Mini Project report on

SOCIETY MAINTENANCE SYSTEM

by

JAY VISAVE [2019450059]

NINAD PATWARDHAN [2019450042]

Under the guidance of

Internal Supervisor

Dr. AARTI M KARANDE

Department of Master of Computer Applications

Sardar Patel Institute of Technology

Autonomous Institute Affiliated to Mumbai University

2020-21

CERTIFICATE OF APPROVAL

This is to certify that the following students

JAY VISAVE [2019450059]

NINAD PATWARDHAN [2019450042]

Have satisfactorily carried out work on the project entitled

"**SOCIETY MAINTENANCE SYSTEM**"

Towards the fulfillment of summer project, as laid down by University of Mumbai during year 2020-21.

_____

Project Guide

Dr. Aarti Karande

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

JAY VISAVE [2019450059]

NINAD PATWARDHAN [2019450042]

Have successfully completed the Project report on **"SOCIETY MAINTENANCE SYSTEM"**, which is found to be satisfactory and is approved

At

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI.

INTERNAL EXAMINER

EXTERNAL EXAMINER

Head of Department

(Dr. Pooja Raundale)

Principal

(Dr. B.N.Chaundari)

CONTENT

ABSTRACT

A housing society management and billing project that effectively manages and handles all the functioning of a cooperative housing society. The software system can store the data of various flat owners. 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 bills. 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 facilities. It calculates various associated costs, adds them up and provides a bill accordingly.

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 PROBLEM DEFINITION:

To create an application which can handle the maintenance bills of the society and eliminate the need of printed bills to be distributed to every flat which will result in saving paper and efficient billing of maintenance charges.

The users can view their monthly bills on this application and they do not have to receive printed bills. They can also file complaints using the application and also request of NOCs.

1.2 OBJECTIVES AND SCOPE

1.2.1 OBJECTIVES:

- To eliminate the need of distribution of printed bills.
- To maintain efficient payment of maintenance taxes.

1.2.2 SCOPE:

- The application is designed to handle data of multiple societies.
- To an admin, complaints and NOC requests of that society will be visible.
- When a complaint is resolved, they can change the complaint status from their end which will be reflected to the users end.
- Multiple users can be added and there can be multiple complaints filed by them.
- The users will also see previous month's bill along with the current bill under bills tab.

1.3 SYSTEM REQUIREMENTS

1.3.1 HARDWARE REQUIREMENTS:

Processor	: Dual Core Processor and above.
RAM	: 512 MB or above.
Storage	: Minimum Hard disk space.

1.3.2 SOFTWARE REQUIREMENTS:

Operating System	: Windows OS
Software	: Any code editor
Languages	: TypeScript, HTML
Database	: Firebase

Chapter 2

Literature Survey

Current System:

In many big societies the maintenance taxes are collected physically via cheques or cash. This process becomes very tedious for the office members to follow as they will have to maintain lots of folders containing data of many years. This also causes problems like no effective maintainability, chances of miscommunication, double payment of maintenance charges, etc.

Even in some societies, a desktop application is used to keep track of the maintenance charge payments and dues. Using this system is profitable for the staff members but it still isn't full proof. The maintenance bills still have to be handed out physically and have to be collected via cheques. Housing society management authorities use a traditional way of communication which includes a common notice board system operated by responsible society members. Currently many housing societies are following the traditional way to convey notices and to inform residents about meetings and to contact higher authorities in case of any complaints though it is time consuming and not so reliable.

Proposed System:

The Society Maintenance system we propose will have an application which will maintain the maintenance bills, payments and dues on a database instead of maintaining huge files and collecting cheques from each individual flat. The application will give an option of payment of monthly charges online and the data will be available for the staff members to review. The application will also provide residents with the facility to file a complaint against a fellow resident or any general complaint to the society. In case of a flat changing hands, the application will also provide a facility to request to the society for an NOC. The resident's previous payment dues and any outstanding complaints will be checked before clearing the NOC.

Chapter 3

SOFTWARE REQUIREMENT SPECIFICATION [SRS] AND DESIGN

3.1 INTRODUCTION:

- A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development.
- The SRS fully describes what the software will do and how it will be expected to perform.

3.2 OVERALL DESCRIPTION:

3.2.1 PRODUCT PERSPECTIVE:

- The product is an application which can run on both Android and IOS systems.
- It is designed to target housing societies to avoid chaos which takes place while maintaining printed bills.
- It is also easier for the residents to file complaints via phone.

3.2.2 PRODUCT FUNCTIONS:

- The basic function of the system is to generate monthly bills which contains the factors as parking charges, monthly interest, etc.
- There are two more functions for the residents as file a complaint and request for NOC.
- Where the user can complaint about parking issue, leakage or other type of problems and request for a No Objection Certificate if required.

3.2.3 USER CHARACTERISTICS:

- The user needs to have the basic knowledge about handling the system.
- The manager should just know about the basic functioning of the system.
- The Admin is expected to be familiar with the interface of the tech support system.

3.2.4 DEPENDENCIES AND ASSUMPTIONS:

- The system is dependent on the internet connectivity as the data is stored in firebase cloud.
- Developer must have node js installed in the system.
- Developer must have a google account to access firebase.
- Developer must know ionic and typescript.

3.3 SYSTEM FEATURES:

3.3.1 FUNCTIONAL REQUIREMENTS:

- **Registration fields:** All the values must be entered or selected at the time of registration and they should be successfully transferred to the database and a document by the user id must be created.
- **Generate bill:** A bill must be generated according to the society's parameters which will be obtained from the society's document at the start of every month for every user.
- **Download bill:** The user must be able to download the bill in PDF format.
- **Complaints:** A complaint counter must be increased in the database when a user files a complaint.

3.3.2 NON- FUNCTIONAL REQUIREMENTS:

- **Usability:** Usability is the degree to which an application can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use. This system can be used very easily by anyone.
- **Efficiency:** It is the ability to do things well, successfully. The functions are efficient enough to carry out their functionalities.
- **Performance:** Performance is another criterion which is fulfilled by this system.
- **Reliability:** The system is trustworthy and is performing consistently well.

Chapter 4

ANALYSIS AND DESIGN

4.1 METHODOLOGIES ACCEPTED:

- For this project, we decided to use **Rapid Application Development Model.**
- Since the project was of short duration, the Rad model fit perfectly to it. Even the team of 2 members.
- In RAD, parallel prototypes are made and they are merged together. We started with the login/registration and instructions page parallelly and merged them when they were done. Similarly, we did the other pages parallelly as well.
- **Business modelling:** We took around two weeks of time to decide our problem statement. In that time, we studied how the maintenance system works in a society on a physical level. On the basis of the results, we decided what we want to include in our application.
- **Data modelling:** Based on information obtained from business model, we decided what all classes and database documents our application will have.
- **Process modelling:** After deciding the classes and documents we decided which functions will connect to the database and what the process flow will be like.
- **Application Generation:** Then we started with the implementation of the application and we started the testing along with it.

4.2 DESIGN:

4.2.1 ACTIVITY DIAGRAM:

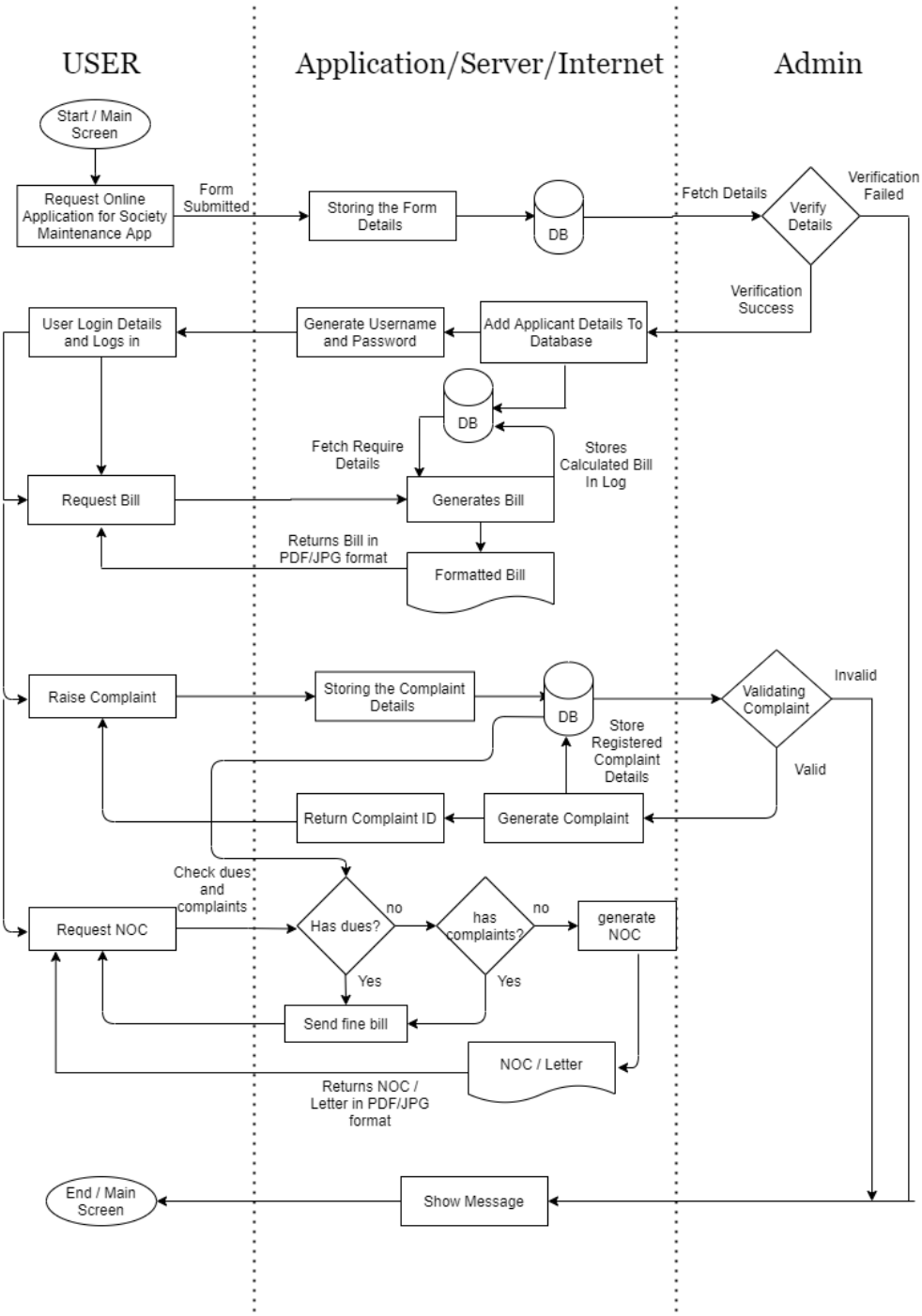

Fig 4.1 Activity Diagram

Activity Diagram Specification

Step	User	System	Business Rule
1	Login/Register		
2		Verify Details	BR_1
3	Request bill		
4		Generate bill	BR_2
5	Raise complaint		
6		Store complaint	BR_3
7		Error Message	BR_4
8	Request NOC		
9		Check dues	BR_5
10		Accept/Reject NOC	BR_6
11		Error Message	BR_7

ID	Business rule	Business rule description
BR_1	User login/registration details	First Name Last Name Society name Wing Flat Type Phone Number Email Id Password
BR_2	Generation of monthly bill	Month Year Other society details
BR_3	Accepting complaint	Complaint type Description Other user details
BR_4	Error Message	If BR_3 does not match
BR_5	Checking dues when requested for NOC	Unpaid bills Pending complaints
BR_6	Accept NOC	Depending on result of BR_5
BR_7	Error Message	If BR_6 does not match

Table 4.1

4.2.2 Bill Generation:

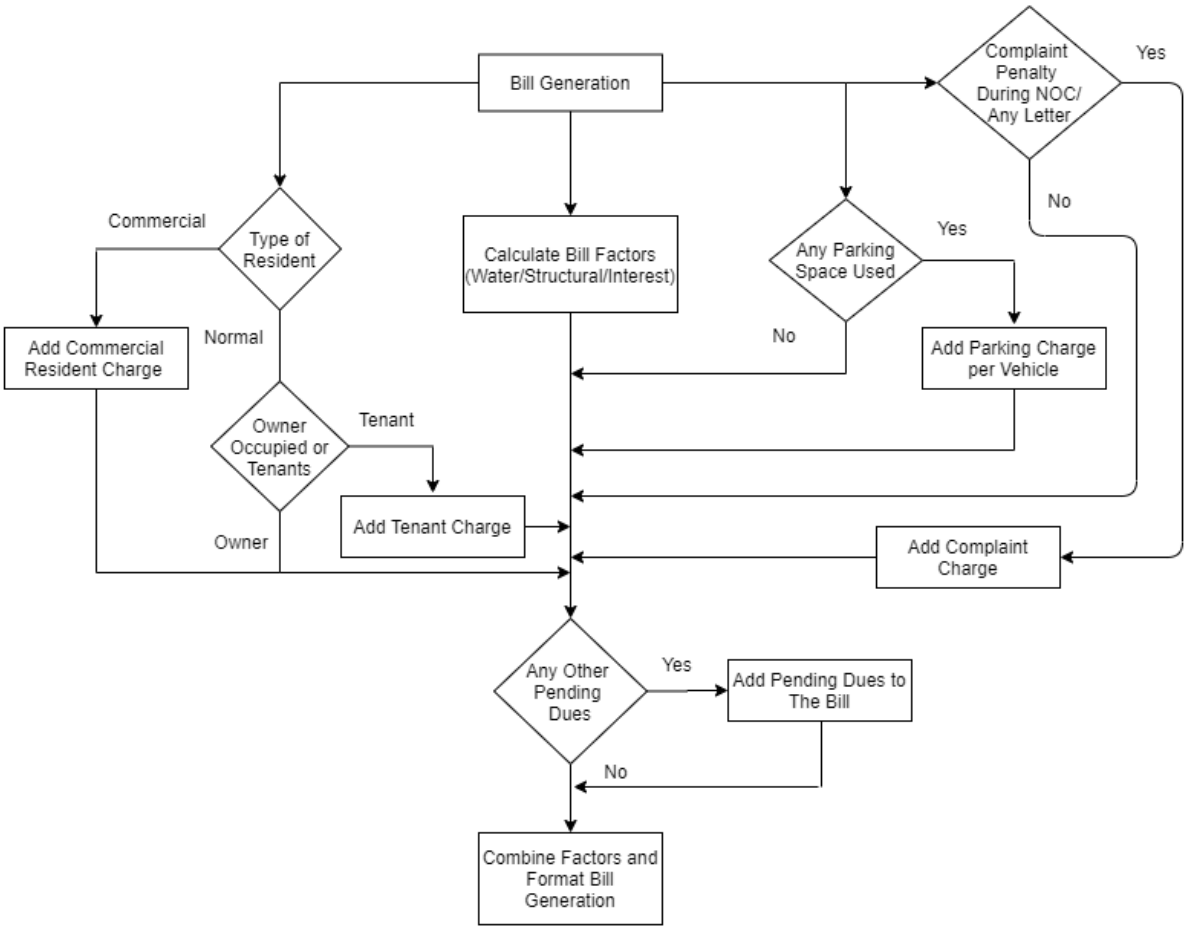

Fig 4.2 Bill generation

4.2.3 USE-CASE DIAGRAM:

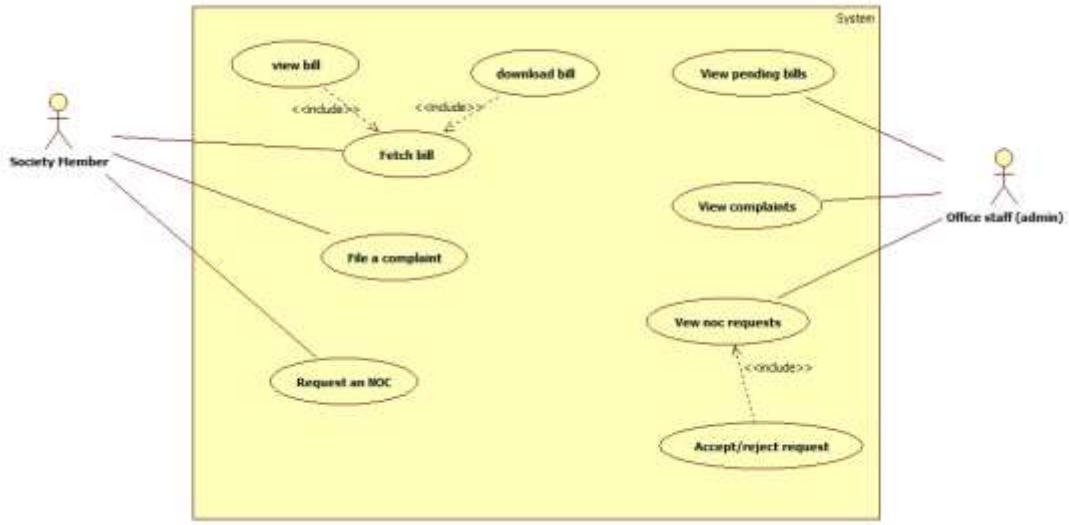

Fig 4.3 Usecase diagram

Usecase Specification

UseCase ID:	101
UseCase Name:	Society Management System UseCase Diagram
Created By:	Ninad Patwardhan & Jay Visave
Date Created:	18/8/2020

Actors:	Users(Society Members), Admin(Office Staff)
Description of UseCases:	1. **Fetch bill:** This will contain the monthly maintenance bill 2. **File a complaint:** Here, the user can complain about the problems they face. 3. **Request NOC:** The user can request for a No Object Certificate when they need to change flat or apply for loan. 4. **View complaints:** The admin can see all the complaints filed by the users. 5. **View NOC requests:** All the NOC requests will be available here.
Preconditions:	1. The user must have registered to the Society Management Application to access the features

	2. The admin should be logged in to the system to view all complaints and NOC requests.
Postconditions:	1. When the user clicks on download PDF, a pdf must be generated of that bill. 2. When the admin resolves a complaint, it should be reflected as solved at the user side.
Includes:	1. Fetch bill includes view bill and download bill 2. View NOC requests includes accept/reject NOC request.
Assumptions:	1. To use the above usecases, it is assumed that the user and office staff has logged in to the system.

Table 4.2

4.2.4 CLASS RELATIONAL DIAGRAM:

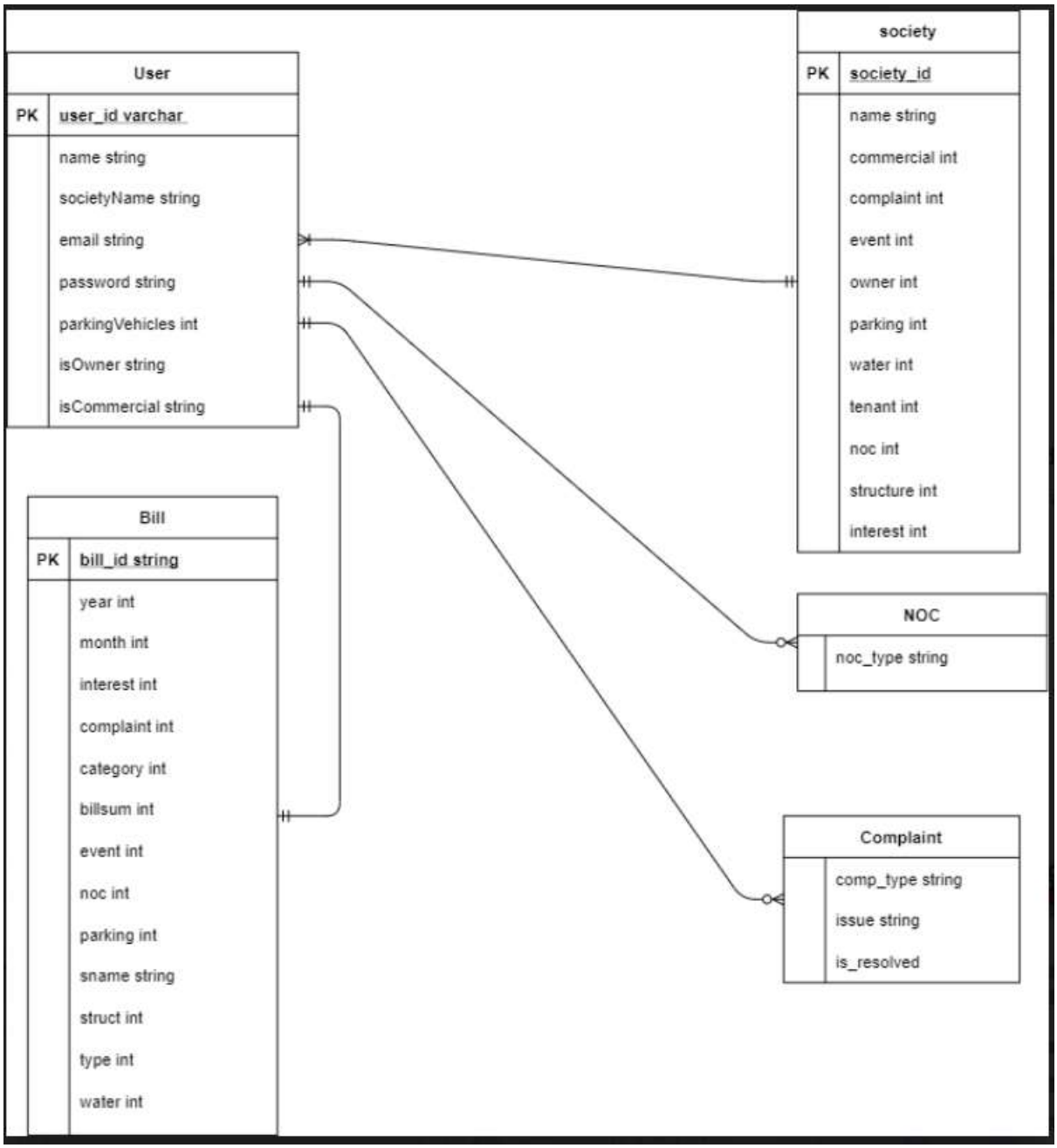

Fig 4.4 Class Relational diagram

Class relational diagram specification

Sr. No.		
1	Class Name	User
2	Description/ Responsibilities	Contains all the information of a user.
3	Type/ Collaborations	-
4	Multiplicity Values	User-Society [Many-1] User-Complaint [1-Many] User-Bill [1-1] User-NOC [1-Many]
5	Functions	getBill(), getFiledcomplaints(), FileaComplaint(), RequestNOC(), getNOC()
6	Objects	Ninad, Jay

Sr. No.		
1	Class Name	Society
2	Description/ Responsibilities	Contains all the information about the charges a society for maintenance.
3	Type/ Collaborations	-
4	Multiplicity Values	Society-User [1-Many]
5	Functions	getBillDetails()
6	Objects	Wakanda CHS

Sr. No.		
1	Class Name	Bill
2	Description/ Responsibilities	Contains all the information of a user's monthly maintenance bill.
3	Type/ Collaborations	-
4	Multiplicity Values	Bill-User [Many-1]

5	Functions	displayBill()
6	Objects	Sep-2020

Sr. No.		
1	Class Name	Complaint
2	Description/ Responsibilities	Contains all the information of a complaint raised by the user
3	Type/ Collaborations	-
4	Multiplicity Values	Complaint-User [Many-1]
5	Functions	FileComplaint()
6	Objects	leakage

Sr. No.		
1	Class Name	NOC
2	Description/ Responsibilities	Contains all the information of an NOC request by a user.
3	Type/ Collaborations	-
4	Multiplicity Values	NOC-User [Many-1]
5	Functions	ReuqestNOC()
6	Objects	HomeLoan

Table 4.3

4.2.5 COMMUNICATION DIAGRAM:

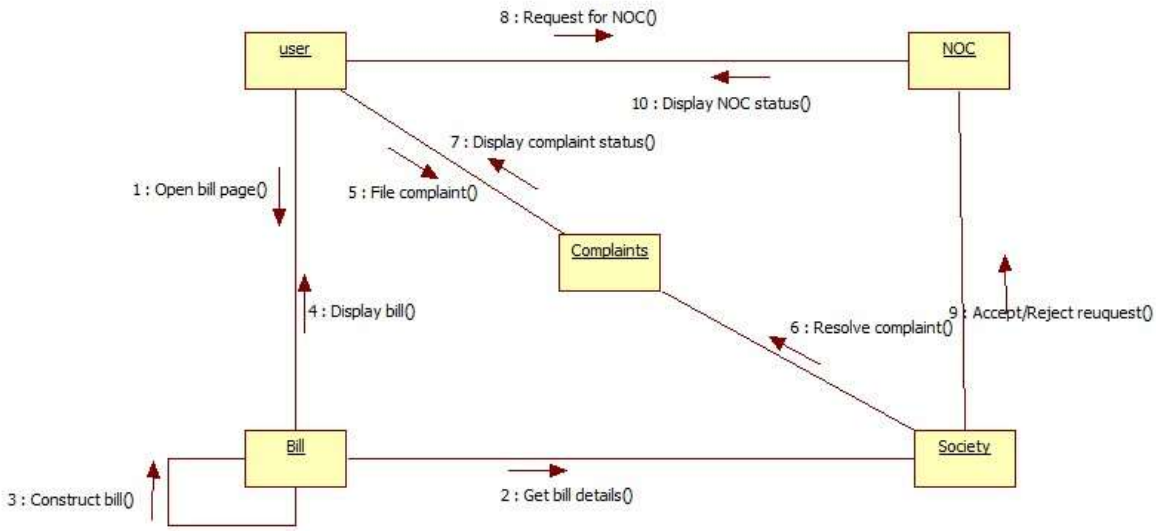

Fig.4.5 Communication diagram

Communication diagram specification

Sr No	Artifacts	Description
1	Objects	User, Society, Bill, Complaint, NOC
2	Messages	1: Open bill page 2: Get bill details 3: Construct bill 4: Display bill 5: File complaint 6: Resolve complaint 7: Display complaint status 8: Request for NOC 9: Accept/Reject request 10: Display NOC status

Table 4.4

4.2.6 SQUENCE DIAGRAM

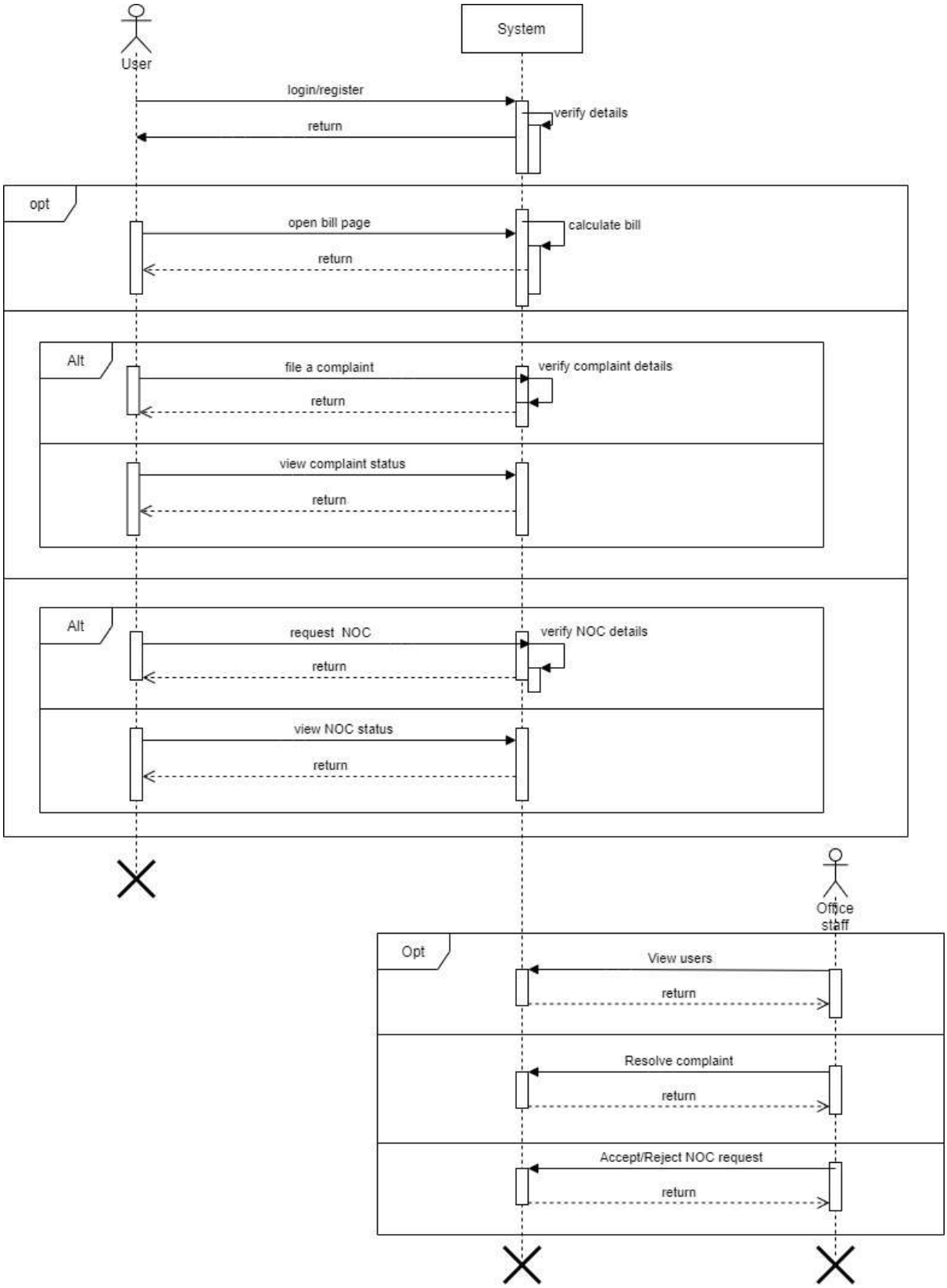

Fig 4.6 Sequence diagram

Sequence diagram Specification

Sr.No.	Artifacts	Description
1	Objects	**System**: The system will handle all the verifications and checks. It will display the available users, complaints and NOC requests to the staff.
2	Synchronous messages	All the messages are synchronous. When user sends a request to the system, they will wait for the system to process the request and return the message before proceeding further.
3	Asynchronous messages	-
4	Guard	When a user sends requests as **Login, Open bills page, file a complaint and request for NOC** the system will check the necessary conditions before returning a message.
5	Types of frame	**Opt:** Opt frame is used because the user has the option to view tabs as bills, complaints and NOCs. **Alt:** Alt frame is used when the user has to select one option from the available options. Here in complaints tab. user can either file a complaint or view the complaints they have filed. Similarly, for NOCs page.
6	Actors	**User:** Users of the system are society residents who can view their monthly bill on the application and they have options to file complaints and request for NOCs **Office staff:** The staff members can view all the users along with their wing and flat numbers. They can resolve complaints and accept or reject NOC requests.
7	Self-loop	Self-loop is used in several cases where the system has to verify some details.

4.2.7 COMPONENT DIAGRAM:

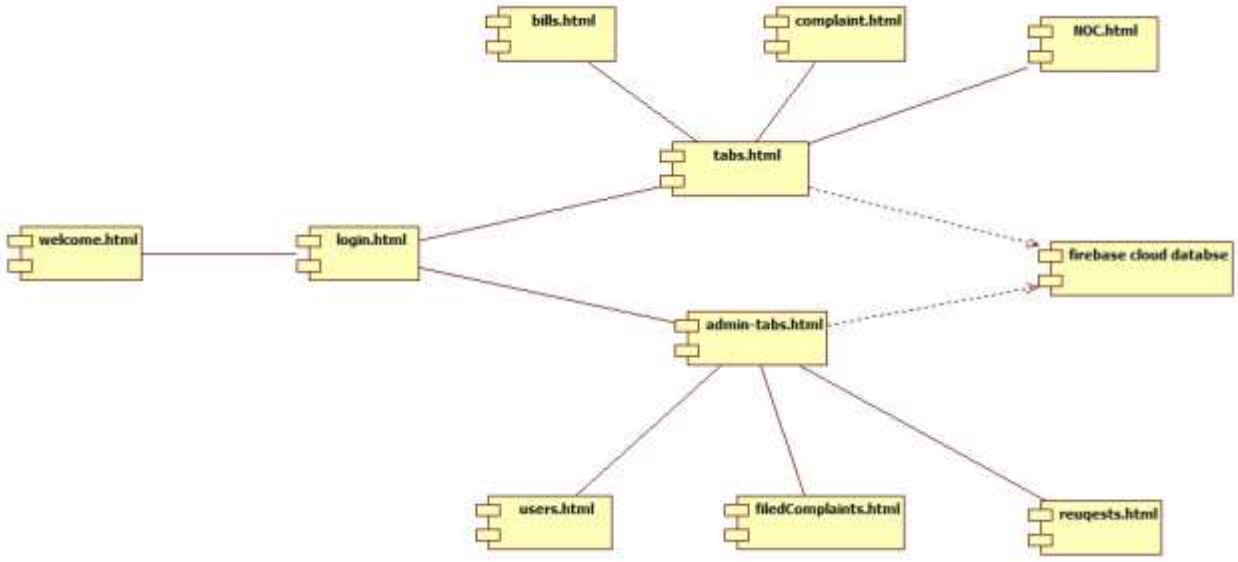

Fig 4.7 Component diagram

4.2.8 DEPLOYMENT DIAGRAM

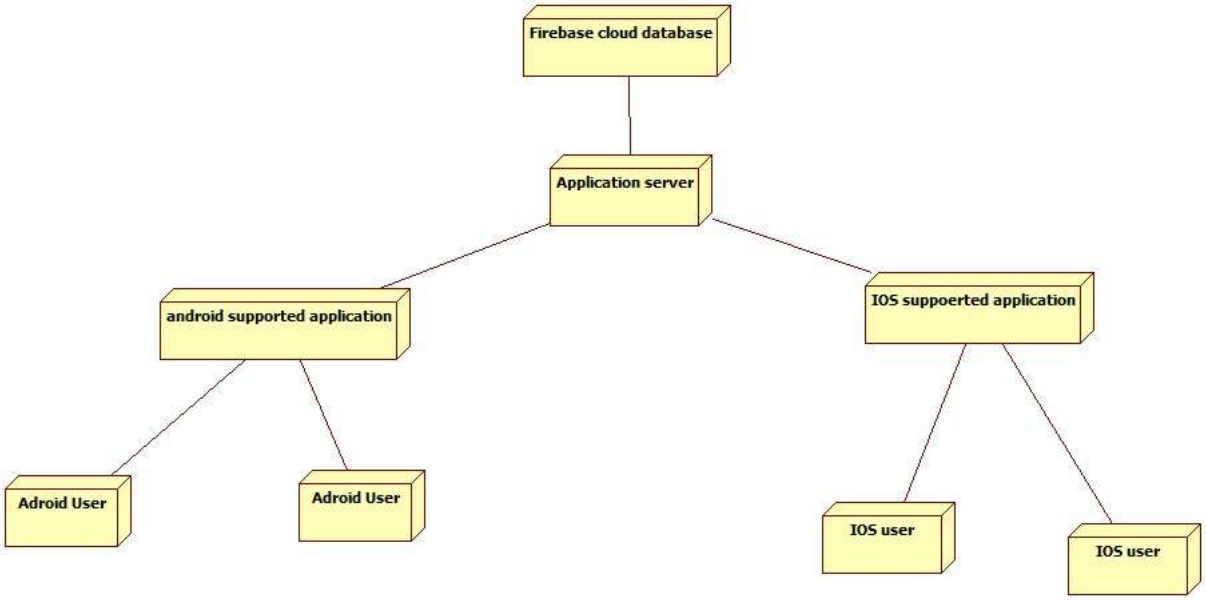

Fig 4.8 Deployment diagram

4.2.9 GANTT CHART

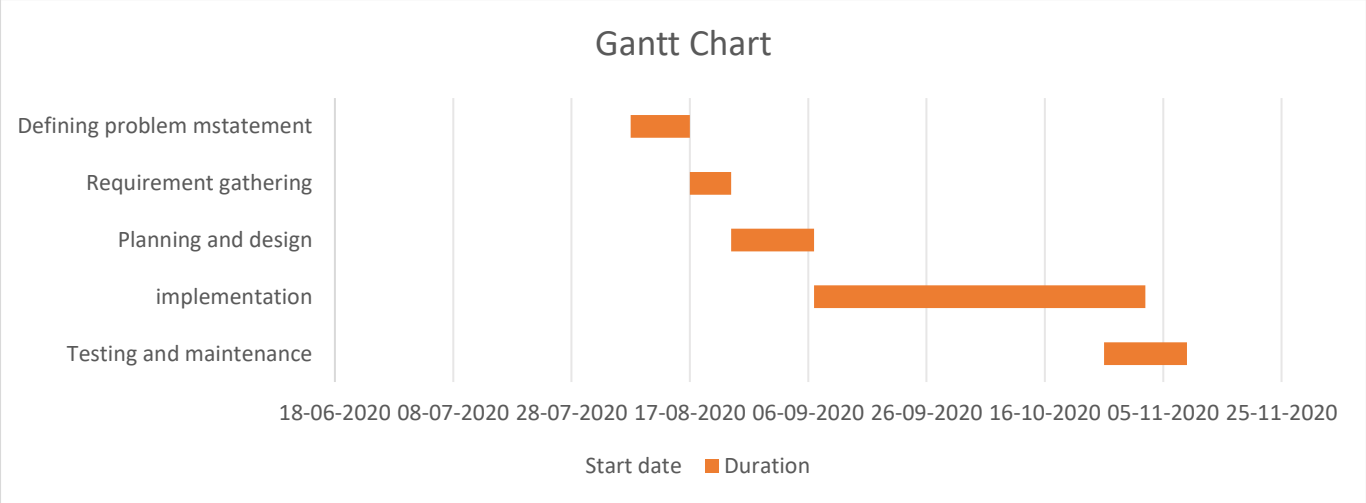

Fig 4.9 Gantt chart

4.2.10 PERT CHART

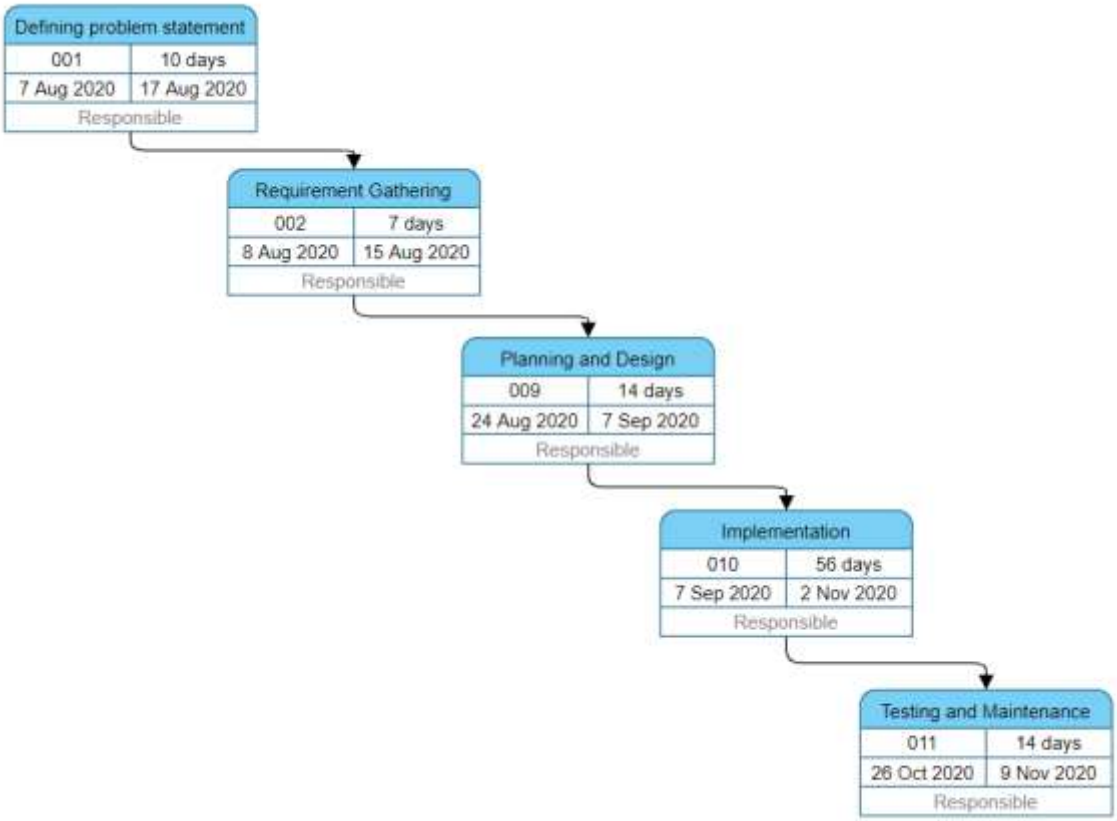

Fig 4.10 Pert chart

Chapter 5

PROJECT IMPLEMENTATION AND TESTING

The society maintenance system is implemented using ionic and angular framework. Where front end is designed by HTML and ionic UI components and back is done using TypeScript.

- Programming Languages used:

  TypeScript [backend]

  HTML [GUI]

  ionic UI components [GUI]

- Database:

  Firebase

5.1 CODE SNIPPET:

Global.service.ts

```
import { Injectable } from "@angular/core";
import { Plugins } from "@capacitor/core";

const { Storage } = Plugins;

export async function set(key: string, value: any): Promise<void> {
  await Storage.set({
    key: key,
    value: JSON.stringify(value)
  });
}
export async function get(key: string): Promise<any> {
  const item = await Storage.get({ key: key });
  return JSON.parse(item.value);
}
export async function remove(key: string): Promise<void> {
  await Storage.remove({
    key: key
```

```
  });
}

@Injectable({
  providedIn: "root",
})
export class GlobalService {
  public static userId;
  public static societyId;

};
```

Bill generation page

tab1.page.html

```
<ion-content [fullscreen]="true" >
  <ion-refresher slot="fixed" (ionRefresh)="getUserDetails($event)">
    <ion-refresher-content></ion-refresher-content>
  </ion-refresher>
    <ion-card color="primary" *ngFor="let bill of bills" >
    <div id="pdfTable" #pdfTable>
    <ion-card-header>
      <ion-card-title class="ion-text-center">{{bill.sname}}</ion-card-title>
    </ion-card-header>
      <ion-item>
        <ion-label>{{bill.month}}-{{bill.year}}</ion-label>
        <ion-button (click)="downloadPDF(bill.month,bill.year)">
          <ion-icon name="cloud-download" slot="end"></ion-icon>
          Download PDF
        </ion-button>
      </ion-item>
    <ion-card-content class="ion-no-padding">
      <ion-grid>
        <ion-row  >
          <ion-col size="8" class="ion-align-self-start">
            <ion-item color="dark" class="ion-no-margin">
            <ion-label>Details</ion-label>
            </ion-item>
          </ion-col>
```

```
          <ion-col class="ion-align-self-center">
            <ion-item color="dark">
            <ion-label>Rs.</ion-label>
          </ion-item>
          </ion-col>
        </ion-row>
      <ion-row  >
        <ion-col size="8" class="ion-align-self-start">
          <ion-item  class="ion-no-margin">
          <ion-label >Type Bill</ion-label>
          </ion-item>
        </ion-col>
        <ion-col class="ion-align-self-center">
          <ion-item class="ion-no-margin">
          <ion-label>{{bill.type}}</ion-label>
        </ion-item>
        </ion-col>
```

tab1.page.ts

bill generation function

```
async getUserDetails(event){
    let loader = this.loadingCtrl.create({
      message: 'Please wait...'
    });
    (await loader).present();
    try{

      this.nocGenerated = 0;
      this.hasComplaints = 0;
      // var day = generationDate.getDay();
      GlobalService.userId = await get('userId');

      console.log('Global ' +  GlobalService.userId);
```

```
      this.fireStore.firestore.collection('userDetails').doc(GlobalService.use
rId).get()
      .then(doc => {
        console.log('Document data:', doc.data()['societyName']);
        this.parkingNumber =  doc.data()['parkingVehicles'];
        this.nocGenerated = doc.data()['no_of_noc'];
        this.hasComplaints = doc.data()['no_of_comp'];
        this.isCommercial = doc.data()['isCommercial'];
        this.isOwner = doc.data()['isOwner'];

        this.extraDetails.name =  doc.data()['name'];
        this.extraDetails.flat =  doc.data()['flatNumber'];
        this.extraDetails.wing =  doc.data()['wing'];
        set('societyID',doc.data()['societyID']);

        this.userDetails =  [doc.data()].map(e => {
          return{
            name: e['name'],
            sname: e['societyName'],
            parking: e['parkingVehicles'],

          };

        });
        console.log('Outside :', this.userDetails[0]['sname']);
        });
        GlobalService.societyId  = await get('societyID');
      this.fireStore.firestore.collection('society').doc(GlobalService.society
Id).get()
      .then(doc => {
        var generationDate = new Date();
        this.todaysyear = generationDate.getFullYear().toString();
        this.todaysmonth = generationDate.toLocaleString('default', { month: '
short' });
        this.todaysmonthNumber = generationDate.getMonth();
        if (this.isCommercial == 'true'){
          this.catUser = doc.data()['commercial'];
        }
```

```
        else{
          this.catUser = doc.data()['normal'];
        }
        if (this.isOwner == 'true'){
          this.typeUser = doc.data()['owner'];
        }
        else{
          this.typeUser = doc.data()['tenant'];
        }
        this.sum = this.typeUser + this.catUser + doc.data()['water'] + doc.da
ta()['structure'] + doc.data()['complaint'] + doc.data()['parking'] + doc.data
()['noc'] + doc.data()['event'] + doc.data()['interest'];
        this.societyDetails =  [doc.data()].map(e => {
          return{
            type: this.typeUser,
            category : this.catUser,
            water : e['water'],
            struct: e['structure'],
            parking : e['parking'] *  this.parkingNumber,
            complaint : e['complaint'] * this.hasComplaints,
            noc : e['noc'] * this.nocGenerated,
            interest : e['interest'],
            sname : e['name'],
            event : e['event'],
            billsum : this.sum,
            month: this.todaysmonth,
            year: this.todaysyear,
            monthNumber: this.todaysmonthNumber,
          };

        });
        this.fireStore.collection('userDetails').doc(GlobalService.userId).col
lection('bills').doc(this.todaysmonth + ' ' + this.todaysyear).set({...this.so
cietyDetails[0]});
        this.fireStore.collection('userDetails').doc(GlobalService.userId).col
lection('bills', ref => ref.orderBy('monthNumber', 'desc')).snapshotChanges().
subscribe( data => {
          this.bills = data.map(e => {
```

```
            console.log('Type ' + e.payload.doc.data()['type']);
            return{
            type: e.payload.doc.data()['type'],
            category : e.payload.doc.data()['category'],
            water : e.payload.doc.data()['water'],
            struct: e.payload.doc.data()['struct'],
            parking : e.payload.doc.data()['parking'],
            complaint : e.payload.doc.data()['complaint'],
            noc : e.payload.doc.data()['noc'],
            interest : e.payload.doc.data()['interest'],
            sname : e.payload.doc.data()['sname'],
            event : e.payload.doc.data()['event'],
            billsum :  e.payload.doc.data()['billsum'],
            month:  e.payload.doc.data()['month'],
            year:  e.payload.doc.data()['year'],
          };
          });
        });
        });
      (await loader).dismiss();
    }
    catch (e){
      this.showToast(e);
    }
    setTimeout(() => {

      event.target.complete();
    }, 1000);

  }
```

Download pdf function

```
downloadPDF(month: string, year: string)
  {
    this.todaysyear = year;
    this.todaysmonth = month;
    console.log("got data "+ month + "  year" + year);
    var billID = this.todaysmonth + ' ' + this.todaysyear;
    console.log("Bill id" + billID);
    this.fireStore.firestore.collection('userDetails').doc(GlobalService.userI
d).collection('bills').doc(billID).get().then(data => {
      this.dataStore = [data.data()].map(e => {
        console.log('Herereererer ' + e['type']);
        return{
        type: e['type'],
        category : e['category'],
        water : e['water'],
        struct: e['struct'],
        parking : e['parking'],
        complaint : e['complaint'],
        noc : e['noc'],
        interest : e['interest'],
        sname : e['sname'],
        event : e['event'],
        billsum :  e['billsum'],
        month:  e['month'],
        year:  e['year'],
      };

      });
      console.log("In fetch "+ this.dataStore['sname']);
      this.createPDF(this.dataStore);
      this.downloadPDFFile();
    });
```

5.2 SNAPSHOT OF USER INTERFACE:

1. Instructions page

Fig 5.2.1 Instructions page

This is an instruction page for the users so that they can get an idea of how the application can be used, the features and the available components on the page.

2. Registration page

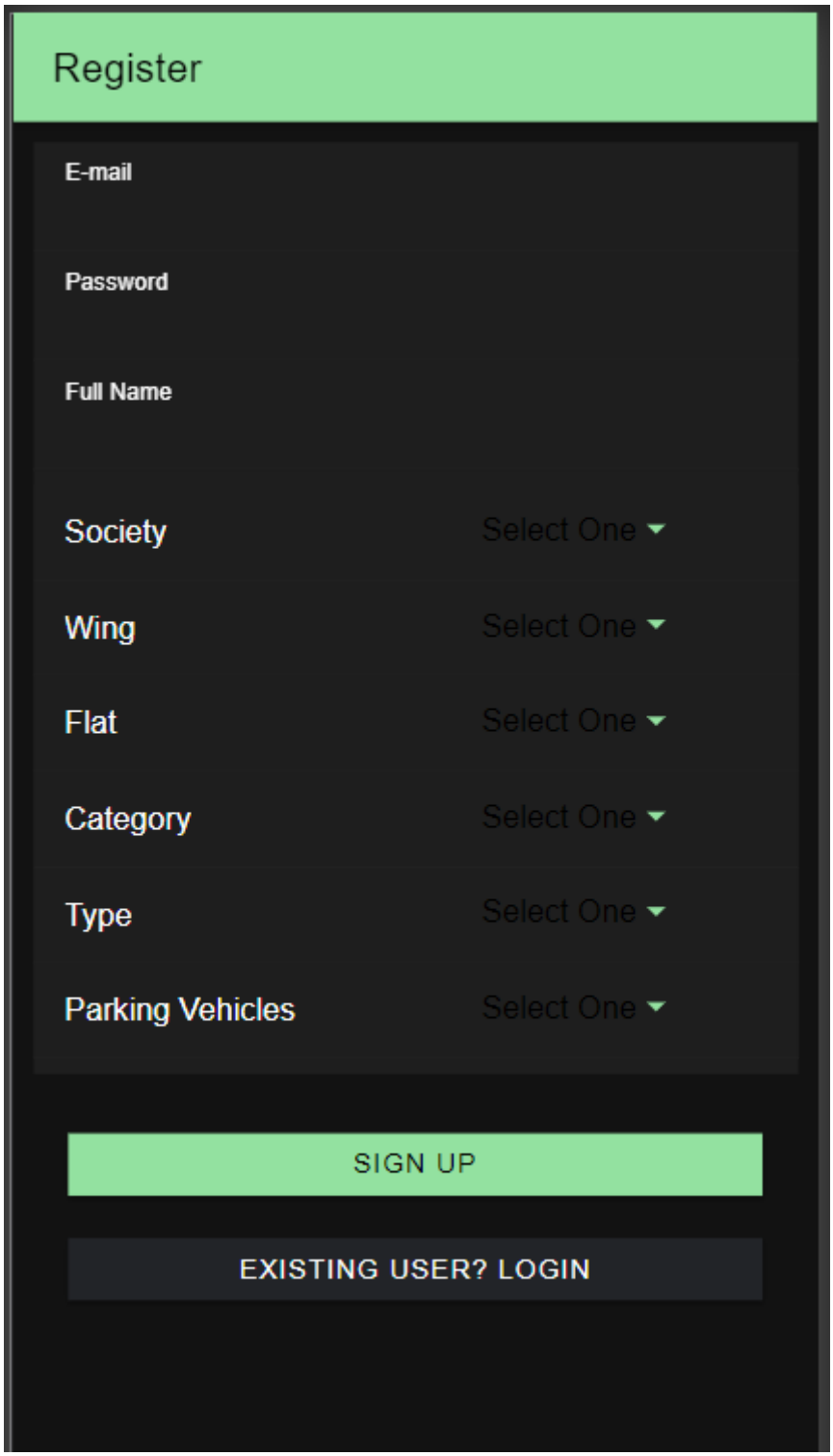

Fig 5.2.2 Registration page

This is the registration page where the users will have to select their society, wing and the flat number along with other details.

3. Login page

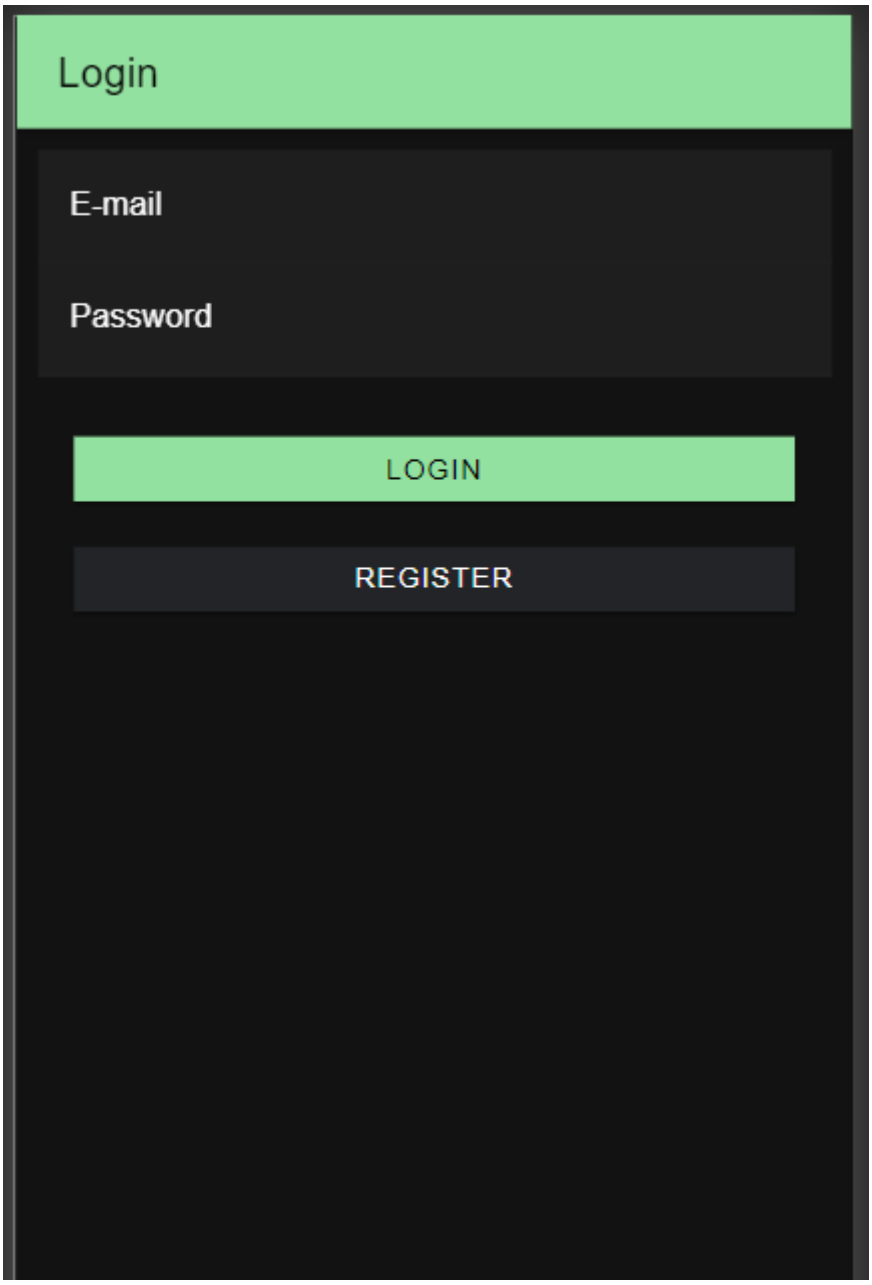

Fig 5.2.3 Login page

After registering once, the users can login simply by using their email and password.

4. Bills page

Fig 5.2.4 Bills page

The bills page will display the bill of the current month along with bills of previous months. The bill contains parameters like water charges, parking charges, society event charges, etc.

5. Complaints page

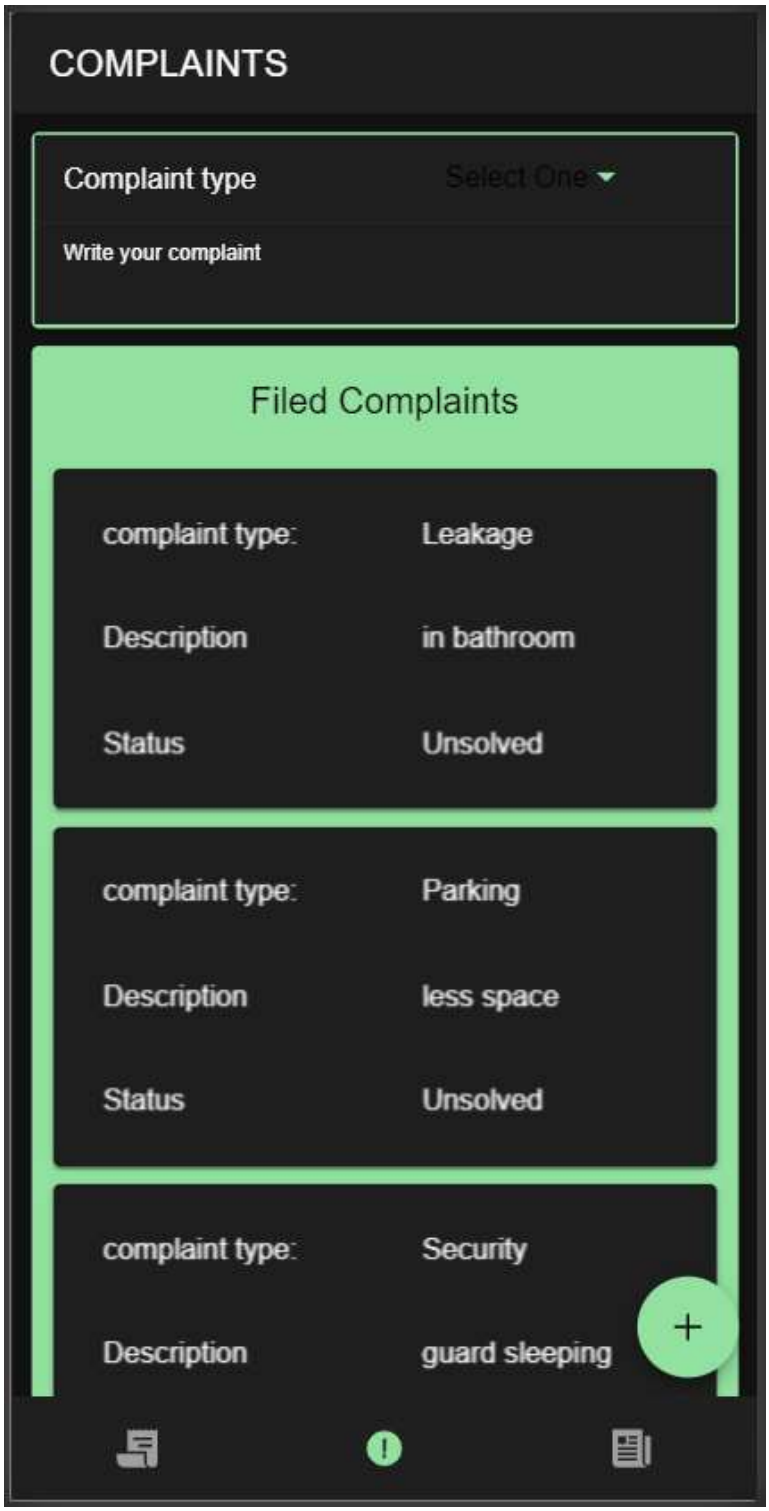

Fig 5.2.5 Complaints page

Using this page, the users can file complaints when they face any problems in the society or their house. E.g. leakage, security issue, etc. The files complaints will also be seen here.

6. NOC page

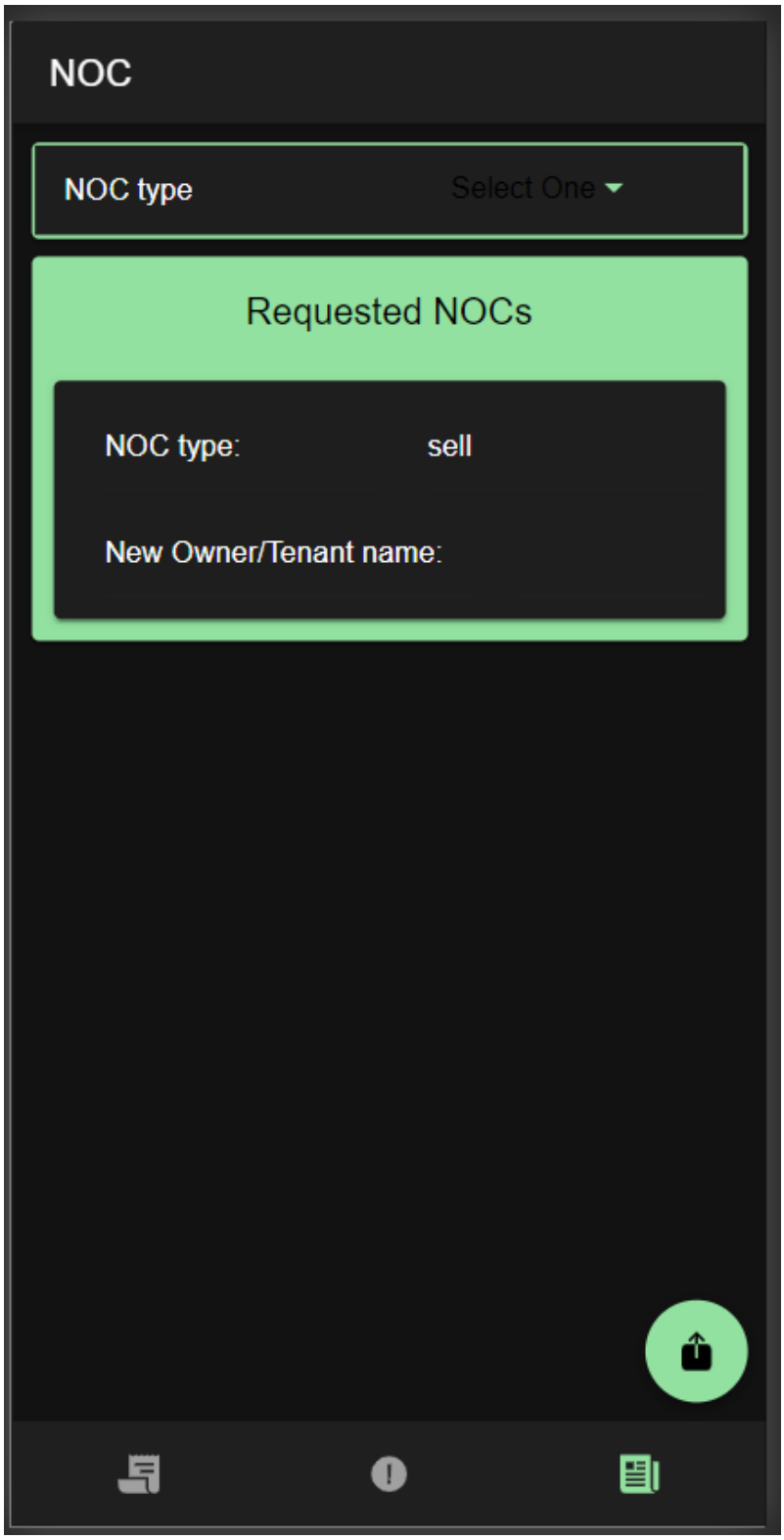

Fig 5.2.6 NOC page

At the time of changing flat or applying for loan, the users can request to the society for an NOC using this page.

7. Admin pages

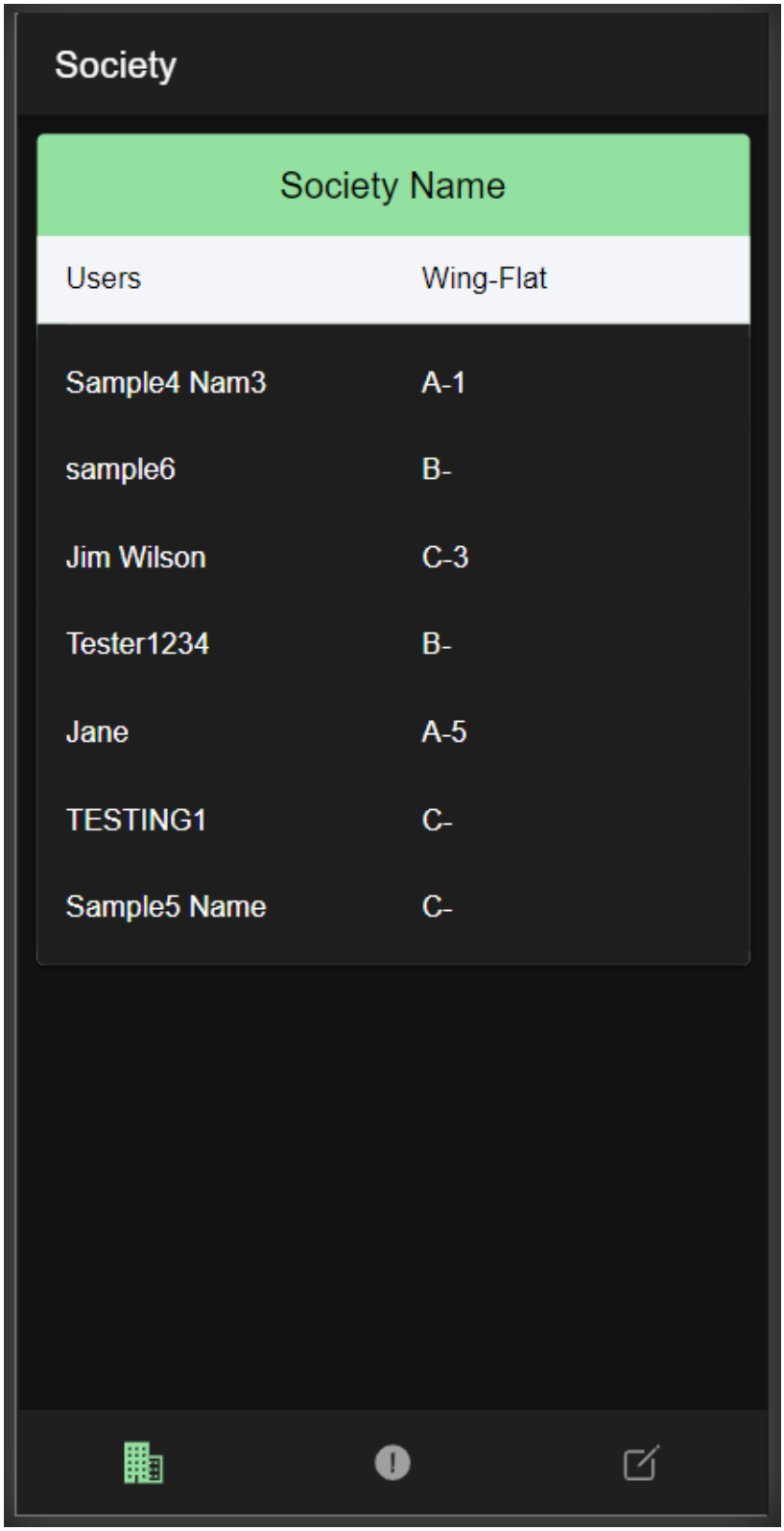

Fig 5.2.7 admin-Users page

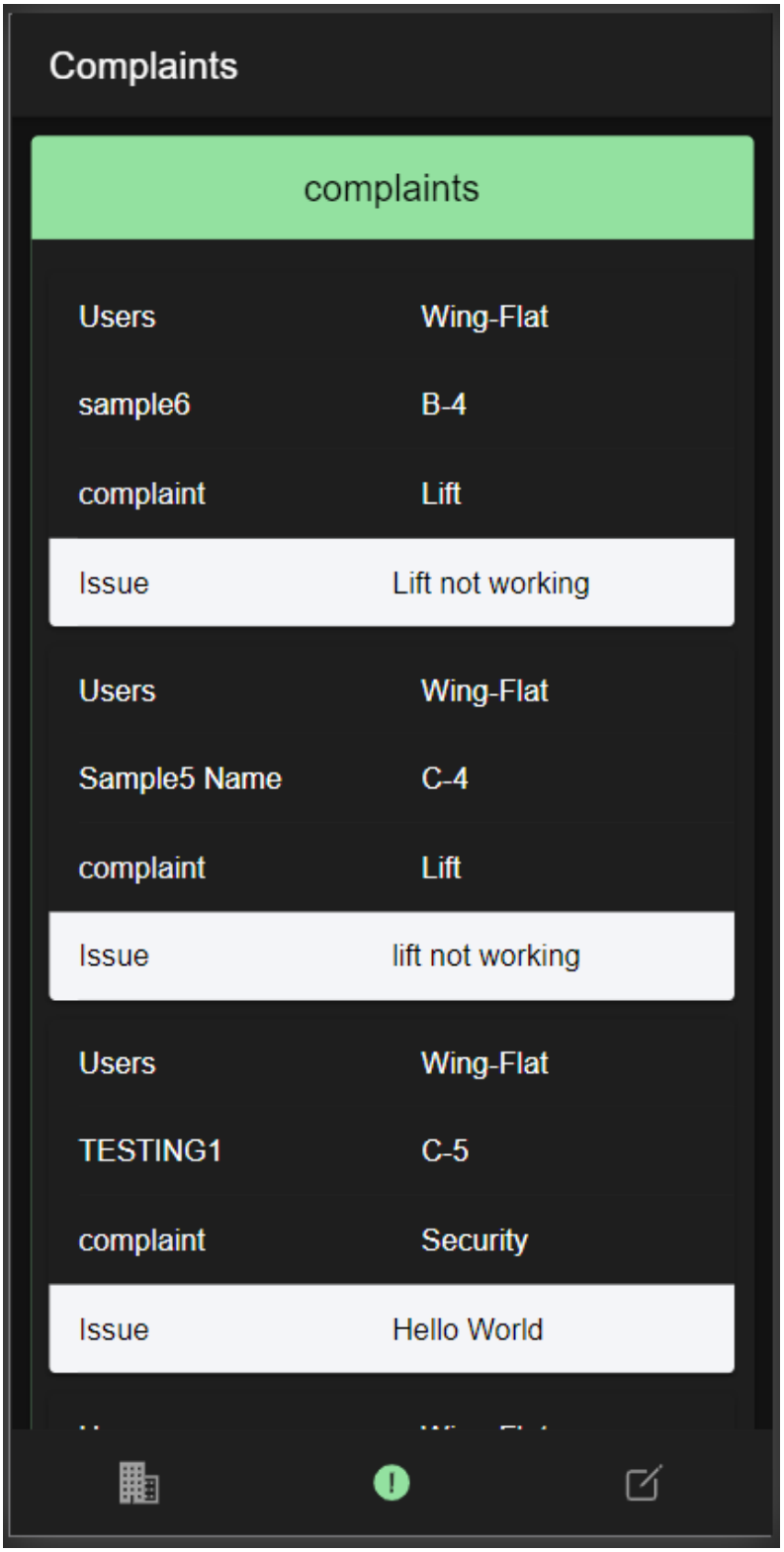

Fig 5.2.8 admin-complaints page

The admin pages let the admin view all the users in their society. They will also be able to see the filed complaints and will be able to resolve them.

5.3 TESTING AND TEST CASES:

5.3.1 TESTING:

Test case ID	Test data	Expected output	Actual output	Result
1	Entering registration data fields and registering the user	Registration successful, user document created, redirect to tabs(bill) page	Registration successful, user document created, redirect to tabs(bill) page	PASS
2	Registering with incorrect email format	Toast saying improper email id	Toast saying improper email id	PASS
3	Logging in by entering pre-registered email id and password	Login successful, redirect to tabs(bill) page	Login successful, redirect to tabs(bill) page	PASS
4	Login from unregistered email id	A login failed toast	A login failed toast	PASS
5	Displaying bill of this month.	Society name, all the parameters individually and total bill	Society name, all the parameters individually and total bill	PASS
6	Filing a complaint	Complaint should be added in the complaint collection	Complaint added in the complaint collection	PASS
7	Requesting for NOC	Select NOC type and add. A document should be made of the NOC request in the user's document	Document created	PASS
8	Viewing users list	On selecting a society, the admin should be able to see all the registered users	List displayed	PASS

Table 5.1

Chapter 6

SYSTEM MAINTENANCE

- Any queries related to downloading of bill or calculation mistake in the bills should be solved as soon as possible.
- Maintenance work is also carried out when the system fails to work properly.
- If any of the available module fails to work properly, it should be immediately taken out of the application and should be worked upon and put back in the application.
- Database should be refined time and time again if any vague entries are present.
- System Maintenance is needed when a new version of the existing software is released.

Chapter 7

FUTURE ENHANCEMENT

- The plan is to add more tabs for users such as organizing events.
- In future, the users will get an email when their monthly bill is generated.
- The users will be able to pay the bill online.

Chapter 8

LIMITATIONS

- The system does not have a payment gateway.
- The system has a time-consuming process to create a new society.

Chapter 9

CONCLUSION

- It helps the society secretary to handle and manage flat owner's data.
- It helps them manage society funds.
- It brings transparency and efficiency in the working of housing societies.
- It removes the overhead of maintaining documents regarding maintenance.
- It provides an automated notification system which allows the user to be alerted if the due date to pay the bill is upcoming.
- It allows users to apply for NOC and register complaints against other flat owners.
- It reduces the time required to address complaints physically and allows to fine the culprit in a no human interaction fashion.

Chapter 10

BIBLIOGRAPHY

- https://firebase.google.com/docs?gclid=Cj0KCQiAhs79BRD0ARIsAC6XpaUVwOIPCkThbPyI_iEtnZXt9-X0I4hboxLcW5ko-iqXZlM6OSOBEbMaAtQhEALw_wcB

- https://www.tutorialspoint.com/firebase/firebase_read_data.htm

- https://stackoverflow.com/questions/34458930/most-efficient-way-to-increment-a-value-of-everything-in-firebase

- https://fireship.io/snippets/firestore-increment-tips/

- https://ionicframework.com/docs/components

- https://ionicframework.com/docs

- https://angular.io/docs

- https://www.javatpoint.com/ionic-slides

- https://firebase.google.com/docs/auth

- https://www.positronx.io/how-to-connect-firebase-realtime-nosql-cloud-database-with-angular-app-from-scratch/#:~:text=Setup%20Google%20Firebase%20Database%20Account,-Go%20to%20Firebase&text=Once%20your%20project%20is%20set,your%20firebase%20credentials%20from%20here.