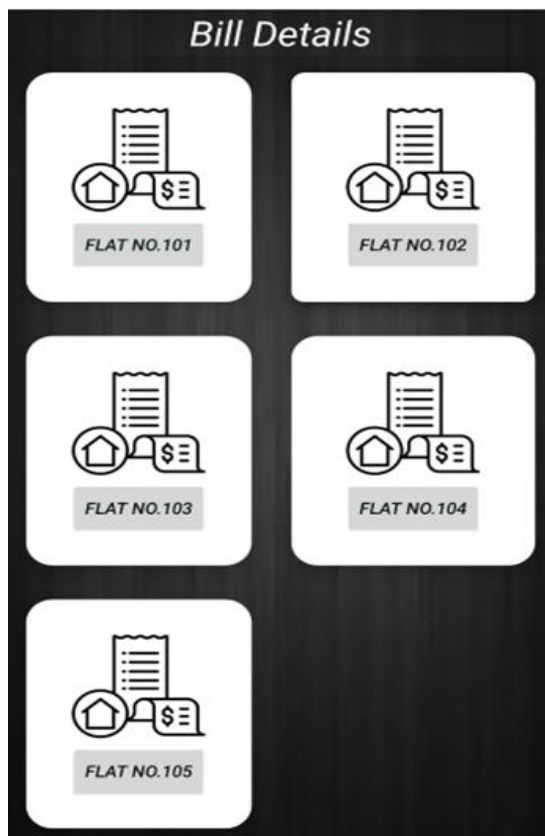
5.Flat Details (Admin)-



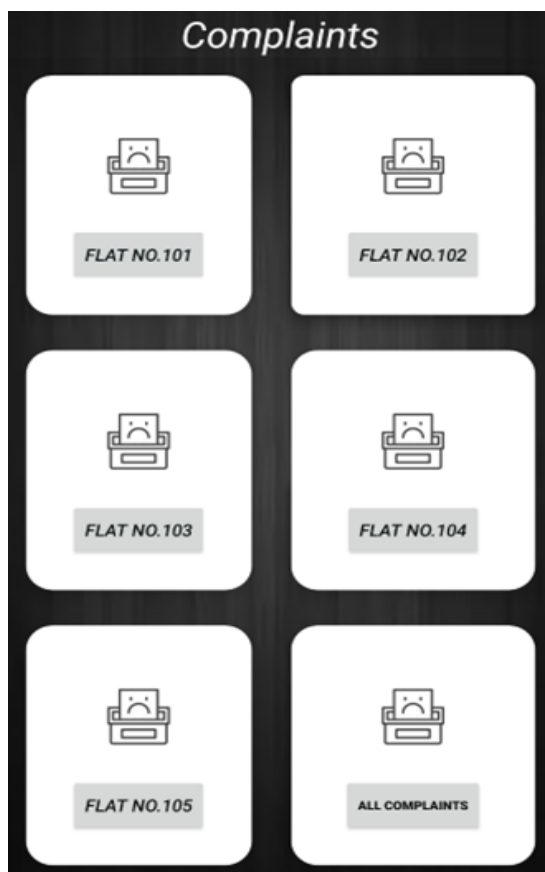
6.User Details (Admin)-



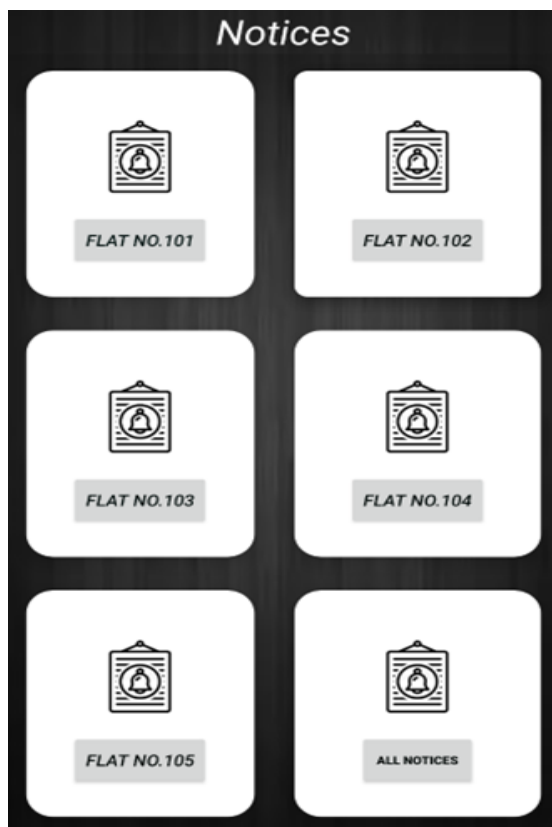
7.Bill Details (Admin)-



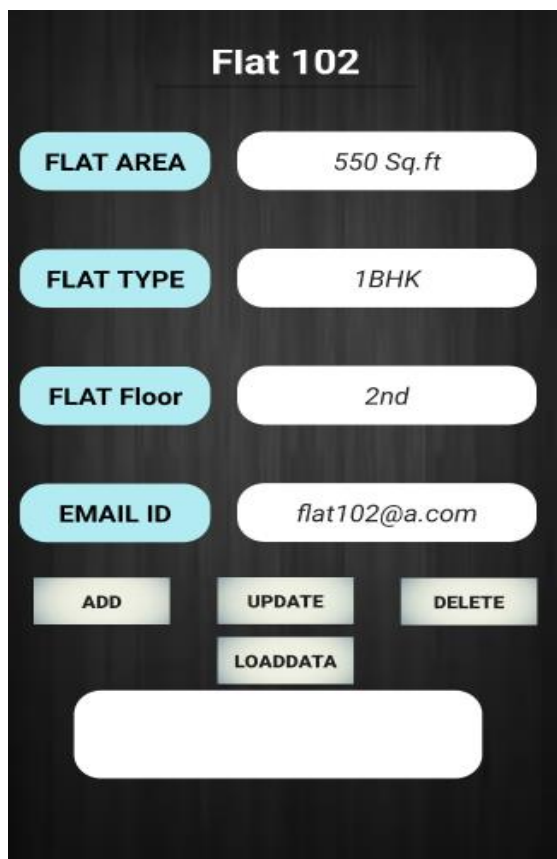
8.Complaints (Admin/User)-



9. Notices (Admin/User)-



10.Flat Details – Admin/User (Flat 101)-



11. User Details – Admin/User (Flat 101)-

Flat 102

NAME	Pankaj
MOBILE NO.	9147961788
DOB.	24-4-1994
ADDRESS	DNYA bldg.Mumbai

ADD

UPDATE

DELETE

LOADDATA

12. Bill Details – Admin/User (Flat 101)-

Flat 101

WATER BILL	100 Rs.
PARKING BILL	200 Rs.
EVENT FUND	300 Rs.
SERVICE CHARGE	400 Rs.
TOTAL	1000 Rs.
PAID	750 Rs.
BALANCE	250 Rs.


ADD

UPDATE

DELETE

13.Complaints – Admin/User (Flat 101)-

Flat 101 - User



Enter Your Message

SEND

14.Notices – Admin/User (Flat 101)-

Flat 101 - Admin



Enter Your Message

SEND

15.Feedback – Admin-

Feedback - Admin

Name :-

Email :-

☐ Male ☐ Female

☐ ☐ ☐ ☐ ☐

☐ Are you sure you want to submit? Check the box then!

16.Documents Upload – User (Flat 101)-

DOCS - 101

4.4 Test Case Design

Test Cases for Login (Secretary/Flat Member) :-

Test Case No.	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Login with valid username and valid password (Admin/User).	Login should be successful and it should show the home screen of the admin and user.	Pass	Login Successful
2.	Login with invalid username and invalid password (Admin/User).	“error” message should be displayed.	Pass	Error Message
3.	Login with valid username and invalid password (Admin/User).	“error” message should be displayed.	Pass	Error Message
4.	Login with invalid username and valid password (Admin/User).	“error” message should be displayed.	Pass	Error Message

5.	Login without entering username and password (Admin/User).	“error” message should be displayed.	Pass	Error Message
----	--	--------------------------------------	------	---------------

Test Cases for Admin (Secretary) :-

Test Case No.	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Click on the Flat button	It should display the flat detailed information of flat members	Pass	Result as expected
2.	Click on the User button	It should display the user detailed information of flat members.	Pass	Result as expected
3.	Click on the Bill button	It should display the bill detailed information of flat members.	Pass	Result as expected
4.	Click on the Complaint button	It should display a text where complaints are being viewed.	Pass	Result as expected

5.	Click on the Notice button	It should display a text where notices are being send.	Pass	Result as expected
----	----------------------------	--	------	--------------------

Test Cases for User (Flat Member) :-

Test Case No.	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Click on the Add/Update button	It should display the user information where he can add and update his data.	Pass	Result as expected
2.	Click on the view pay bill button	It should display the detailed information where he can pay bills.	Pass	Result as expected
3.	Click on the complaint button	It should display a text where complaints are being send.	Pass	Result as expected
4.	Click on the notice button	It should display a text where notices are being viewed.	Pass	Result as expected

Test Cases for Flat Details :- (Secretary)

Test Case No.	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Flat No.	It should display the flat number of the flat owner.	Pass	Result as expected
2.	Email Id.	It should display the email id of the flat owner	Pass	Result as expected
3.	Flat Type	It should display the flat type (eg.2BHK)	Pass	Result as expected
4.	Flat Area	It should display the flat sq. ft area	Pass	Result as expected
5.	Flat Floor	It should display the floor no. of the flat owner	Pass	Result as expected

Test Cases for User Details :- (Secretary)

Test Case No.	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Flat No.	It should display the flat number of the flat owner.	Pass	Result as expected
2.	Name	It should display the name of the flat owner.	Pass	Result as expected
3.	Mobile	It should display the mobile number of the flat owner.	Pass	Result as expected
4.	Date of Birth	It will display the birthdate of the flat owner.	Pass	Result as expected
5.	Address	It will display a specific address of the flat owner.	Pass	Result as expected

Test Cases for User Details :- (Flat Member)

Test Case No.	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Flat No. (Click on Add/Update/Delete)	It should display the flat number of the flat owner and also edit the details.	Pass	Result as expected
2.	Name (Click on Add/Update/Delete)	It should display the name of the flat owner and also edit the details.	Pass	Result as expected
3.	Mobile (Click on Add/Update/Delete)	It should display the mobile number of the flat owner and also edit the details.	Pass	Result as expected
4.	Date of Birth (Click on Add/Update/Delete)	It will display the birthdate of the flat owner and	Pass	Result as expected

		also edit the details.		
5.	Address (Click on Add/Update/Delete)	It will display a specific address of the flat owner and also edit the details.	Pass	Result as expected

Test Cases for Bill Details :- (Secretary)

Test Case No.	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Flat No.	It should display the flat number of the flat owner.	Pass	Result as expected
2.	Water Bill	It will display the amount of water bill.	Pass	Result as expected
3.	Parking Bill	It will display the amount of parking bill.	Pass	Result as expected
4.	Event Fund	It will display the amount of Funds that are being collected.	Pass	Result as expected
5.	Service Charge	It will display the service charges of the society.	Pass	Result as expected

6.	Bill Payments (Click on the Generate Bill)	It will display the total amount of the bill payment and records.	Pass	Result as expected
----	--	---	------	--------------------

Test Cases for Bill Details :- (Flat Member)

Test Case No.	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Flat No.	It should display the flat number of the flat owner.	Pass	Result as expected
2.	Water Bill	It will display the amount of water bill.	Pass	Result as expected
3.	Parking Bill	It will display the amount of parking bill.	Pass	Result as expected
4.	Event Fund	It will display the amount of Funds that are being collected.	Pass	Result as expected
5.	Service Charge	It will display the service charges of the society.	Pass	Result as expected
6.	Bill Payments (Click on Pay Bill)	It will show a records of the bill payments	Pass	Result as expected

Test Cases for Complaints :- (Admin/User)

Test Case No	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Click on show button (Admin)	It will display a complaint message that has been sent from the user	Pass	Result as expected
2.	Click on Submit button (User)	It will send a complaint message to the admin.	Pass	Result as expected

Test Cases for Notices :- (Admin/User)

Test Case No	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Click on show button (User)	It will display a notice that has been sent from the admin.	Pass	Result as expected
2.	Click on Submit button (Admin)	It will send a notice to the user.	Pass	Result as expected

Test Cases for Feedbacks :- (Admin/User)

Test Case No	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Click on submit button (Admin/User)	It will submit the feedback.	Pass	Result as expected
2.	Click on Submit Rating button (Admin/User)	It will submit the rating.	Pass	Result as expected

Test Cases for Documents :- (User)

Test Case No	Test Case Description	Expected Result	Pass / Fail	Actual Result
1.	Click on browse button	It will select images from the gallery.	Pass	Result as expected
2.	Click on upload button	It will upload the images into the database.	Pass	Result as expected

CHAPTER 5 : IMPLEMENTATION AND TESTING

5.1 Implementation Approaches

In this project we have developed a working prototype model with the current available requirement details and get the feedback from the customer for the actual requirement of the product to develop the system. Prototype is the trimmed version of the actual product with limited features and functionalities.

The system has been developed using Android Studio as the software and FIREBASE as database. The front end or the GUI part of the system is coded in XML along with Java as the main language.

Testing such as Unit testing, Integration testing, functional testing and System testing has been performed in order to understand the quality of the project and how efficiently it works.

The main intention of using Android studio is that, it makes it easy to develop an android application. Android Studio's Apply Changes feature lets you push code and resource changes to your running app without restarting it.

It provides intelligent code editor which helps you write better code, work faster and be more productive by offering advanced code completion, refactoring, and code analysis.

Android studio also provides extensive tools to help you test your Android app with Junit 4 and functional UI test frameworks. You can run your test on a device or emulator, a continuous integration environment, or in Firebase Test Labs.

Hence, the implementation and installation of the system becomes easy and can be easily installed on a user's device.

5.2 Code Details

Login Page-

```
package com.example.societymanagement;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText _txtUser, _txtPass;
    Button _btnLogin;
    Spinner _spinner;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        _txtPass=(EditText)findViewById(R.id.txtPass);
        _txtUser=(EditText)findViewById(R.id.txtUser);
        _btnLogin=(Button)findViewById(R.id.btnLogin);
        _spinner=(Spinner) findViewById(R.id.spinner);
        ArrayAdapter <CharSequence> adapter =
        ArrayAdapter.createFromResource( this,R.array.usertype,
        R.layout.support_simple_spinner_dropdown_item);
        _spinner.setAdapter(adapter);
        _btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String item = _spinner.getSelectedItem().toString();
                if (_txtUser.getText().toString().equals("Secretary") &&
                _txtPass.getText().toString().equals("sec123") && item.equals("admin")) {
                    Intent intent = new Intent(MainActivity.this,
                    admin.class);
                    startActivity(intent);
                }
                else if (_txtUser.getText().toString().equals("Flat 101")
                && _txtPass.getText().toString().equals("f1") && item.equals("user")) {
                    Intent intent = new Intent(MainActivity.this,
                    user.class);
                    startActivity(intent);
                }
            }
        });
    }
}
```

```

        }
        else if (_txtUser.getText().toString().equals("Flat 102")
&& _txtPass.getText().toString().equals("f2") && item.equals("user")) {
            Intent intent = new Intent(MainActivity.this,
User1.class);
            startActivity(intent);
        }
        else if (_txtUser.getText().toString().equals("Flat 103")
&& _txtPass.getText().toString().equals("f3") && item.equals("user")) {
            Intent intent = new Intent(MainActivity.this,
User2Activity.class);
            startActivity(intent);
        }
        else if (_txtUser.getText().toString().equals("Flat 104")
&& _txtPass.getText().toString().equals("f4") && item.equals("user")) {
            Intent intent = new Intent(MainActivity.this,
User3Activity.class);
            startActivity(intent);
        }
        else if (_txtUser.getText().toString().equals("Flat 105")
&& _txtPass.getText().toString().equals("f5") && item.equals("user")) {
            Intent intent = new Intent(MainActivity.this,
User4Activity.class);
            startActivity(intent);
        }
        else {
            Toast.makeText(getApplicationContext(),"Login
Unsuccessful", Toast.LENGTH_LONG).show();
        }
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu,menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    int id = item.getItemId();
    switch (id) {
        case R.id.item1:
            Toast.makeText(MainActivity.this, "Info has been pressed",
Toast.LENGTH_SHORT).show();
            break;
        case R.id.item2:
            Toast.makeText(MainActivity.this, "Sharing has been
pressed", Toast.LENGTH_SHORT).show();
            break;
        case R.id.item3:
            Toast.makeText(MainActivity.this, "Bluetooth has been
pressed", Toast.LENGTH_SHORT).show();
            break;
    }
}

```



```

        }
        return true;
    }

    @Override
    public void onBackPressed() {
        new AlertDialog.Builder(this)
            .setIcon(R.drawable.icon1)
            .setTitle("Society Management App")
            .setMessage("Are you sure you want to close the App?")
            .setPositiveButton("YES", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which)
{
                    finishAffinity();
                }
            })
            .setNegativeButton("NO", null)
            .show();
    }
}

```

Details (Add/Update/Delete)-

```
package com.example.societymanagement;

import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.SetOptions;

import org.w3c.dom.Document;

import java.lang.reflect.Field;
import java.util.HashMap;
import java.util.Map;

public class adminflat101 extends AppCompatActivity {

    private static final String TAG = "adminflat101";

    private static final String KEY_FLATAREA = "flatarea";
    private static final String KEY_FLATTYPE = "flattype";
    private static final String KEY_FLATFLOOR = "flatfloor";
    private static final String KEY_EMAILID = "emailid";

    private EditText ed1;
    private EditText ed2;
    private EditText ed3;
    private EditText ed4;
    private TextView t1;

    private FirebaseFirestore db = FirebaseFirestore.getInstance();
    private DocumentReference noteRef = db.document("Flat Details/Flat
101");

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_adminflat101);

        ed1 = findViewById(R.id.ed1);
        ed2 = findViewById(R.id.ed2);
        ed3 = findViewById(R.id.ed3);
        ed4 = findViewById(R.id.ed4);
        t1 = findViewById(R.id.t1);
    }
}
```

```

    }

    public void add(View view) {
        String flatarea = ed1.getText().toString();
        String flattype = ed2.getText().toString();
        String flatfloor = ed3.getText().toString();
        String email = ed4.getText().toString();

        Map<String, Object> note = new HashMap<>();
        note.put(KEY_FLATAREA, flatarea);
        note.put(KEY_FLATTYPE, flattype);
        note.put(KEY_FLATFLOOR, flatfloor);
        note.put(KEY_EMAILID, email);

        noteRef.set(note)
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    Toast.makeText(adminflat101.this, "Data Inserted
Successfully", Toast.LENGTH_SHORT).show();
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(adminflat101.this, "Data Inserted
Successfully", Toast.LENGTH_SHORT).show();
                    Log.d(TAG, e.toString());
                }
            });
    }

    public void loaddata(View view) {
        noteRef.get()
            .addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
                @Override
                public void onSuccess(DocumentSnapshot
documentSnapshot) {
                    if (documentSnapshot.exists()){
                        String flatarea =
documentSnapshot.getString(KEY_FLATAREA);
                        String flattype =
documentSnapshot.getString(KEY_FLATTYPE);
                        String flatfloor =
documentSnapshot.getString(KEY_FLATFLOOR);
                        String email =
documentSnapshot.getString(KEY_EMAILID);

                        //Map<String, Object> note =
documentSnapshot.getData();

                        t1.setText("FLATAREA: " + flatarea + "\n" +
"FLATTYPE: " + flattype + "\n" + "FLATFLOOR: " + flatfloor + "\n" +
"EMAILID: " + email);
                    }else{
                        Toast.makeText(adminflat101.this, "Data Does
Not Exist", Toast.LENGTH_SHORT).show();
                    }
                }
            });
    }

```

```

        }
    }
})
.addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText(adminflat101.this, "",
Toast.LENGTH_SHORT).show();
        Log.d(TAG,e.toString());
    }
});
}

public void update(View view) {
    String flatarea = ed1.getText().toString();
    String flattype = ed2.getText().toString();
    String flatfloor = ed3.getText().toString();
    String email = ed4.getText().toString();

    Map<String, Object> note = new HashMap<>();
    note.put(KEY_FLATAREA,flatarea);
    note.put(KEY_FLATTYPE,flattype);
    note.put(KEY_FLATFLOOR,flatfloor);
    note.put(KEY_EMAILID,email);

    //noteRef.set(note, SetOptions.merge());
    noteRef.update(note);

    Toast.makeText(this, "Data Updated Successfully",
Toast.LENGTH_SHORT).show();
}

public void delete(View view) {
    //Map<String, Object> note = new HashMap<>();
    //note.put(KEY_FLATAREA, FieldValue.delete());
    //note.put(KEY_FLATTYPE, FieldValue.delete());
    //note.put(KEY_FLATFLOOR, FieldValue.delete());
    //note.put(KEY_EMAILID, FieldValue.delete());

    //noteRef.update(note);

    noteRef.update(KEY_FLATAREA, FieldValue.delete());
    noteRef.update(KEY_FLATTYPE, FieldValue.delete());
    noteRef.update(KEY_FLATFLOOR, FieldValue.delete());
    noteRef.update(KEY_EMAILID, FieldValue.delete());

    Toast.makeText(this, "Data Deleted Successfully",
Toast.LENGTH_SHORT).show();
}
}

```

Image Upload-

```
package com.example.societymanagement;

import android.app.ProgressDialog;
import android.content.ContentResolver;
import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.webkit.MimeTypeMap;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;

import java.io.IOException;

public class docuser1 extends AppCompatActivity {

    Button btnbrowse, btnupload;
    EditText txtdata ;
    ImageView imgview;
    Uri FilePathUri;
    StorageReference storageReference;
    DatabaseReference databaseReference;
    int Image_Request_Code = 7;
    ProgressDialog progressDialog ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_docuser1);

        storageReference =
        FirebaseStorage.getInstance().getReference("Images");
        databaseReference =
        FirebaseDatabase.getInstance().getReference("Images");
        btnbrowse = (Button)findViewById(R.id.btnbrowse);
        btnupload= (Button)findViewById(R.id.btnupload);
        txtdata = (EditText)findViewById(R.id.txtdata);
        imgview = (ImageView)findViewById(R.id.image_view);
        progressDialog = new ProgressDialog(docuser1.this);// context name
as per your project name
```

```

        btnbrowse.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent();
                intent.setType("image/*");
                intent.setAction(Intent.ACTION_GET_CONTENT);
                startActivityForResult(Intent.createChooser(intent, "Select
Image"), Image_Request_Code);
            }
        });
        btnupload.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                UploadImage();

            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {

        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == Image_Request_Code && resultCode == RESULT_OK &&
data != null && data.getData() != null) {

            FilePathUri = data.getData();

            try {
                Bitmap bitmap =
MediaStore.Images.Media.getBitmap(getContentResolver(), FilePathUri);
                imgview.setImageBitmap(bitmap);
            }
            catch (IOException e) {

                e.printStackTrace();

            }
        }
    }

    public String GetFileExtension(Uri uri) {

        ContentResolver contentResolver = getContentResolver();
        MimeTypeMap mimeTypeMap = MimeTypeMap.getSingleton();
        return
mimeTypeMap.getExtensionFromMimeType(contentResolver.getType(uri)) ;

    }

    public void UploadImage() {

```

```

        if (FilePathUri != null) {

            progressDialog.setTitle("Image is Uploading...");
            progressDialog.show();
            StorageReference storageReference2 =
storageReference.child(System.currentTimeMillis() + "." +
GetFileExtension(FilePathUri));
            storageReference2.putFile(FilePathUri)
                .addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {

                        String TempImageName =
txtdata.getText().toString().trim();
                        progressDialog.dismiss();
                        Toast.makeText(getApplicationContext(), "Image
Uploaded Successfully ", Toast.LENGTH_LONG).show();
                        @SuppressWarnings("VisibleForTests")
                        uploadinfo imageUploadInfo = new
uploadinfo(TempImageName, taskSnapshot.getUploadSessionUri().toString());
                        String ImageUploadId =
databaseReference.push().getKey();

                        databaseReference.child(ImageUploadId).setValue(imageUploadInfo);
                    }
                });
        }
        else {

            Toast.makeText(docuser1.this, "Please Select Image or Add Image
Name", Toast.LENGTH_LONG).show();

        }
    }
}

```

5.3 Testing Approach

5.3.1 Unit Testing-

Unit Testing is a level of software testing where individual units/components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

It is concerned with functional correctness of the standalone modules. Unit testing is performed on each module of the system i.e. login page, admin page, user page, flat details page, user details page, bill details page, complaint page, notice page, feedback page and documents page.

5.3.2 Integration Testing-

In integration testing all the modules are combined together and executed. This helps to understand if all the modules work accordingly and is error free and runs in the required sequence i.e. as the user logs into the account the home page of the system opens which consists of the buttons for the aforementioned modules of the system.

The user can select any of the modules, when a particular module is selected the system takes the user to that particular module page i.e. when the pill reminder module is selected by the user the system opens the pill reminder page of the system where the user can add their medicines and set alarm.

CHAPTER 6 : RESULTS AND DISCUSSION

Test Report-

Depending on the test cases we can drive that the outcome of the test cases is positive and the system developed is successful in performing the required functions.

All the functionalities are cross checked with valid and invalid inputs and with the help on those outputs this test report is derived.

If any invalid inputs are entered the system alters the user to enter valid information.

Multiple combinations of inputs (valid, invalid) have been entered to check systems accuracy and its response.

If the valid input is enter no error message is displayed and with invalid input error message is displayed, this helps the user to enter the necessary and required information.

CHAPTER 7 : CONCLUSIONS

- This Society management system is an amazing one which can reduce the time and increase the convenience of the members to a great extent. They are provided with a lot of features related to their house.
- The society members can communicate with the help of this system and can come together in case of any problem caused. Members will also get notified about their status of payment to several areas like the maintenance fees, bills, rents, etc.
- The software product thus developed is fully operational and desired solution. It is been tested practically which results in successful outcome.
- Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
- Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time than manual.
- Facilities and ease of use is provided to the user with the help of the system. The system provides accurate, fast, easy data access which is beneficial for both the user and the admin.

CHAPTER 8 : REFERENCES

https://www.youtube.com/watch?v=zSgrMVt_MFg

<https://www.youtube.com/watch?v=K2V6Y7zQ8NU>

<https://www.youtube.com/watch?v=gWHt2m86CgM>

<https://becody.com/how-to-design-login-screen-using-relative-layout/>

<https://www.youtube.com/watch?v=Z5mXiFW0Iws>

<https://developer.android.com/studio/write/vector-asset-studio>

<https://www.youtube.com/watch?v=7OvsWwbvYsM>

<https://developer.android.com/studio/install>

<https://developer.android.com/support>