A

Project Report

On

"Friendly Neighborhood"

IN PARTIAL FULFILLMENT
OF

MASTER OF COMPUTER APPLICATION (MCA)

BY

Ms. Shivani Mukesh Awatade

MCA- II SEM-III

(2022-23)

Guided by

Dr. Milind Godase

Submitted To

SAVITRIBAI PHULE PUNE UNIVERSITY

SINHGAD INSTITUTE OF MANAGEMENT,

Vadgaon Bk, Pune-411041

Date: -

CERTIFICATE

This is to certify that Ms. Shivani M. Awatade, has successfully / partially completed her project work entitled **"FRIENDLY NEIGHBORHOOD"** in partial fulfillment of MCA-II SEM-III Mini Project for the year 2022-2023. She has worked under our guidance and direction.

All help received by her from various sources have been duly acknowledged. No part of this project has been submitted elsewhere for award of any other degree.

Dr. Milind Godase
Project Guide

Dr. Chandrani Singh
Director SIOM - MCA

Examiner 1

Examiner 2

Date:
Place:

DECLARATION

I certify that the work contained in this report is original and has been done by me under theguidance of my supervisor(s).

- The work has not been submitted to any other Institute for any degree or diploma.
- I have followed the guidelines provided by the Institute in preparing the report.
- I have conformed to the norms and guidelines given in the Ethical Code of Conduct of theInstitute.
- Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and givingtheir details in the references.

Name and Signature of Project Team Members:

Sr. No.	Seat No.	Name of students	Signature of students
1	20222	Shivani Mukesh Awatade	

ACKNOWLEDGMENT

I have immense pleasure in expressing my sincerest and deepest sense of gratitude towards my guide Dr. Milind Godase for the assistance, valuable guidance and co- operation in carrying out this Project successfully. I have developed this project with the help of Faculty members of my institute and I am extremely grateful to all of them. I also take this opportunity to thank Dr. Chandrani Singh, Director SIOM - MCA, for providing the required facilities in completing this project. I am greatly thankful to my parents, friends and faculty members for their motivation, guidance and help whenever needed.

Thank You,
Shivani Mukesh Awatade

INDEX

CHAPTER 1: INTRODUCTION

1.1 Abstract

"Friendly Neighborhood" is an android application introduced with an objective of handling day-to-day activities of residential societies. It is a society and accounting management system for apartments and housing societies. It provides a platform for users to perform activities like paying maintenance, file a complaint in the society's forum, club house reservation, events section and much more. It will ease the efforts for owners as well as the society's secretary and will save their time and efforts. It also eliminate a lot of paper work and make things hassle-free.

This application offers something to each stakeholder in a society's daily affairs. From residents to management committee, this app has something for everyone. It is an efficient and comprehensive system that will make your life super convenient and secure. The objective of this project is to reduce human-efforts and save time.

This project is divided into the modules as follows:

- **Login/Registration:** The user's data-privacy is always a priority; and hence to enhance the user experience and keep the user data safe, this project includes the login/signup for the authentication purpose. The user data is password protected and only can be accessed with valid credentials.
- **Forum:** Stay updated with all the society related information, announcements and complaints made by other residents with a digital forum. You can connect with neighbors using forum's comment section and have a hassle-free communication experience.
- **Payments:** Using this application one can make and track maintenance and utility payments all in one place. You can also check your payment history based on the type of payment you choose to make (monthly, quarterly or yearly). Once the payment has been done the admin or secretary receives it and the notification is generated from the same platform. Accounts and billing solution for management committee/ admin so they can manage everything related to society's finances. From book keeping and fine management to managing the records, it's all here. Management committee members can manage maintenance and utility bill payments easily with automatic generation of invoices and receipts.
- **Events:** With the events section, the management committee can create and host events for the societies. Users can view the events and be informed about its details timely.
- **Clubhouse Reservation:** Using the reservation section, one can choose to reserve the society clubhouse for hosting personal events based on the availability. All of this can be done with at most ease using a same platform thus enhancing the user experience and promoting convenience.
- **Admin Module:** Through this section the super user activities can be implemented. This section has been developed by keeping the managing committee of the society in mind. They can receive notifications about payments, reservations, complains and can create and post an event happening in the society.

1.2 Existing System and Need for System

- **Existing System:** The current system is manual. It has a high risk of data loss and includes a lot of paper work and maintenance. The present system does contribute itself tothe world but not in all every way. It lacks detailed and accurate information all at one place. One may or may not get the desired part of information as it isn't available at singleplatform. The residents has to hustle through society office to a personal from maintenance committee for gathering information. The Secretary has to send physical notices to every resident's house and it may even get damaged while moving from one house to another. It doesn't even ensure if the information has reached every resident.

  The current system lacks convenience. The data isn't saved and maintained properly and the user's feedback isn't taken care of. Many such flaws are needed to overcome to develop anideal website which will mark its share for making things convenient for users. One can't check with the notice board every day for being updated about society's internal affairs.

  Making a manual system into computerized system is not an easy task, especially due to the complexity of work it involves. To serve better to users is the aim of every management,but the present working system doesn't seems to work properly.

  Following are the flaws of current system:

- A resident's personal information can be leaked by any one even the secretary's irresponsibility so securityis not so ensured in manual form.
- Working manually is always hectic and time consuming. If the resident has a complaint regarding certain issue, he/she cannot always physically go / call regarding it as receiving so many complains at the same time from so many residents is not possible and inconvenient at the same time.
- Saving and maintaining data with accuracy is a difficult task. The maintenance receipts, fines and the generated bills all contribute to a lot of paper work and make things complex to be managed.
- It is difficult to be updated about the events and the announcement made by the society's managing committee.
- Reserving the clubhouse is another difficult task that includes paper work. Checking for availability in such cases is hectic task.

- **Need for System**: A system which will prevent data loss and can be managed easily isrequired. It will hold data in the database which can be manipulated or viewed easily when required. In order to drive the objective of the project, the system is selected in such a wayso that it would satisfy all the requirements of the project. It also helps to get the desired output.

  In today's world of technology where every aspect of the life is computerized so that the system used should be efficient and accurate. If we use the old techniques, we can access only a limited amount of a data and accuracy of the obtained result isn't assured. A platformwhich is completely designed for maintaining user data and maintenance statistics is the need ofthe hour because it will facilitate the users maintaining data of interest. The new system isrequired to save resources such as time and manpower, which are valuable in the present scenario of system. The new system helps us handle large database in a smooth manner and also makes the updation of data very easy.

  This project will be user friendly. It will provide quick performance application and provide correct output. Accuracy and efficiency is an ordinary function of this project.

1.3 Scope of System

- **Current System:** Currently this project is designed as an android application. It has been developed using Android Studio tool in java. Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.
- **Future Scope:** The aim of this project is to create an integrated platform for people living in residential societies. This application can further be extended with features like security check and identity verification etc. This application can be easily extended with elaborated features in future.

1.4 Operating Environment Hardware and Software

- **Server side requirement**
  - **Hardware requirements**
    - **Processor** **:** Intel Core i5 and onwards.
    - **RAM** **:** 4GB
    - **HDD** **:** 20GB
  - **Software requirements:**
    - **Operating System :** Windows 10
    - **Database: (**Firebase**)** NoSQL
    - **Front End :** XML (Extensive Markup Language)
    - **Back End :** JAVA
    - **Software Development Tool** : Android Studio Tool
- **Client side requirement**
  - **Hardware requirements**
    - **Device:** Smart Phone
    - **RAM :** 2GB
  - **Software requirements**
    - **Operating System:** Android 5.0 and above
    - **Browser:** NA

1.5 Brief Description of Technology used

1.5.1 Operating Systems used

- **For Development:** Windows is the most popular and commonly used Operating System for development and testing. It is easily available and easy to install as well. For this project, Windows 10 OS has been used for development. With the help of JAVA programming language and an IDE called Android Studio Tool.

  Windows 10 is the most recent version of the operating system from Microsoft. Officially it was released in 2015 and was initially offered free of charge to legitimate users of Windows 7 and Windows 8.1. This new version combines features from those two previous installments to suit the users in a better way for both desktop/laptop computers as well as mobile devices. The most notable change in Windows 10 is that Microsoft replaced the Start screen tiles from Windows 8, and brought back the Start Menu. They also removed the vertical toolbars (or "charms") that appeared from the sides of the screen. These changes make this Windows version easier to use for users of both desktop/laptops and mobile devices.

- **For Project Execution:** This project has been developed for smart phones with android operating system. Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.

  It is free and open-source software. Its source code is Android Open Source Project (AOSP), primarily licensed under the Apache License. However, most Android devices dispatch with additional proprietary software pre-installed, mainly Google Mobile Services (GMS), including core apps such as Google Chrome, the digital distribution platform Google Play and the associated Google Play Services development platform.

  This Project is supported on devices with Android 5 OS and onwards.

1.5.2 RDBMS/NoSQL used to build the database

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

NoSQL databases (aka "not only SQL") are non-tabular databases and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.
Each NoSQL database has its own unique features. At a high level, many NoSQL databases have the following features:

- Flexible schemas
- Horizontal scaling
- Fast queries due to the data model
- Ease of use for developers

While a variety of differences exist between relational database management systems (RDBMS) and NoSQL databases, one of the key differences is the way the data is modeled in the database. Many other important differences exist, including:

- Flexibility of the schema
- Scaling technique
- Support for transactions
- Reliance on data to object mapping

The Firebase Real-time Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real-time. Following are some of its characteristics:

- **Collaborate across devices with ease**

Real-time syncing makes it easy for your users to access their data from any device: web or mobile, and it helps your users collaborate with one another.

- **Build server less apps**

Real-time Database ships with mobile and web SDKs so you can build apps without the need of servers. You can also execute backend code that responds to events triggered by your database using Cloud Functions for Firebase.

- **Optimized for offline use**

When your users go offline, the Real-time Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.

- **Strong user-based security**

The Real-time Database integrates with Firebase Authentication to provide simple and intuitive authentication for developers. You can use our declarative security model to allow access based on user identity or with pattern matching on your data.

CHAPTER 2: PROPOSED SYSTEM

2.1 Feasibility Study

1. Technical Feasibility

The system requirements for launching this application are basic and can be seen in majority of smart phones. Android is a very popular and commonly available OS for smartphones. The development tool and framework are easily available and accessible in market.

2. Economic Feasibility

By collaborating with other websites or by publishing this application on the App stores, revenue can be generated. It can be generated using this app as an advertising platform as well. We can also launch some premium features with this application that users can access by paying a certain amount.

3. Operational Feasibility

This app will efficiently provide all the necessary features at one platform and its UI will be user friendly. It will be easy to understandand use for the fellow users. Mobile devices are increasing day by day because of its portable features and effective applications. Nowadays not only individuals but entrepreneurs also depend on portable devices for their small and big assignments. However, rapid use of mobile technology increasing feasibility of mobile users and programmers.

2.2 Objectives of the proposed system

- **A platform to share opinions**

  This project provides a common platform for the people living in a same residential society to socialize and connect. It provides a forum for the residents to complain/announcement or receive the same from other residents or the management committee. It provides a space for raising complaints regarding society'sinternal affairs and operations.

- **Review platform**

  This platform enables the secretary to view and solve the resident's complaints atone place. They can also update regarding it solutions conveniently. With this features all of the complaints can be received and addressed at one place and it becomes easy for the admin to view and solve them all. It also eliminates time and effort and reduces paper work.

- **Time saving**

  This application is saving time for residents by eliminating the need to physically visit the society office regarding any issue or pending maintenance and utility payments. One can also post a complaint and receive and announcement through this app without having to visit society office or notice board. Thus saving a lot of time for users.

- **To eliminate manual system**

  It eliminates the need of paper work and saves time. In the current system, the users have to submit hardcopies of documents or receive hardcopies of maintenance receipts which has a risk of being misplaced. The admin also has to send hard copy of important announcements and events which is time consuming. All of this can be put at ease using this platform.

- **Enhanced user experience**

  This application provides a common platform for all the basic day to day activities of a residential society. Be it posting a complaint, viewing an event, maintenance and utility payments or club house reservation all of this can be done under a common place and thus enhances user experience.

2.3 Users of the system

Residents: The residents have access to the dashboard containing the following:

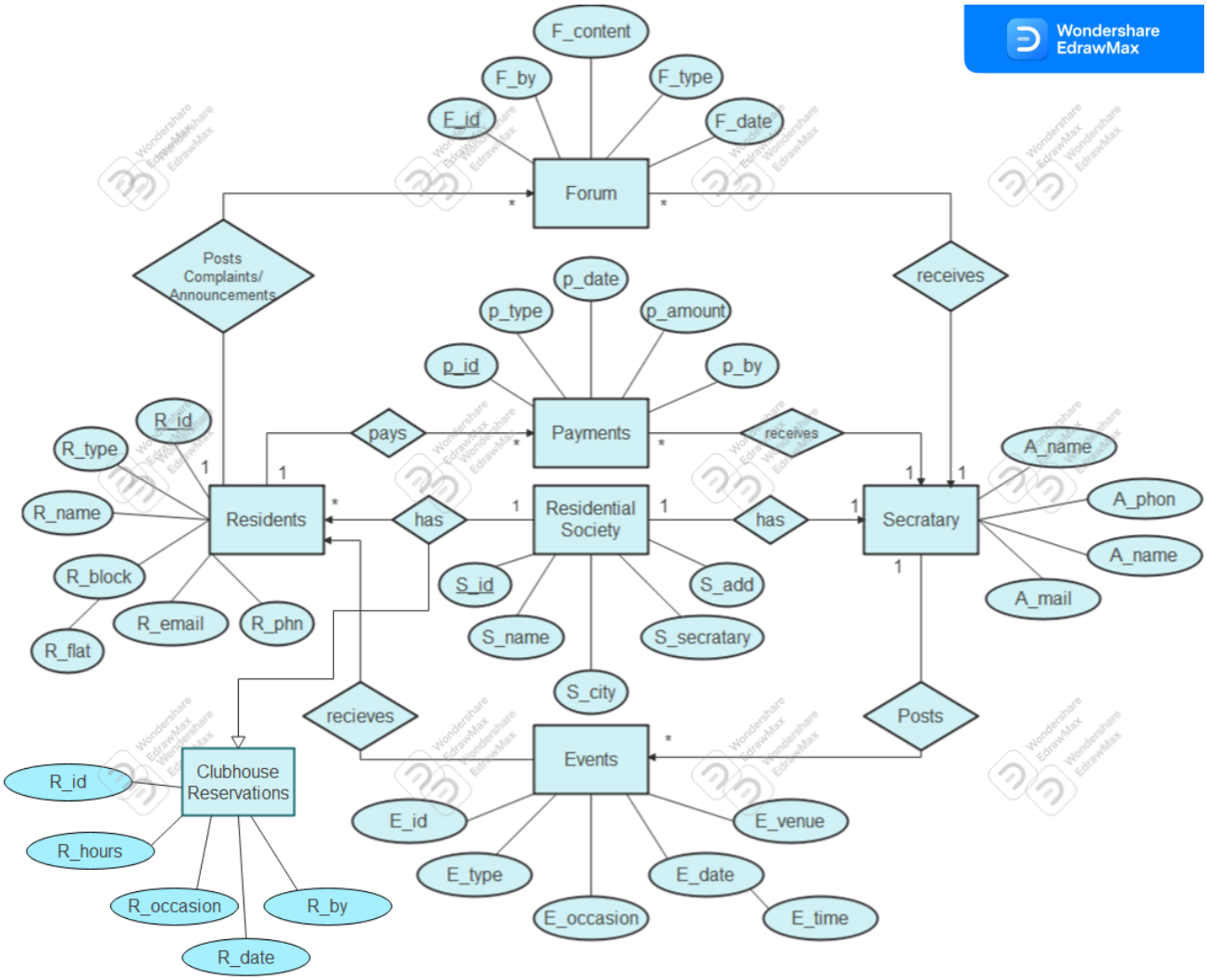
- Forum: Here the residents can browse the announcements made by the maintenance committee and complains raised by other residents. He/she can also choose to respond to the above via comments. The user can also post a complaint/announcement if he wishes to.
- Payments: Here the residents can pay all the maintenance and utility payments to their respective society. The user first has to select the type of payment he/ she has to make and the app will direct them to the payment gateway. One can also view the generated bill/receipt.
- Reservations: Here the residents can check the availability of the clubhouse on the required date and time and can send the reservation request accordingly.
- Events: Here the residents can view the events that have been posted by the committee and can receive and accept invites for the same.

Admin/Secretary: The admin has access to the dashboard containing the following:

- Forum: Here the admin can receive complaints raised by residents and address them. He/she can also choose to respond to the above via comments. The admin can also post an announcement if he wishes to.
- Payments: Here the admin can receive all the maintenance and utility payments to the residents of the society. The admin can also send alerts for the pending payments. He can also send the generated bill/receipt.
- Reservations: Here the admin can receive the request for the reservation of the clubhouse and can choose to accept/deny it.
- Events: Here the admin can post the events that will be conducted in the society and can send the invites for the same.

CHAPTER 3: ANALYSIS AND DESIGN

3.1 Entity Relationship Diagram (ERD)



3.2 Table Structure

- Database name: Fneighbor
- Table name : Society

Column_name	Data type	Size	Description
S_id	integer	3	Society id
S_name	varchar	25	Society Name
S_city	integer	5	Society city
S_secretary	varchar	10	Society's secretary
S_add	varchar	80	Society Address

- Table name : Resident

Column_name	Data type	Size	Description
R_id	integer	3	Resident id
R_name	varchar	150	username
R_password	varchar	128	password
R_type	varchar	10	If user is owner/tenant
is_superuser	bool	-	If user is admin/secretary
R_block	varchar	2	Block that user lives in
R_flat	integer	3	Flat that user lives in
R_contact	integer	10	Resident contact
R_mail	varchar	50	Resident email

❖ Table name : Secretary

Column_name	Data type	Size	Description
A_id	integer	3	Secretary id
A_name	varchar	150	username
A_password	varchar	128	password
A_contact	integer	10	Secretary contact
A_mail	varchar	50	Secretary email

❖ Table name : Forum

Column_name	Data type	Size	Description
F_id	integer	3	Forum id
F_by	varchar	25	Complaint/announcement posted by
F_content	text	300	Complaint/announcement content
F_type	varchar	10	Forum type
F_date	datetime	-	Complaint/announcement date

❖ Table name : Payments

Column_name	Data type	Size	Description
P_id	integer	3	Payment id
P_by	varchar	25	Payment done by
P_amount	integer	5	Amount
P_type	varchar	10	Payment/type
P_date	datetime	-	Payment date/time

❖ Table name : Events

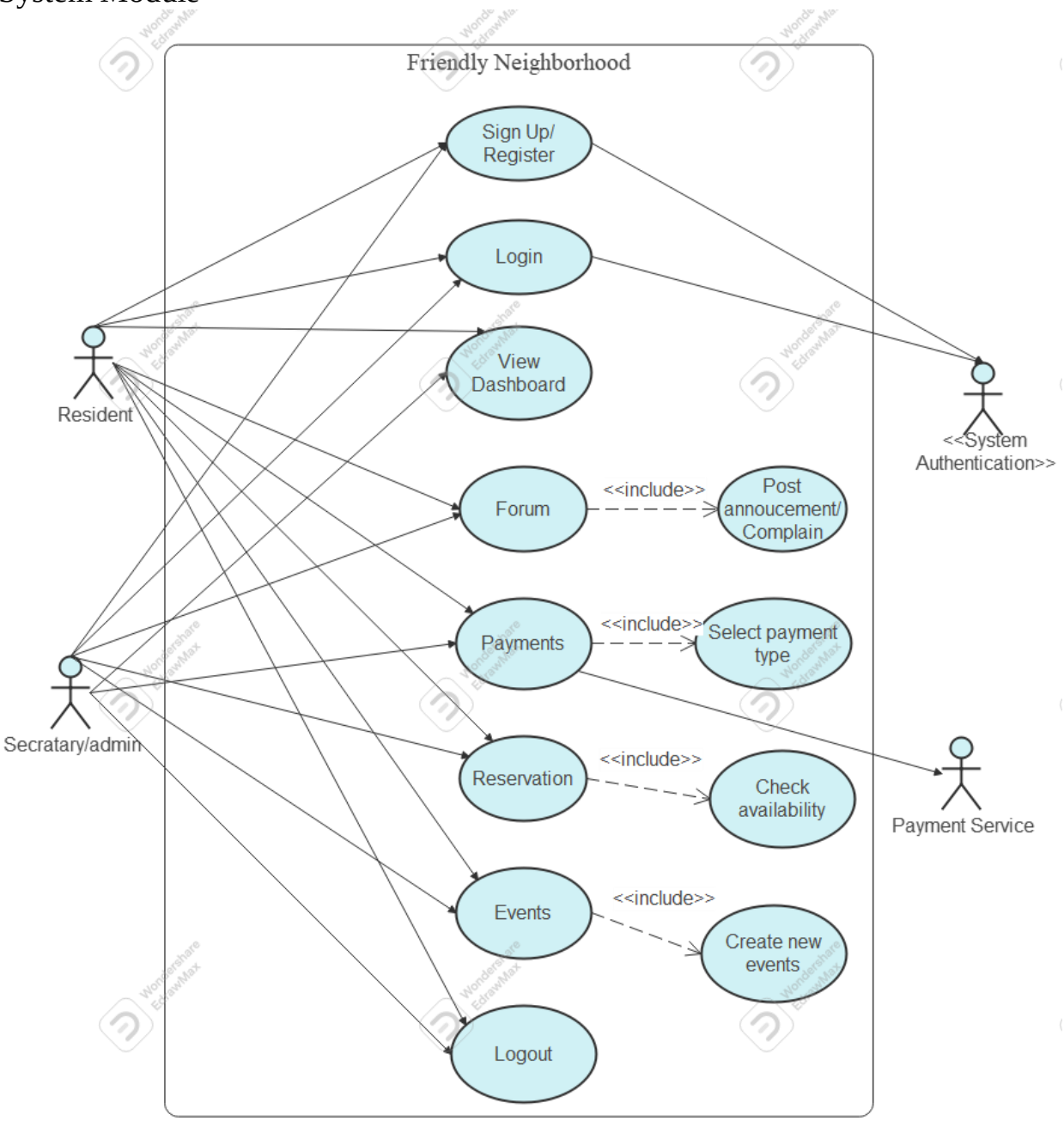
Column_name	Data type	Size	Description
E_id	integer	3	Event id
E_type	varchar	25	Event Type
E_occasion	text	150	Event occasion
E_venue	varchar	20	Event venue
E_date	datetime	-	Event date/time

❖ Table name : Reservations

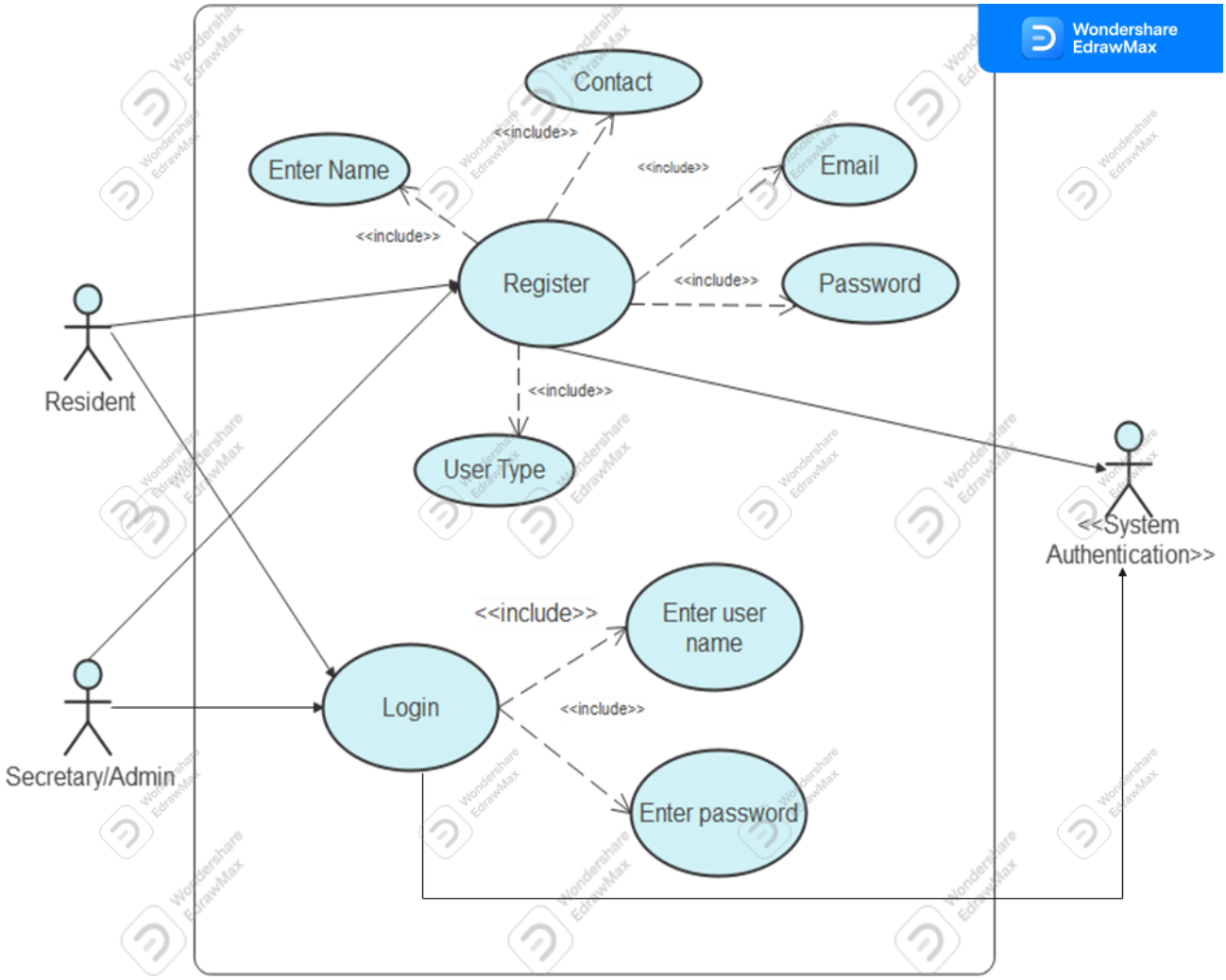
Column_name	Data type	Size	Description
R_id	integer	3	Reservation id
R_hours	varchar	10	No. of hours clubhouse reserved for
R_occasion	text	150	Occasion of reservation
R_date	datetime	-	Reservation date/time
R_by	varchar	30	Reservation done by

3.3 Use case diagrams

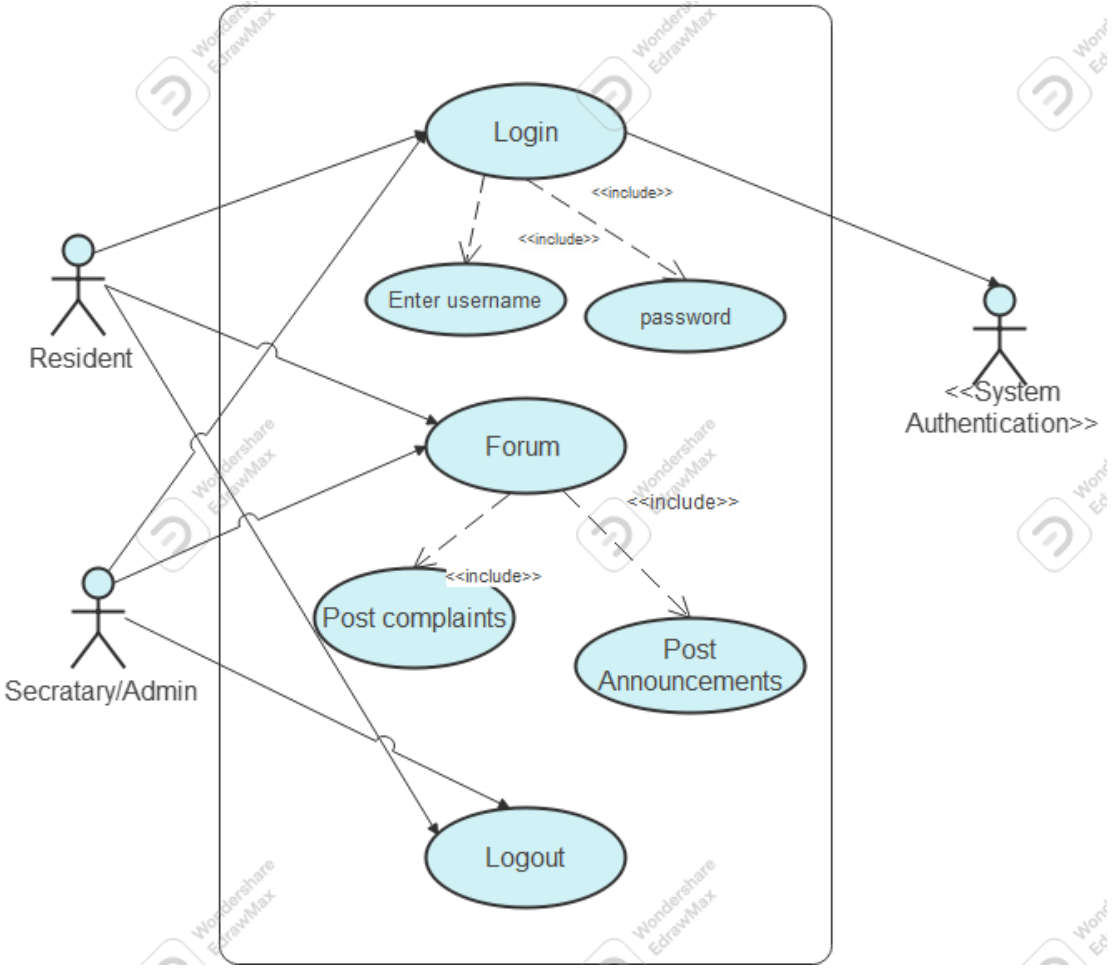
- System Module



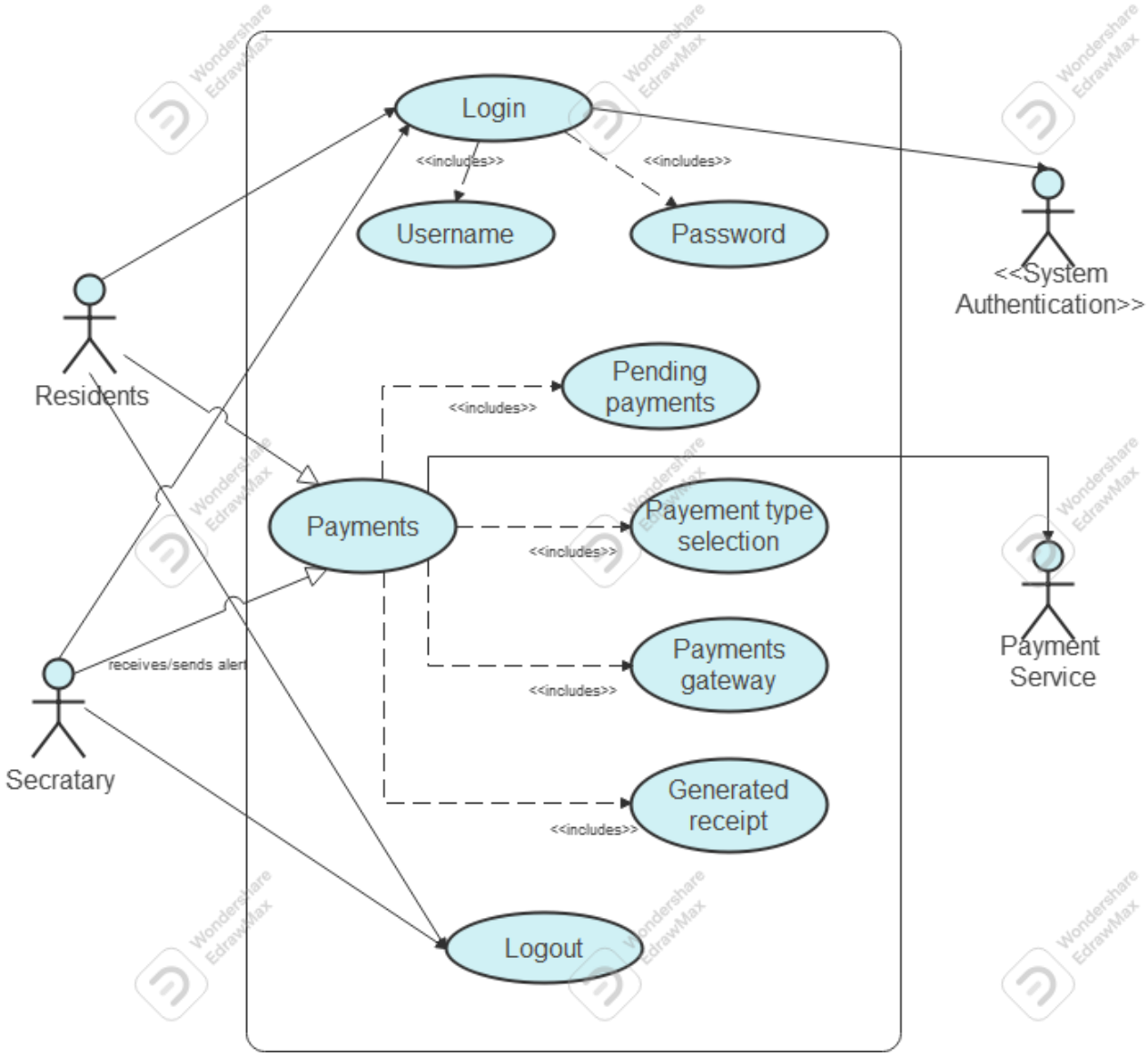
❖ Registration/Login Module



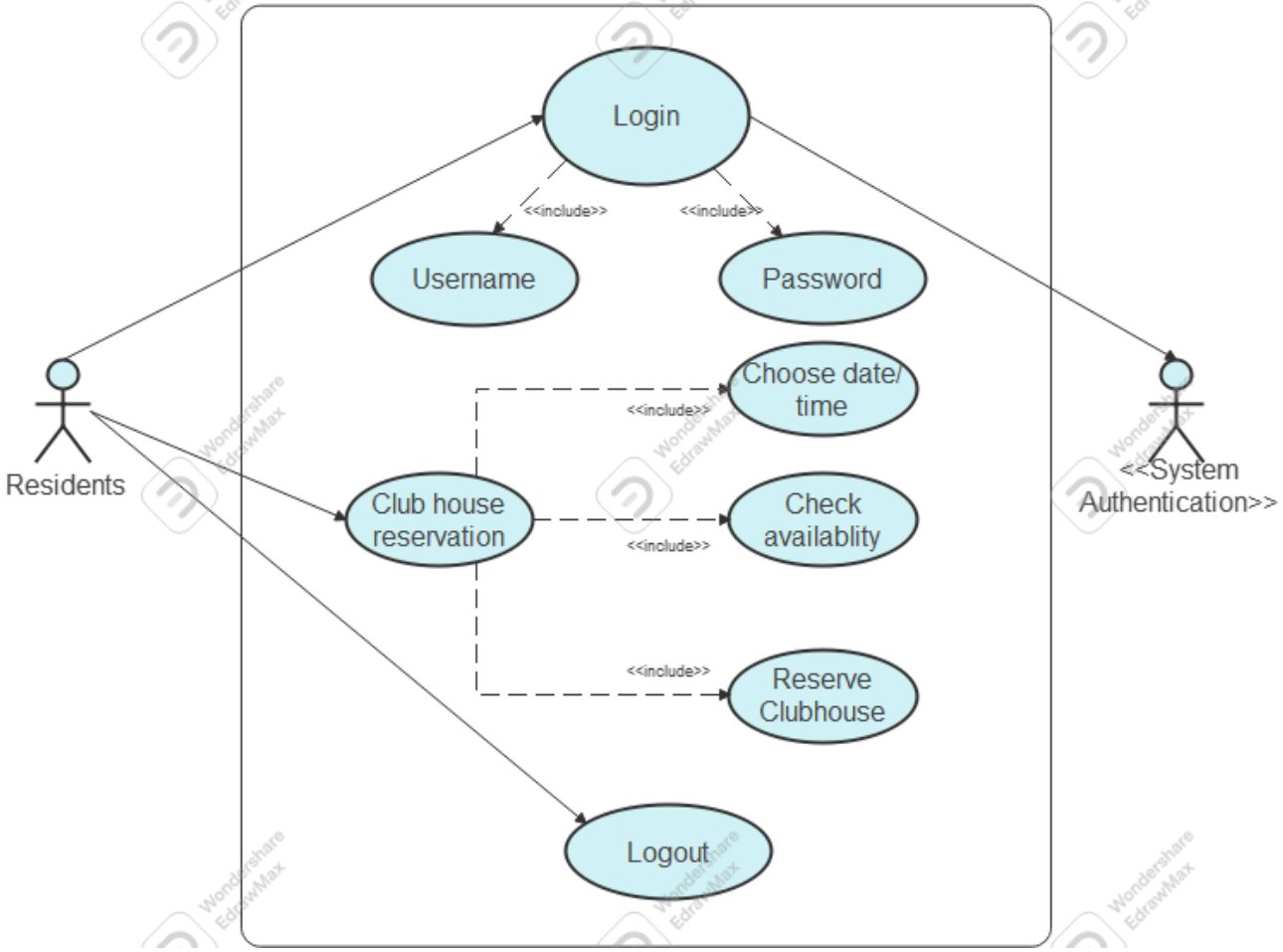
❖ Forum module



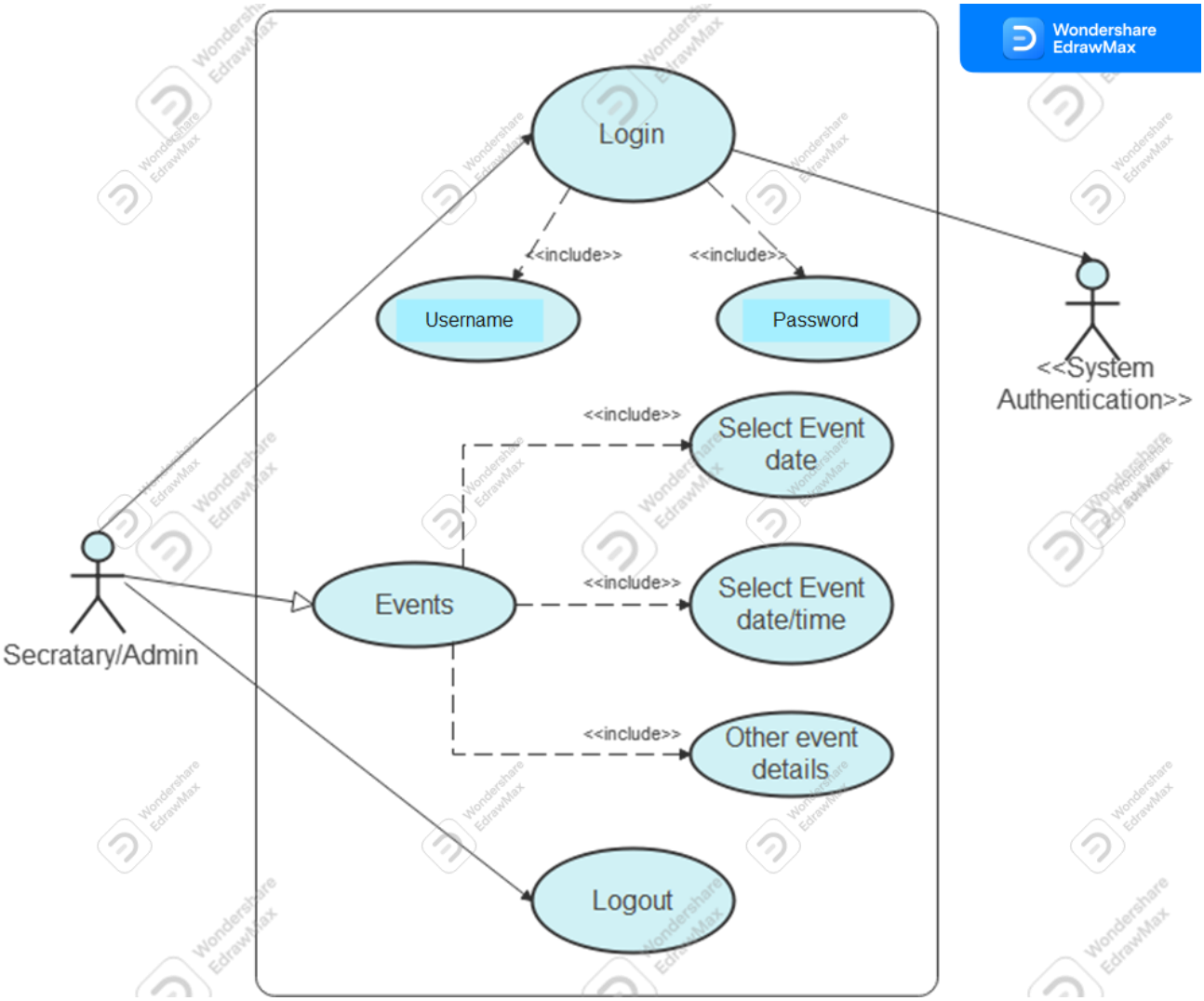
❖ Payments/Maintenance Module



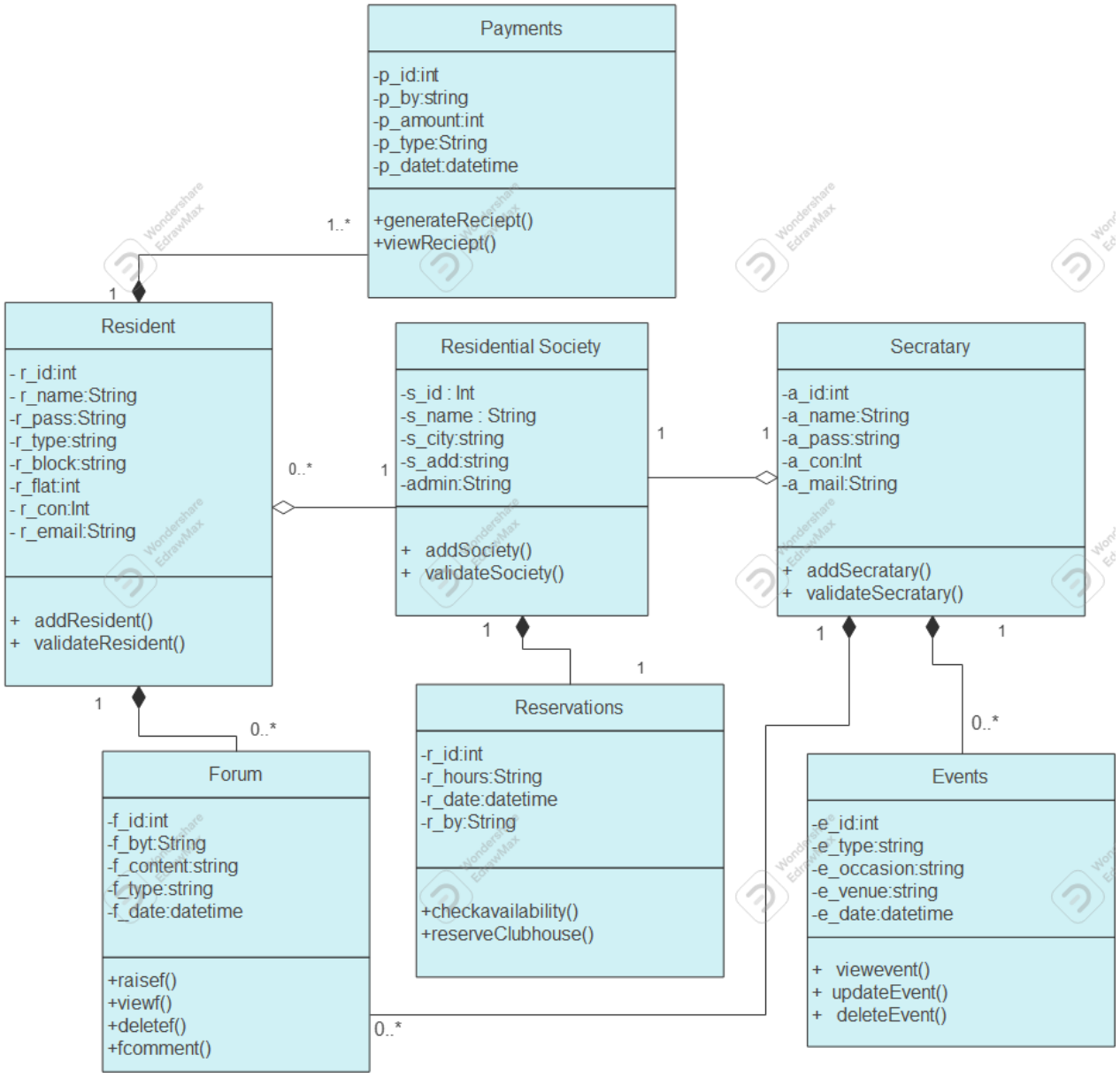
❖ Clubhouse Reservation



❖ Events Module

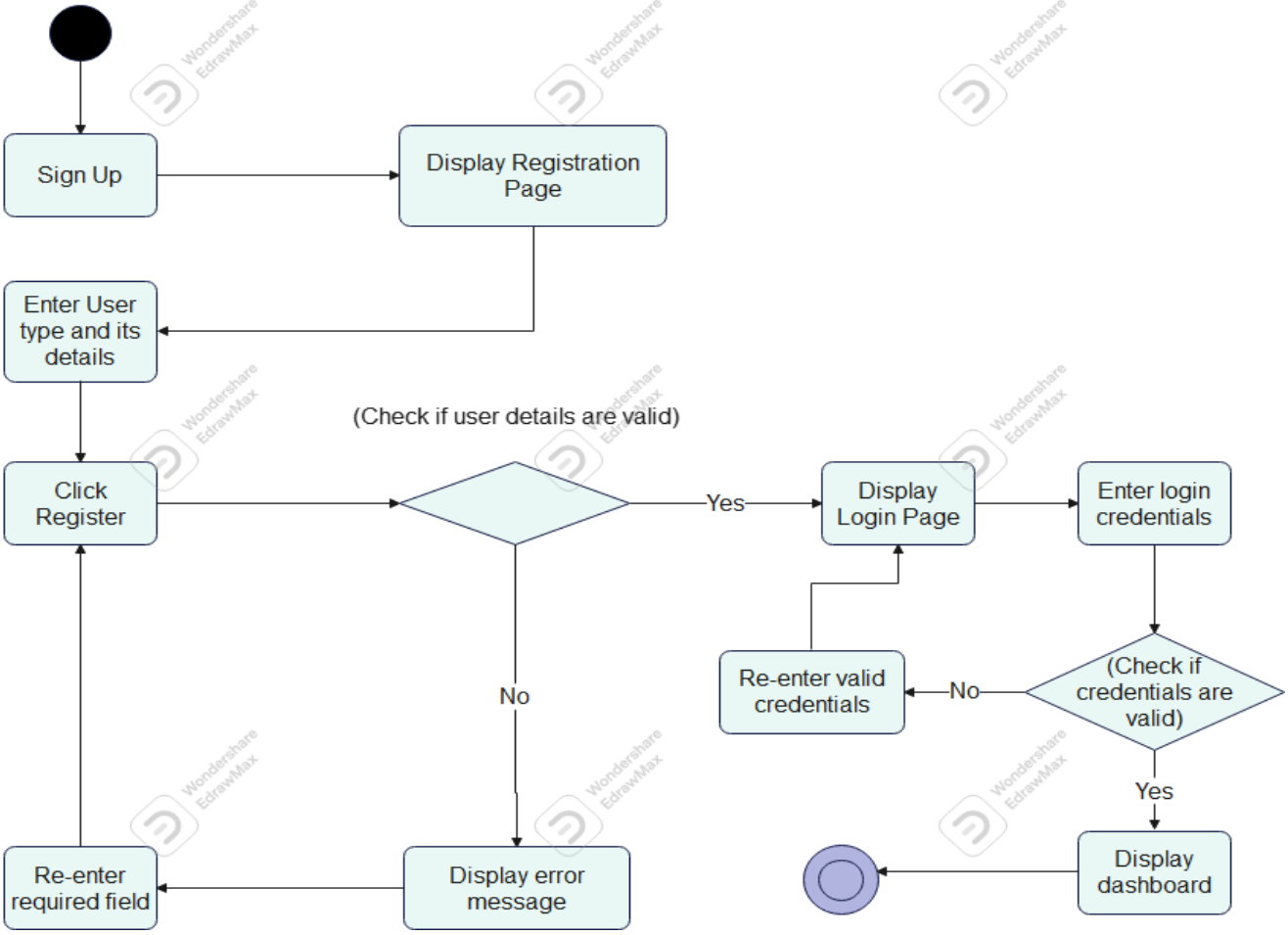


3.4 Class diagram

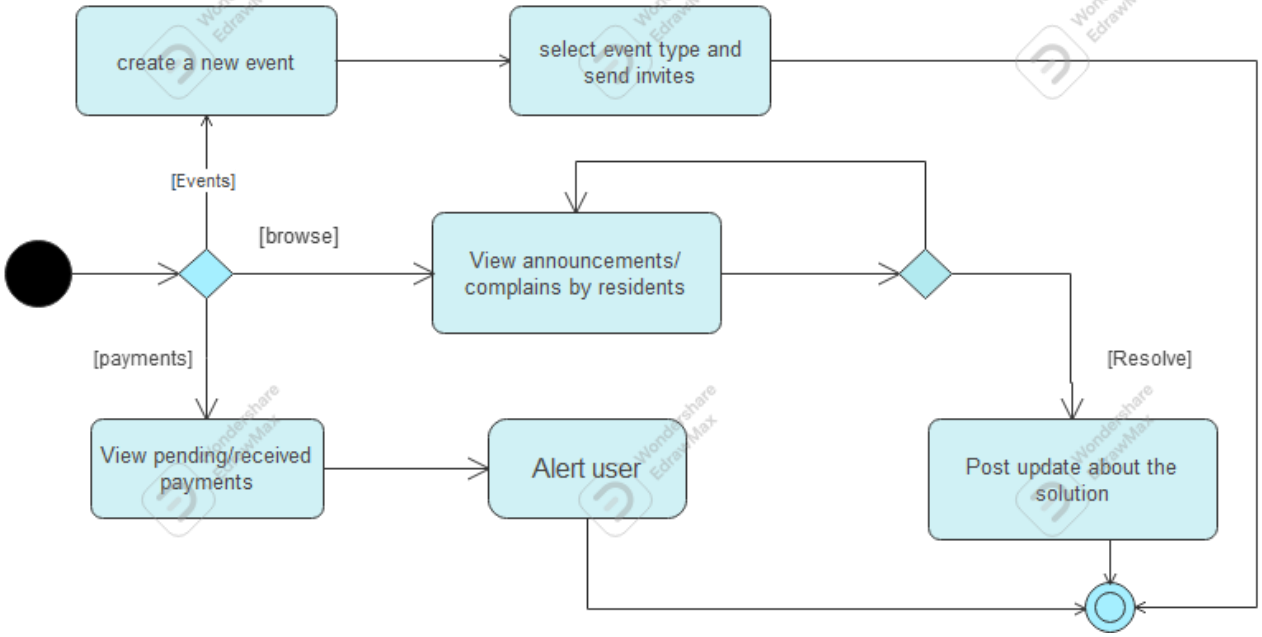


3.5 Activity Diagram

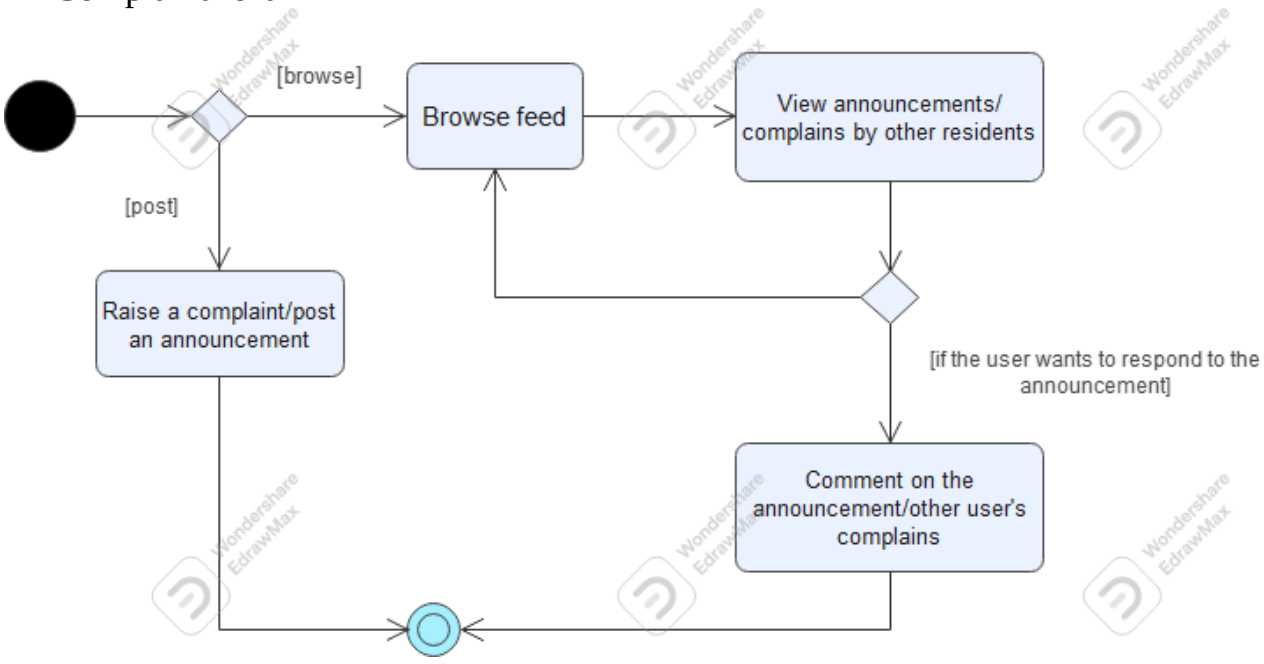
- Registration /Login module



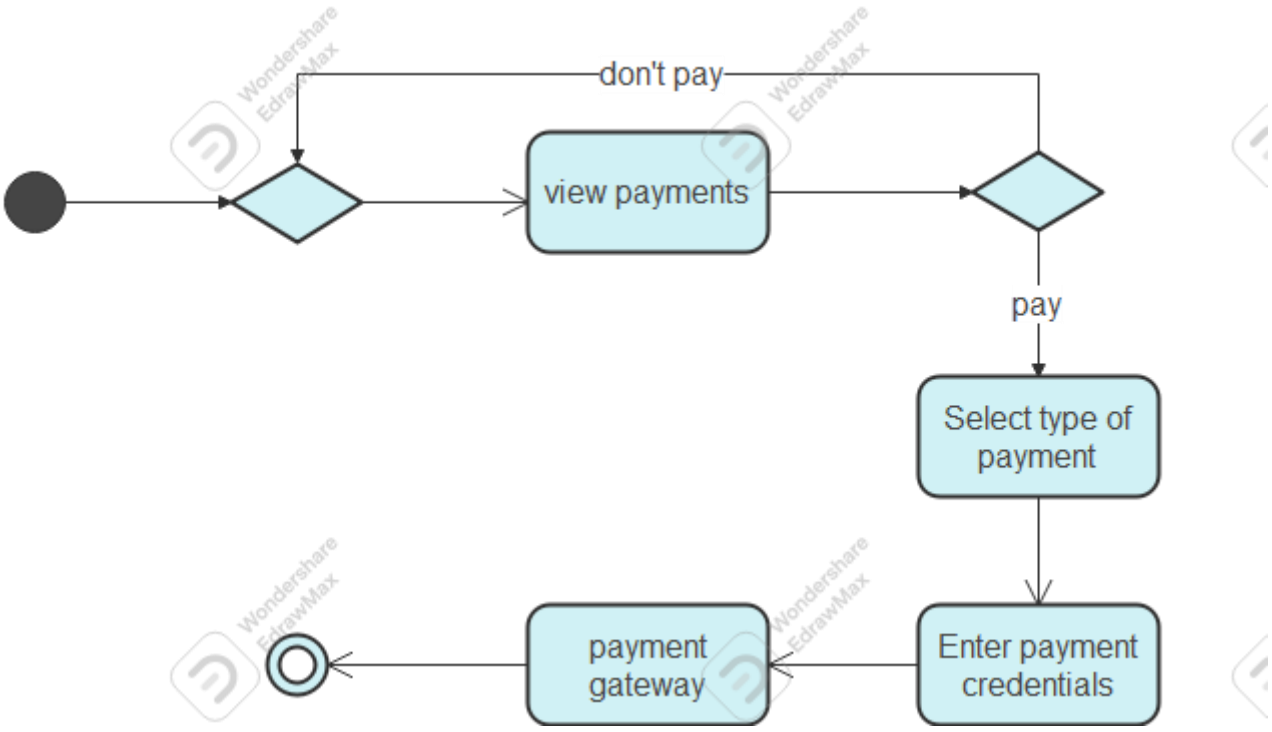
❖ Secretary/Admin module



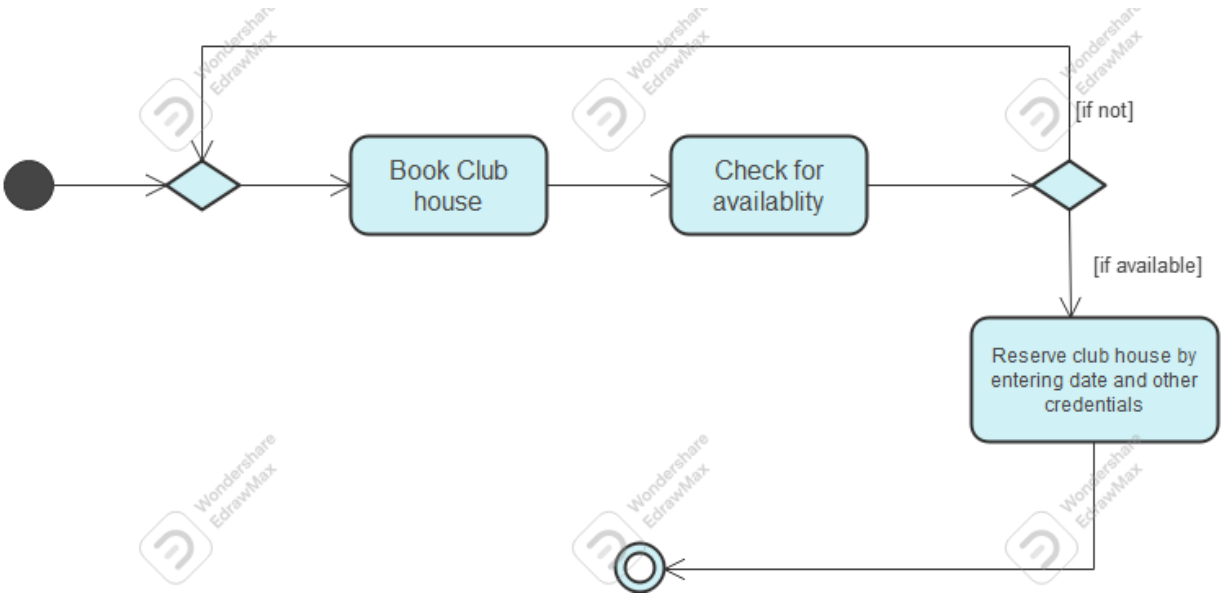
❖ Complaint forum



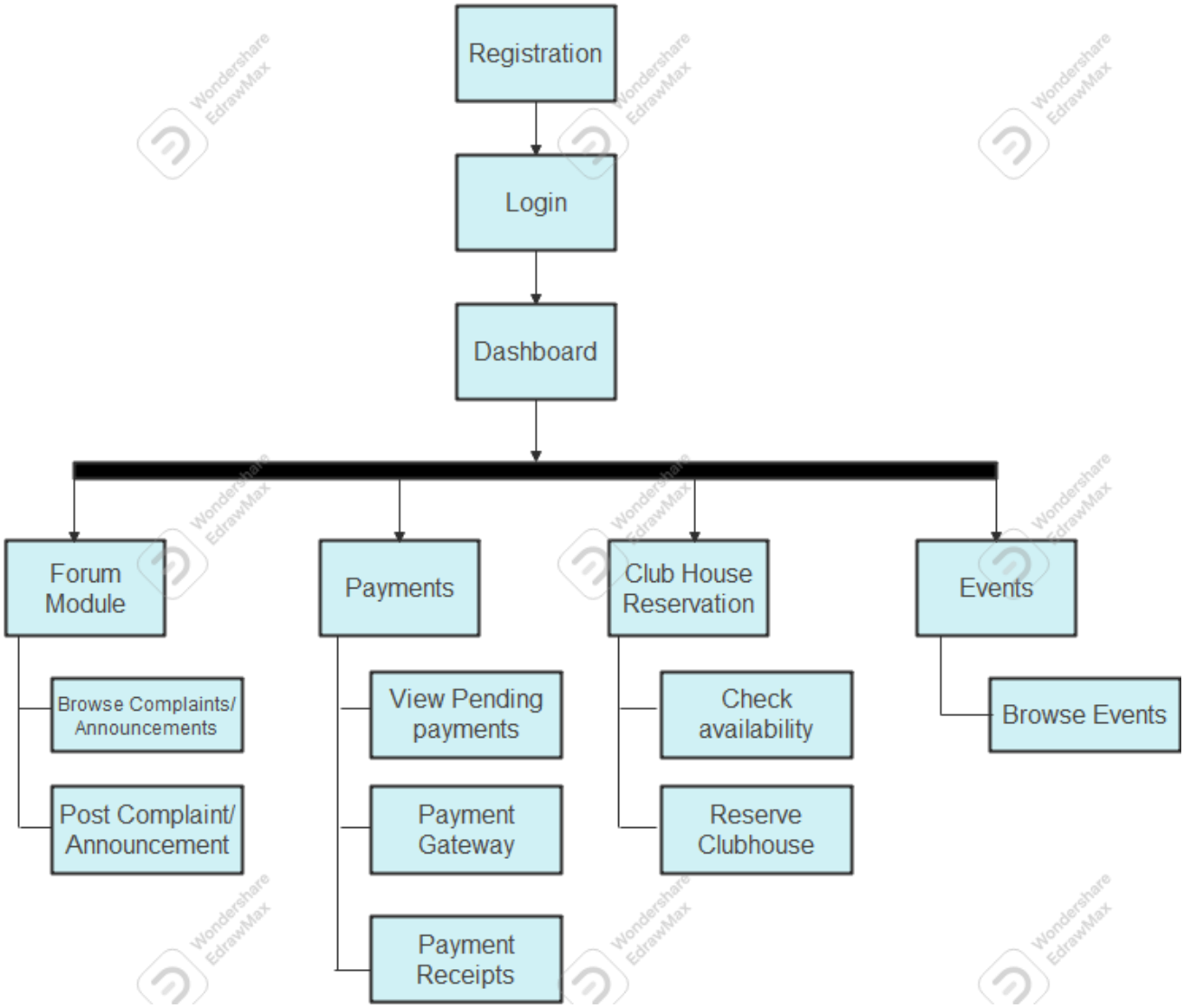
❖ Maintenance/Fine payment module



- Reserve clubhouse/ events module



3.6 Module Hierarchy Diagram



CHAPTER 4: CODING

4.1 Code Snippets

MainActivity.java

```
package com.example.friendlyneighborhood;

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

import android.os.Bundle;

import com.example.friendlyneighborhood.Model.UserModel;
import com.example.friendlyneighborhood.databinding.ActivityMainBinding;
import com.example.friendlyneighborhood.fragments.AdminEventsFragment;
import com.example.friendlyneighborhood.fragments.AdminPaymentsFragment;
import com.example.friendlyneighborhood.fragments.NotificationsFragment;
import com.example.friendlyneighborhood.fragments.EventsFragment;
import com.example.friendlyneighborhood.fragments.ForumFragment;
import com.example.friendlyneighborhood.fragments.PaymentsFragment;
import com.example.friendlyneighborhood.fragments.ReservationFragment;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.iammert.library.readablebottombar.ReadableBottomBar;

public class MainActivity extends AppCompatActivity {

    ActivityMainBinding binding;
    String userType;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        FragmentTransaction transaction =
getSupportFragmentManager().beginTransaction();
        transaction.replace(R.id.container, new ForumFragment());
        transaction.commit();

        FirebaseDatabase.getInstance().getReference().child("Users")

.child(FirebaseAuth.getInstance().getUid()).addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        UserModel user = snapshot.getValue(UserModel.class);
                        userType = user.getType();
                    }
```

```
  @Override
 public void onCancelled(@NonNull DatabaseError error) {
}});
binding.readableBottomBar.setOnItemSelectListener(new
ReadableBottomBar.ItemSelectListener() {
            @Override
            public void onItemSelected(int i) {
FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
switch (i)
  {
case 0:
transaction.replace(R.id.container, new ForumFragment());
break;
case 1:
if(userType.equalsIgnoreCase("secretary"))
 {
 transaction.replace(R.id.container, new AdminPaymentsFragment());
 break;
 }
{
  transaction.replace(R.id.container, new PaymentsFragment());
break;
 }
 case 2:
 transaction.replace(R.id.container, new NotificationsFragment());
break;
 case 3:
if(userType.equalsIgnoreCase("secretary"))
{
transaction.replace(R.id.container, new AdminEventsFragment());
 break;
 }
 else {
transaction.replace(R.id.container, new EventsFragment());
 break;
 }
case 4:
 transaction.replace(R.id.container, new ReservationFragment());
  break;
}
transaction.commit();
}});
}}
```

AndroidManifest.XML

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

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/main_icon"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MaterialComponents.DayNight.NoActionBar"
        tools:targetApi="31">
        <activity
            android:name=".CommentActivity"
            android:exported="false">
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
        <activity
            android:name=".Signup"
            android:exported="false">
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
        <activity
            android:name=".MainActivity"
            android:exported="false">
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
        <activity
            android:name=".Login"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

            <meta-data
                android:name="com.google.android.gms.wallet.api.enabled"
                android:value="true" />
        </activity>
    </application>
</manifest>
```

Acitivity_main.XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Signup">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:fontFamily="@font/poppinsbold"
        android:gravity="center"
        android:text="@string/signup"
        android:textColor="@color/purple_200"
        android:textSize="30sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/fullname"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="30dp"
        android:drawableEnd="@drawable/ic_baseline_assignment_ind_24"
        android:fontFamily="@font/poppinsmedium"
        android:hint="@string/fullname"
        android:inputType="textPersonName"
        android:maxLines="1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView4" />

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/contact"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="8dp"
        android:layout_marginRight="30dp"
        android:drawableEnd="@drawable/ic_baseline_phone_24"
        android:fontFamily="@font/poppinsmedium"
        android:hint="@string/contact"
        android:inputType="number"
        android:maxLength="10"
        android:maxLines="1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/Reenterpas"
        tools:ignore="MissingConstraints" />

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/SignUpEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="30dp"
        android:hint="@string/email"
        android:inputType="textEmailAddress"
        android:maxLines="1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/contact">

    </com.google.android.material.textfield.TextInputEditText>

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/Reenterpas"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp"
        android:drawableEnd="@drawable/ic_baseline_lock_reset_24"
        android:fontFamily="@font/poppinsmedium"
        android:hint="@string/setpassword"
        android:inputType="textPassword"
        android:maxLength="12"
        android:maxLines="1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/fullname"
        tools:ignore="MissingConstraints" />


    <TextView
        android:id="@+id/RedirectToLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:fontFamily="@font/poppinsmedium"
        android:gravity="center"
        android:text="@string/already_have_an_account_login"
        android:textColor="@color/purple_200"
        android:textSize="15sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/SignUpBtn1">

    </TextView>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayout2"
```

```
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDropd
ownMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="30dp"
        android:hint="@string/select_type"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/SignUpEmail">

        <AutoCompleteTextView
            android:id="@+id/userType"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:fontFamily="@font/poppinsmedium"
            android:inputType="none" />

    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayoutSociety"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDropd
ownMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="30dp"
        android:hint="@string/select_society"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textInputLayout2">

        <AutoCompleteTextView
            android:id="@+id/SelectSocietyName"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:fontFamily="@font/poppinsmedium"
            android:inputType="none" />

    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayoutFlat"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDropd
ownMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
```

```
        android:layout_marginTop="8dp"
        android:layout_marginEnd="30dp"
        android:hint="@string/select_your_block_and_flat_no"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textInputLayoutSociety">

        <AutoCompleteTextView
            android:id="@+id/SelectBlockFlat"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:enabled="false"
            android:fontFamily="@font/poppinsmedium"
            android:inputType="none" />

    </com.google.android.material.textfield.TextInputLayout>


    <Button
        android:id="@+id/SignUpBtn1"
        style="@style/TextAppearance.AppCompat.Widget.Button.Colored"
        android:layout_width="200dp"
        android:layout_height="60dp"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="16dp"
        android:layout_marginRight="30dp"
        android:text="@string/signup"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textInputLayoutFlat"
        tools:ignore="MissingConstraints" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

CHAPTER 5: TESTING

5.1 Test Strategy

The major aim while developing this project was to deliver a bug free application that is functional in every aspect and serves the purpose for which it was created in first place. To deliver a bug free software, this project has been through various testing phases. Testing is the process of execution of the program with explicit intention of searching for errors. A successful test is one that reveals errors in the system. Test cases are designed with intent of determining whether the system will process it in the correct & pre-planned order. Testing is a step, which is actual usage of the system to remove the various discrepancies present.

The strategy chosen by me is the traditional one that follows all the rules and regulations that comes with SDLC i.e. Software Development Life Cycle. It goes through all the steps in testing phase a software does in SDLC. This project has been through every level of testing. Testing levels are the procedure for finding the missing areas and avoiding overlapping and repetition between the development life cycle stages. In order to test any application, we need to go through all the phases of SDLC. Like SDLC, we have multiple levels of testing, which help us maintain the quality of the software.

5.2 Unit Test Plan:

Unit testing is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not. The first level of testing involves analyzing each unit or an individual component of the software application. Unit testing is also the first level of functional testing. The primary purpose of executing unit testing is to validate unit components with their performance. A unit component is an individual function or regulation of the application, or we can say that it is the smallest testable part of the software. The reason of performing the unit testing is to test the correctness of inaccessible code.

Every module of this project has been through unit testing. All of them has been tested individually through every scenario. The login and registration module has been tested by entering valid as well as invalid credentials and the bugs has been fixed. The forum module, payment module, event and registration etc. has been tested through every perspective and have returned positive results.

5.3 Acceptance Testing

The final level of software testing is acceptance testing, which is used to evaluate whether a specification or the requirements are met as per its delivery. The software has passed through three testing levels. Some minor errors can still be identified when the end-user uses the system in the actual scenario. In simple words, we can say that Acceptance testing is the squeezing of all the testing processes that are previously done.

5.4 Test Cases

Description	Test Steps	Expected Result	Actual Result
Login Success	1. Fetch username and password from forminputs. 2. Fetch username and password from logintable. 3. Compare username and password. 4. Login Successful Message box.	Redirect to Home Page	Redirected to Home Page
Login Fail	1. Fetch username and password from forminputs. 2. Fetch username and password from logintable. 3. Compare username and password. 4. Login Failed Message box.	Login Page	LoginPage
Post Complain	1. Fetch text/data input given by the user. 2. Save the text in the database. 3. Post appear on the user, admin and other residents forum fragment.	Post appears of the Forum fragment.	Post appears of the Forum fragment.

Post complain Failure	1. Fetch text/data input given by the user. 2. Save the text in the database. 3. Post doesn't appear on the user, admin and other residents forum fragment.	Post doesn't appear of the user feed and update failure toast appears.	Post doesn't appear of the user feed and update failure toast appears.
Payments Successful	1. The user enters payment type and amount and is directed to the payment gateway. 2. User inputs payment details and initializes payment. 3. The payment is done successfully and the data is saved in database and a notification is received by admin.	Payment successful notification is received and toast appears.	Payment successful notification is received and toast appears.
Payment failure	a. The user enters payment type and amount and is directed to the payment gateway. b. User inputs payment details and initializes payment. c. The payment fails and user data isn't saved in database.	Payment failure toast appears.	Payment failure toast appears.

Clubhouse reservation Successful	1. User selects desired date and enter the reason he/she wants to reserve the clubhouse for. 2. The reservation is successful and toast appears and a notification is received for the same. 3. The data is saved in the database.	Reserved successfully toast appears and notification is received.	Reserved successfully toast appears and notification is received.
Clubhouse reservation Failure	4. User selects desired date and enter the reason he/she wants to reserve the clubhouse for. 5. The reservation fails and toast appears and no notification is received. 6. The data isn't saved in the database.	Reservation failure toast appears.	Reservation failure toast appears.
Events posted by admin and received by residents	1. The admin posts the events with details like date venue occasion etc. 2. The event is posted and is received by user in the events fragment. 3. The data is saved in the database.	The event is received by user in the events fragment.	The event is received by user in the events fragment.
Events Post failure	1. The admin posts the events with details like date venue occasion etc. 2. The event post fails and nothing appears in the event fragment. 3. The data isn't saved in the database.	Unable to post event toast appears and nothing appear in events fragment.	Unable to post event toast appears and nothing appear in events fragment.

Admin Activities Successful.	1. The admin logs in with credentials. 2. He/she can view complains, receive payment notifications, reservation notifications, and is able to post event. 3. The data is successfully fetched/saved in the database (based on the activity performed.)	The notifications are received properly and events are being posted successfully.	The notifications are received properly and events are being posted successfully.
Admin Activities failure.	1. The admin logs in with credentials. 2. He/she can't view complains, can't receive payment notifications and reservation notifications, and isn't able to post event. 3. The data isn't fetched/saved in the database (based on the activity performed.)	The notifications aren't received by admin and no events are posted.	The notifications aren't received by admin and no events are posted.

CHAPTER 6: Limitation of proposed system

Following are some limitations of this project:

- The system will not provide facility to make payments by means of credit card/ Net banking.User has to initiate payments via UPI apps like (Gpay, Paytm and PhonePe etc.)
- This project requires an internet connection to stay connected.
- It is only limited to Residents, tenants and secretary. Other important stakeholders like cleaners, watchmen etc. have no access/purpose in this app yet.
- This app only works on the devices with Android OS 5.0 and above.

CHAPTER 7: Proposed Enhancement

This application can further be enhanced in multiple ways. Some are listed as below:

- Modules for stakeholders like watchmen, cleaner and utility workers like plumbers, electricians, gardeners etc. can be introduced. It may include recording their attendance as well.
- Module for contacts with emergency services like house cleaning, pest control etc. can be added.
- A module where watchmen can inform the residents about the visitors/ delivery agents etc. can be added and entry can only be granted once approved by resident; thus enhancing the security measures.

CHAPTER 8: Conclusion

This application is made with the purpose of serving society in a better way. It will reduce paper work, save time and efforts for users. This app will introduce a new way to manage residential societies for their respective managing committees. The users would have an integrated platform to communicate with the managing committee and other fellow residents. The users will always stay informed about all the events and activities happening in the society and be updated and informed. They'd be able to make hassle-free payments in a secured manner. The paper-work will be reduced and the user data will be made more secure.

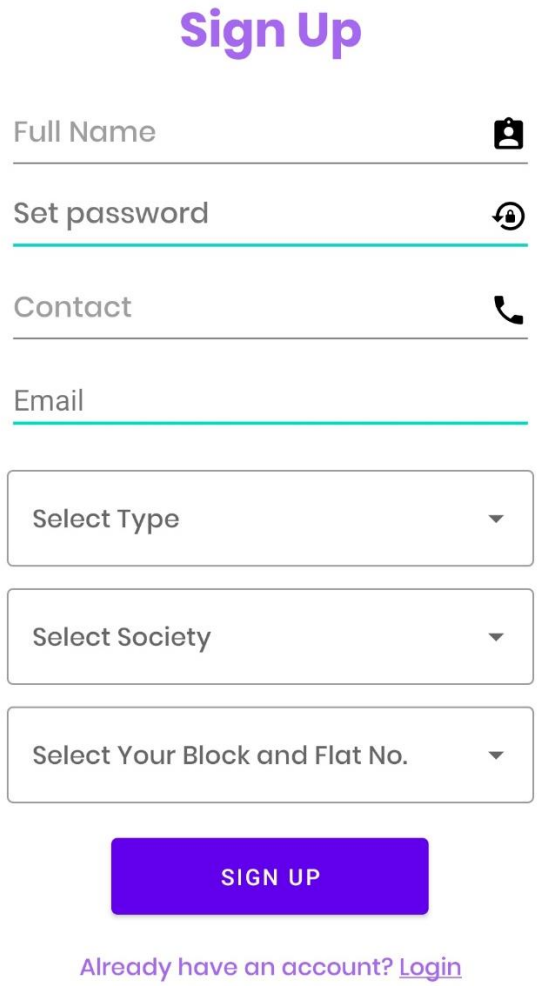
CHAPTER 9: Bibliography

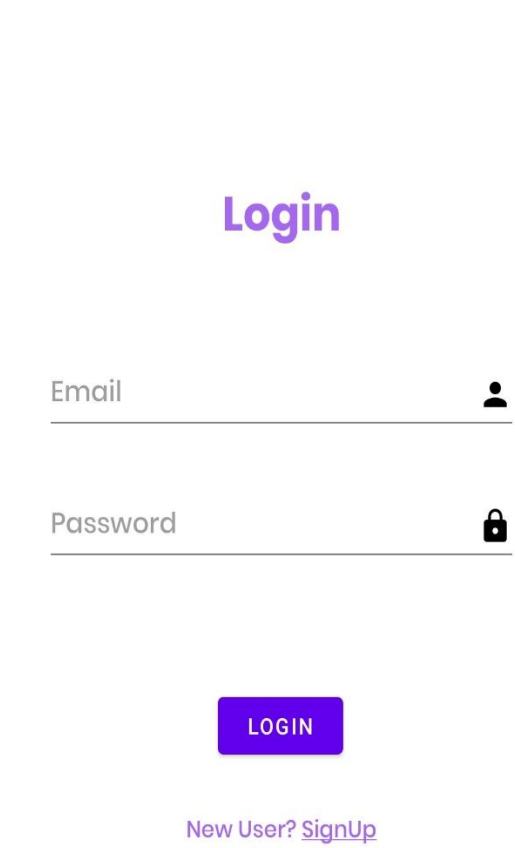
Following are the references that helped in the development of this project.

- https://developer.android.com/
- https://www.geeksforgeeks.org/introduction-to-android-development/
- https://www.youtube.com/watch?v=P1tf_ByFATo&list=PLs1bCj3TvmWmM-qN3FsCuPTTX-29I8Gh7
- https://www.youtube.com/watch?v=TqE8Ji_R_Qw&t=877s
- https://firebase.google.com/

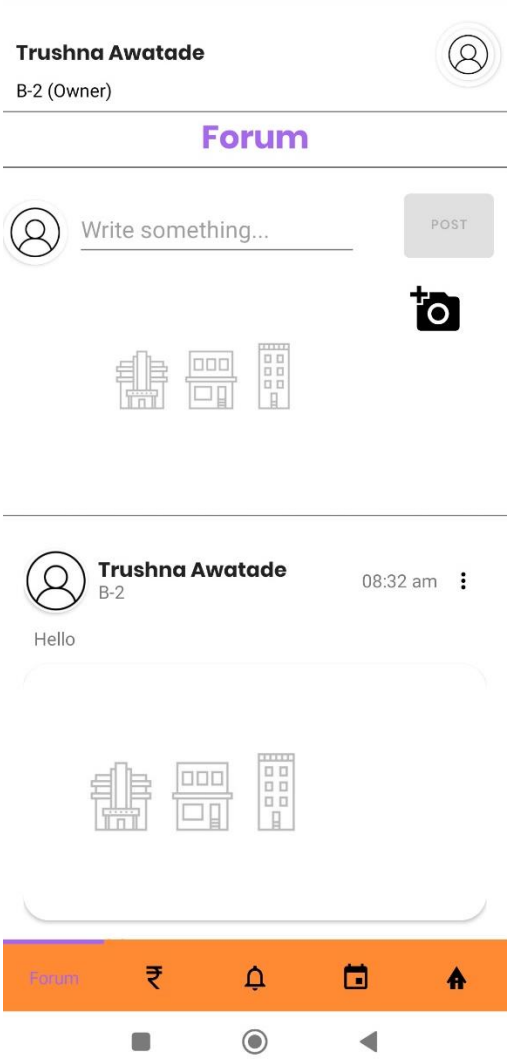
CHAPTER 10: User Manual

1. Login/Registration

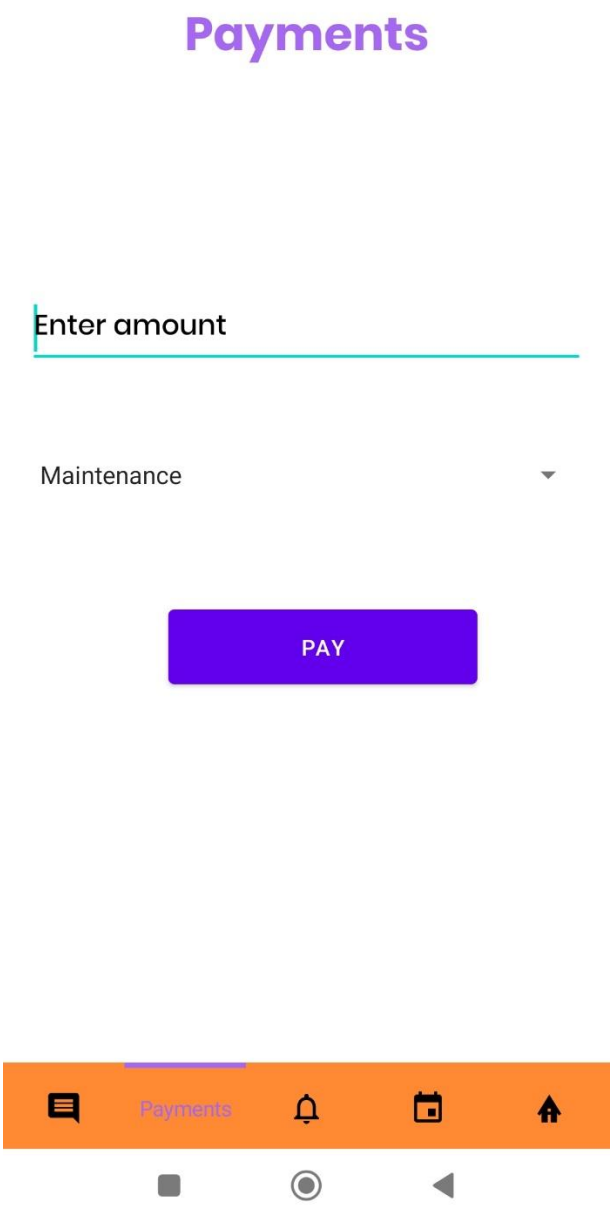




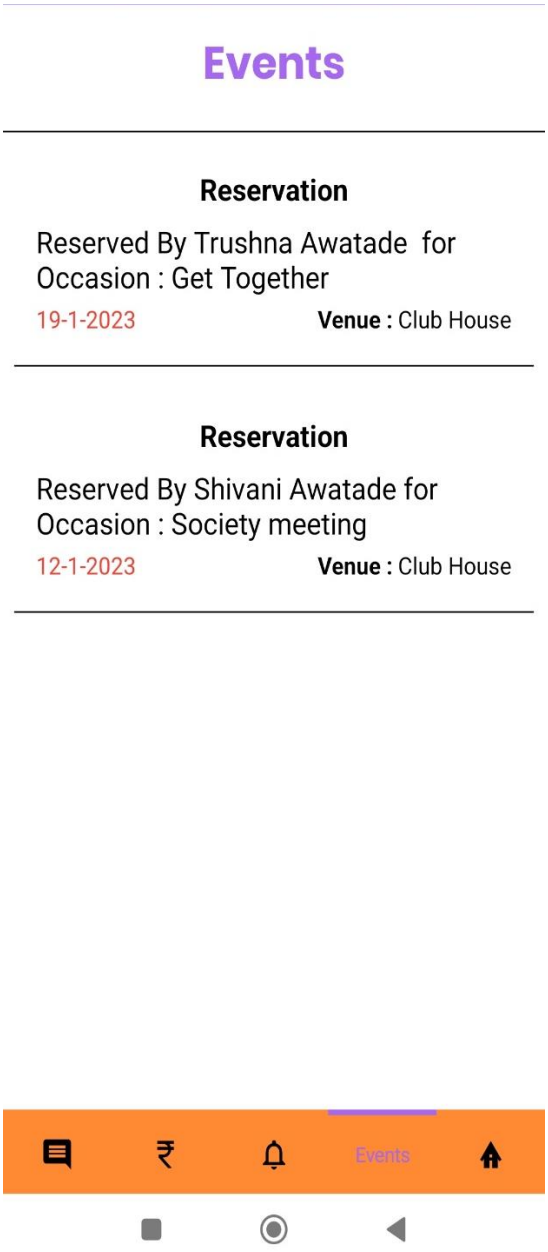
2. Forum Module

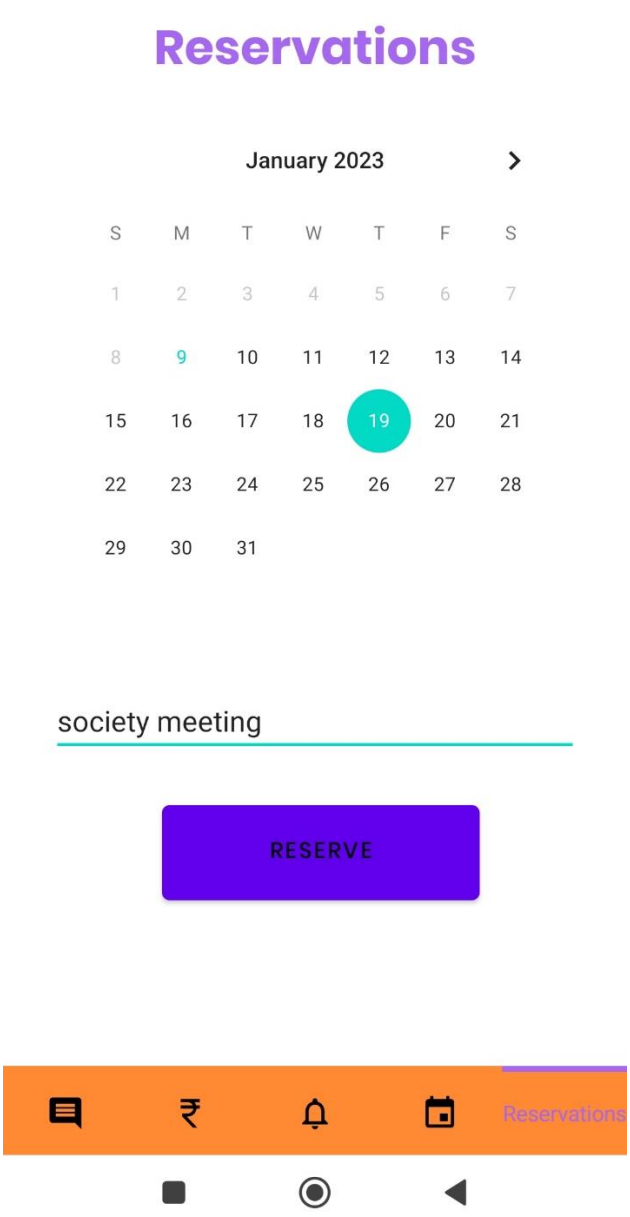


3. Payments Module



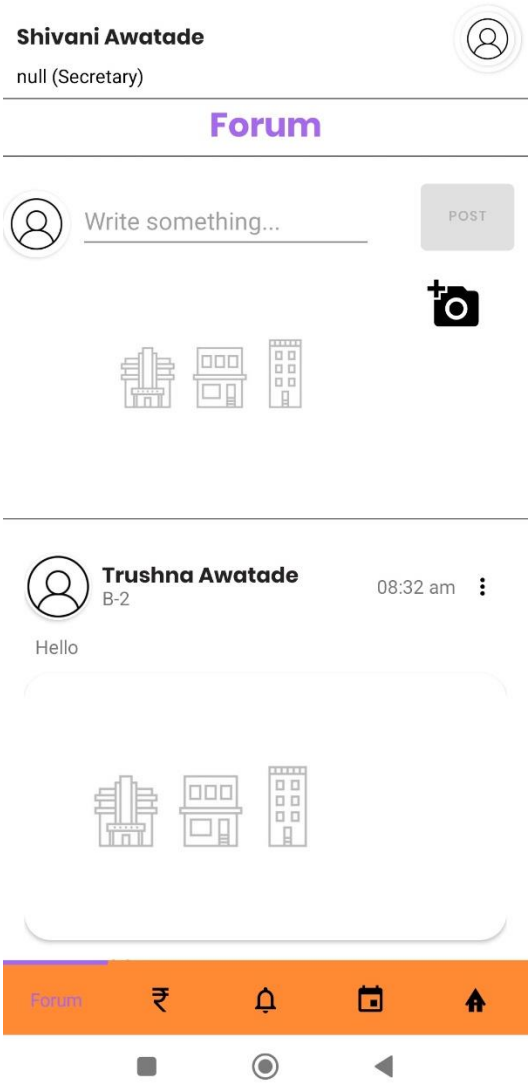
4. Reservation/Events Module



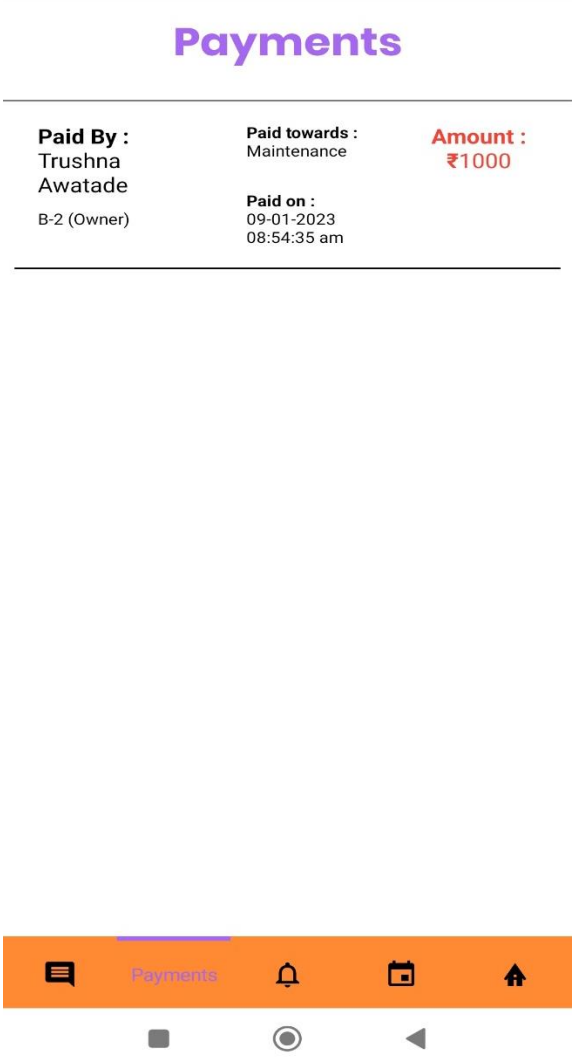


5. Admin Module

Admin Module (Forum)



Admin Module (payments)



Admin Module (events)