VEER NARMAD SOUTH GUJARAT UNIVERSITY
University Campus, Udhna-Magdalla Road, Surat – 395001, Gujarat, India.

Department of Information and Communication Technology
M.Sc. (ICT) Programme

AS PARTIAL REQUIREMENT FOR THE DEGREE

OF

MASTER OF SCIENCE IN INFORMATION AND COMMUNICATION TECHNOLOGY (MSC ICT 2 Year Course)

2024-25

(SEMESTER – III)

PROJECT TITLE

"Home Owner Association Connect"

GUIDED BY

Ms. Shreya Kapadiya

SUBMITTED BY

Jadhav Swati N.[R24110018000710057]

Prajapati Priya V.[R24110018000710038]

VEER NARMAD SOUTH GUJARAT UNIVERSITY

Department of Information and Communication Technology

M.Sc.(Information and Communication Technology) Programme

Certificate

This is to certify that Mr**. Prajapati Priya Vashrambhai** with exam seat number **1033** and PG Enroll Number **2024041600** has worked on his/her project work entitled as **HOME OWNER CONNECT PORTAL** at **Department of ICT** as a partial fulfillment of requirement of M.Sc. (Information and Communication Technology) - 3rd Semester, during the academic year 2024-2025.

Date:

Place: Department of ICT, VNSGU, Surat

Internal Project Guide
MSc(ICT) 3rd Semester
Department of ICT
VNSGU, Surat

Head of the Department
Department of ICT
VNSGU, Surat

VEER NARMAD SOUTH GUJARAT UNIVERSITY

Department of Information and Communication Technology

M.Sc.(Information and Communication Technology) Programme

Certificate

This is to certify that Mr**. Jadhav Swati Narendra** with exam seat number **1050** and PG Enroll Number **2024041619** has worked on his/her project work entitled as **HOME OWNER CONNECT PORTAL** at **Department of ICT** as a partial fulfillment of requirement of M.Sc. (Information and Communication Technology) - 3rd Semester, during the academic year 2024-2025.

Date:

Place: Department of ICT, VNSGU, Surat

Internal Project Guide
MSc(ICT) 3rd Semester
Department of ICT
VNSGU, Surat.

Head of the Department
Department of ICT
VNSGU, Surat

ACKNOWLEDGEMENT

When we reach the completion of the project, giving credit becomes a must, as without the support of other so many people's help and guidance, this project couldn't be completedsuccessfully.

Success in such a comprehensive project cannot be achieved single-handled. It is team effort that sails the ship to the coast. So, we would like to express our sincere thanks to all the dignitaries who were involved in making this project a great joy and turning it into successful piece of work.

We would like to take opportunity to thank our **Department J.P. Dawer Institute of Information Science & Technology, Surat** for giving us this tremendous opportunity to work in the real-time project.

Shreya Kapadia, our professor and project coordinator, has been very prudent to us throughout our college studies. She is the person who has been giving direction to our work and the shape to our imagination and always ready to give best guidance, give solutions whenever required. We express our regards to her from the core of our heart.

Last but not the least, our heartfelt appreciation goes to all those not named here, but who have rendered their co-operation, little or more, directly or indirectly involved in the development of this system.

Thanking All.

Prajapati Priya V.

Jadhav Swati N.

INDEX

Sr. No	Topics	Page No
1	Introduction 1.1 Customer Profile 1.1.1 Current System – about the existing system 1.1.2 Customer Details	1 - 3
2	Proposed System 2.1 Scope 2.2 Objective 2.3 Constraints 2.3.1 H/W Constraints 2.3.2 S/W Constraints 2.4 Advantages 2.5 Limitations	4 - 7
3	Environment Specification 3.1 Hardware & Software Requirements 3.2 Development Description	8 - 11
4	System Planning 4.1 Feasibility Study 4.2 Software Engineering Model 4.3 Risk Analysis 4.4 Project Schedule 4.4.1 Task Dependency 4.4.2 Project Table	12 - 15
5	System Analysis 5.1 Detailed SRS 5.2 UML Diagram 5.2.1 Use Case Diagram 5.2.2 Class Diagram 5.2.3 Activity Diagram 5.2.4 Sequence Diagram 5.3 ER Diagram	16 - 27
6	Software Design 6.1 Database Design 6.2 Interface Design sitemap followed with page snapshots 6.3 Architecture Design	28 - 47
7	Testing 7.1 Unit Testing 7.2 Integration Testing	48
8	Future Enhancement	49
9	Glossary	50
10	Reference	51

1. INTRODUCTION

Home Owner Associations (HOAs) play a vital role in managing residential communities by maintaining amenities, collecting payments, resolving complaints, and ensuring smooth communication between residents and management. As residential societies grow in size and complexity, manual and semi-digital management systems become inefficient, time-consuming, and difficult to monitor.

HOA Connect is a web-based management portal developed using the **MERN Stack (MongoDB, Express.js, React.js, Node.js)** that provides a centralized and secure platform for managing residential communities. The system introduces **three clearly defined user roles — Super Admin, HOA Admin, and Resident** — each with controlled access and responsibilities.

The application ensures **transparency, operational efficiency, and self-governance** by enabling online payments, amenity booking, complaint tracking, announcements, meetings, polls, and analytics. By digitizing core HOA operations, HOA Connect reduces manual effort, improves communication, and enhances resident participation.

1.1 Customer Profile

The HOA Connect Portal is designed for **residential communities and housing societies** that require an organized, scalable, and role-based management system. The system caters to all stakeholders involved in community governance and daily operations.

The primary customers include:

- Home Owner Associations managing one or more residential communities
- HOA Board Members responsible for administration and decision-making
- Residents seeking convenient access to community services

1.1.1 Current System

Most residential communities currently rely on **traditional manual or semi-digital systems** for managing their daily operations. The existing system typically involves:

- Community records maintained using registers, spreadsheets, or isolated software tools
- HOA Admins manually assigning roles without centralized access control
- Amenities managed informally or through notice boards
- Complaints submitted verbally or via messaging applications
- Payments collected through cash, bank transfers, or external payment apps
- Announcements shared through printed notices or informal WhatsApp groups
- Meetings conducted offline with no structured RSVP or tracking mechanism

Limitations of the Existing System:

- No centralized platform for community management
- Lack of role-based access and security
- Difficulty in tracking payments and complaints
- Poor transparency in administrative decisions
- No global reporting or analytics
- High dependency on manual coordination
- Difficult to scale for multiple communities

1.1.2 Customer Details

1. Super Admin (Platform Owner / System Administrator)

- Manages multiple residential communities
- Creates communities and assigns HOA Admins
- Defines amenities available for each community
- Monitors global payment analytics and system-wide activity
- Ensures consistent governance across communities

2. HOA Admin (Community Board Member)

- Responsible for managing a specific residential community
- Manages residents, complaints, announcements, documents, payments, and polls
- Oversees amenity usage and availability
- Organizes meetings and monitors participation
- Has the authority to replace themselves with another resident if required

3. Residents

- Homeowners or tenants residing in the community
- Interact with the system to:
    - Book amenities
    - Raise and track complaints
    - Make online payments
    - View announcements and documents
    - Participate in polls and meetings
- Manage their own profile and receive community notifications

2. PROPOSED SYSTEM

2.1 Scope

- The scope of the HOA Connect Project covers various aspects of residential community (Home Owner Association) management through a centralized web-based platform.

- This includes community management, resident record maintenance, complaint handling, amenities management, online payments, announcements, meetings, and polls.

- The system facilitates transparent communication between HOA Admins and residents by providing real-time notifications, document sharing, and community updates.

- Overall, the system aims to improve operational efficiency, reduce manual work, and ensure smooth day-to-day functioning of residential societies.

2.2 Objective

- To understand and implement the complete workflow of Home Owner Association management, including resident onboarding, complaint resolution, and payment handling.

- To provide a centralized platform that enhances communication between Super Admins, HOA Admins, and Residents.

- To explore the technical implementation of a full-stack web application using the MERN stack for efficient data management and role-based access control.

- To improve resident satisfaction by providing easy access to amenities booking, notifications, and community information.

2.3 Constraints

2.2.1 Hardware Constraints:

1. **Server Limitations :**

- Limited server processing power and memory may affect handling of concurrent users, real-time notifications, and analytics processing.
- Storage limitations could restrict document uploads such as meeting minutes, circulars, and community records.
- Dependence on hosting infrastructure introduces a risk of downtime affecting access to the system.

2. **Network Limitations :**

- Limited internet bandwidth may affect real-time features such as notifications, payments, and dashboard analytics.
- Unstable connectivity could delay data synchronization between frontend and backend services.

3. **Scalability Issues :**

- Hardware limitations may restrict support for a growing number of communities and residents.
- Expanding infrastructure for large-scale deployment may require additional investment.

2.2.1 Hardware Constraints:

1. **Platform Dependency :**

- The system depends on web browser compatibility; older browsers may not fully support all features.
- Mobile responsiveness may vary across devices if not properly optimized.

2. **Performance Constraints :**

- High traffic during peak times (payments, announcements, voting) may affect performance.
- Inefficient database queries or un-optimized backend APIs may cause delays.

2.4 Advantages

1. **Efficient Community Management:**

   - Automates resident records, complaints, payments, and amenities management.
   - Reduces paperwork and manual administrative effort.

2. **Improved Transparency:**

   - Residents can track complaints, payments, and announcements in real time.
   - Polls and meetings ensure democratic decision-making.

3. **Streamlined Communication:**

   - System-wide and community-specific notifications improve engagement.
   - Centralized announcements reduce miscommunication.

4. **Data-Driven Insights:**

   - Admins can view analytics on complaints, payments, and community activity.
   - Helps management make informed decisions.

5. **Enhanced Resident Experience:**

   - Easy access to services like payments, amenities booking, and documents.
   - User-friendly dashboards improve convenience and satisfaction.

6. **Secure and Role-Based Access:**

   - JWT authentication and role-based permissions ensure data security..
   - Each user accesses only authorized information.

2.5 Limitations

1. **Initial Setup Cost :**

   - Development, deployment, and maintenance may require significant investment.
   - Training users to adapt to the system may take time.

2. **Technical Dependency :**

   - Continuous internet access is required for system usage.
   - Non-technical users may require support during initial adoption.

3. **Maintenance Requirements :**

   - Regular updates, bug fixes, and backups are essential.
   - Downtime during maintenance may temporarily affect access.

4. **Scalability Bottlenecks :**

   - Performance issues may arise if the system is not scaled properly for large communities.

5. **Data Privacy Risks :**

   - Improper security configurations could expose sensitive resident or payment data.
   - Strong security practices are required to prevent breaches.

3. ENVIRONMENT SPECIFICATION

3.1 Hardware and Software Requirements

Hardware Requirements	
Processor	Intel Core i5-4310U CPU
RAM	12.0 GB
Hard Disk	1.16 TB
Input Device	Keyboard , Mouse

Software Requirements	
Host Operating System (Server Side)	Windows 10
Operating System	Any Operating System
Browser	Google Chrome, Mozila Firefox
Database	MongoDB
Development	Visual Studio Code
Server	NodeJS

3.2 Development Description

1. Project Planning and Requirements Gathering :-

- **Objective :** To define the project scope, identify stakeholders, gather HOA-specific requirements, and establish a structured development timeline for the HOA Connect system.
- **Activities :**
  - Conduct discussions with HOA board members, residents, and administrators to understand community management needs.
  - Identify core modules such as community management, resident management, payments, complaints, amenities, meetings, and notifications.
  - Prepare a detailed project plan with milestones for module development, integration, testing, and deployment.
  - Finalize role-based access requirements for Super Admin, HOA Admin, and Residents.

2. Design Phase :-

- **Objective :** To design the system architecture, user interfaces, and database structure suitable for managing residential communities efficiently.
- **Activities :**
  - Design the database schema with collections such as Users, Communities, Amenities, Complaints, Payments, Announcements, Meetings, Polls, and Documents.
  - Prepare system architecture diagrams illustrating interaction between frontend (React), backend (Node.js/Express), and database (MongoDB).
  - Design role-based dashboards for Super Admin, HOA Admin, and Residents.
  - Define API structure and data flow between client and server.

3. Frontend Development :-

- **Objective :** To develop a responsive, user-friendly interface that allows smooth interaction for all user roles.
- **Technologies :** HTML, CSS, JavaScript, React.js, Bootstrap/Tailwind CSS
- **Activities :**
  - Implement UI Components for login, registration, dashboards, and role-based navigation.

- Develop pages for announcements, complaints, amenity booking, payments, polls, meetings, and documents.
- Implement form validations and error handling for better user experience.
- Integrate frontend with backend APIs using Axios or Fetch.

4. Backend Development :-

- **Objective :** To build secure and efficient server-side logic for handling HOA operations and enforcing business rules.
- **Technologies :** Node.js, Express.js, JWT Authentication
- **Activities :**
  - Develop RESTful APIs for users, communities, amenities, complaints, payments, polls, meetings and notifications.
  - Implement role-based authorization using JWT tokens.
  - Apply business rules such as Admin replacement, community-wise data isolation, and amenity booking limits.
  - Handle error management, data validation, and secure API access.

5. Database Development :-

- **Objective :** To store and manage HOA-related data securely, efficiently, and in a scalable manner.
- **Technologies :** MongoDB, Mongoose ODM
- **Activities :**
  - Create collections for Users, Communities, Amenities, Complaints, Payments, Announcements, Meetings, Polls, and Documents.
  - Define relationships between collections (e.g., Community → Users, Community → Amenities).
  - Implement indexing for faster data retrieval.
  - Ensure data consistency and validation using Mongoose schemas.

6. Testing Phase:-

- **Objective :** To verify that the system functions correctly, securely, and meets all functional requirements.
- **Activities :**
  - Perform unit testing on individual modules such as login, payments, complaints, and bookings.

- Conduct integration testing to ensure proper communication between frontend, backend, and database.
- Test role-based access control for Super Admin, HOA Admin, and Residents.
- Fix identified bugs and optimize performance.

7. Deployment and Maintenance:-

- **Objective :** To deploy the application and ensure its smooth operation and future enhancements.
- **Activities :**
  - Deploy the backend server and database on a local server or cloud platform.
  - Host the frontend application for user access.
  - Monitor system performance and security.
  - Provide maintenance through bug fixes, updates, and future feature additions such as parking or vendor management.

4. SYSTEM PLANNING

4.1 Feasibility Study

A **Feasibility Study** plays a crucial role in the development of any software system. In the case of the **Home Owner Association (HOA) Connect Portal**, it was essential to evaluate the system's practicality in terms of operational impact, technical requirements, available resources, and time constraints before initiating development.

Before the development phase began, a detailed feasibility analysis was conducted to assess whether the proposed system would meet the needs of residential communities while remaining cost-effective, reliable, and scalable.

Operating Feasibility:

The operational feasibility study focused on evaluating how the HOA Connect system would integrate with the day-to-day functioning of residential communities. The system was designed to simplify administrative tasks such as resident management, payment tracking, complaint handling, and amenity booking without disrupting existing community operations.

Technical Feasibility:

The technical feasibility assessment examined the hardware and software requirements necessary for developing and deploying the HOA Connect portal. Based on project requirements and available infrastructure, the **MERN Stack** was selected:

- **Frontend:** React.js
- **Backend:** Node.js with Express.js
- **Database:** MongoDB
- **Development Tools:** Visual Studio Code, Postman

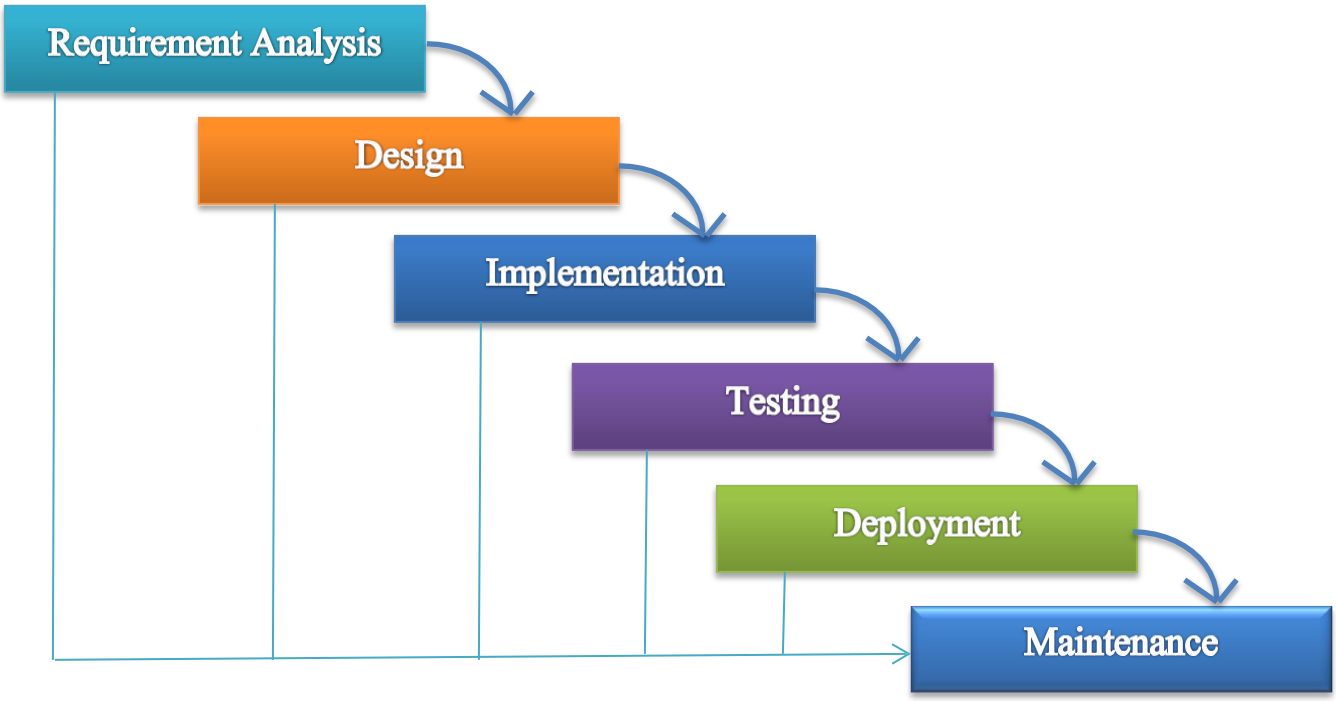
Resources:

The successful implementation of the HOA Connect project required adequate technical resources, skilled personnel, and network infrastructure. The development team possessed prior experience in web-based application development using the MERN stack, which contributed to smooth system design and implementation.

Time Duration:

A total duration of **six months** was allocated for the completion of the HOA Connect system. This time frame included phases such as requirement analysis, system design, development, testing, and deployment. A structured development schedule with defined milestones was followed to ensure timely completion of each phase

4.2 Software Engineering Model

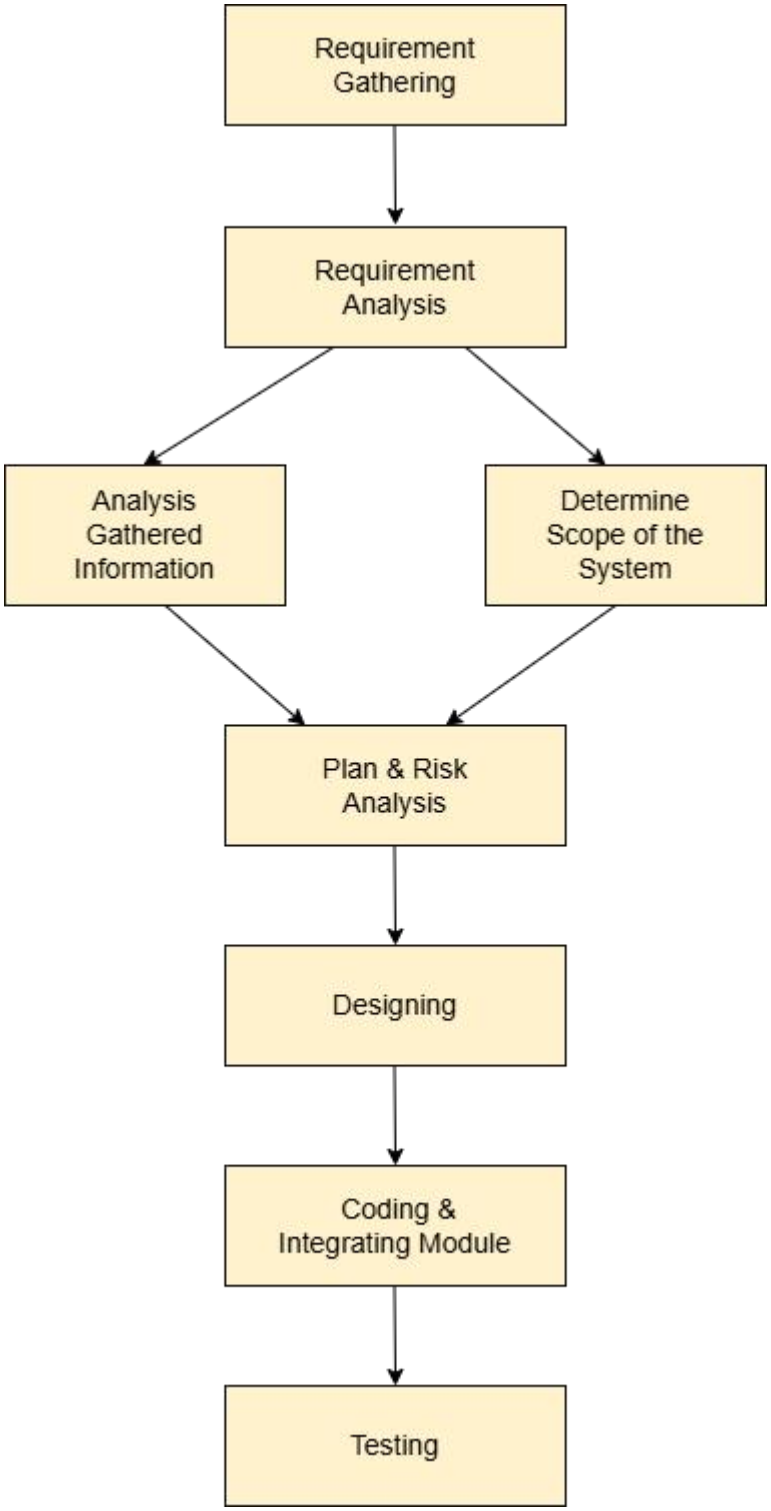


4.3 Risk Analysis

- The main objective of risk analysis is to identify potential problems that may affect the successful development and operation of the **HOA Connect web application**.
- Risk classification helps in understanding different risk factors and allows project managers to focus on areas that may cause maximum impact on the system.
- Risks are grouped based on similar characteristics, which makes analysis easier and helps in planning effective risk control strategies.
- In web-based applications like HOA Connect, risks can be broadly classified into **internal risks** and **external risks**.
- **Internal risks** originate within the project or organization, while **external risks** arise from factors outside the control of the development team.
- Proper planning, regular monitoring, and use of secure and reliable technologies help in minimizing these risks and ensuring successful system implementation.

4.4 Project Schedule

4.4.1 Task Dependency :



4.4.2 Project Table:

Phase	Task Description	Responsible Team	Deliverables
Requirement Analysis	1. Collect requirements from stakeholders 2. Analyze user, admin, and member needs 3. Define scope of application features	Project Manager, Analyst	Requirement Specification DocumentScope Statement
Planning & Risk Analysis	1. Identify risks (data load, integration, scope creep) 2. Plan technical stack and tools 3. Break down system into modules	Project Manager, Tech Lead	Risk RegisterModule Planning Sheet
Designing	1. Design UI/UX wireframes for all panels (Admin, User, Member) 2. Create ER Diagram and Class Diagram from database design 3. Define web page navigation and flow	UI/UX Designer, Database Designer	WireframesER DiagramClass Diagram
Development	1. Develop login/signup functionality with role-based access 2. Implement system modules and features 3. Integrate backend with database	Backend Developers, Full Stack Developers	Fully functional application with integrated modules
Testing	1. Unit testing of individual modules 2. Integration testing across modules 3. User Acceptance Testing (UAT) with feedback	QA Team, Testers	Test PlanBug ReportsFinal UAT Report
Deployment	1. Setup application on local or remote server 2. Deploy database and APIs 3. Perform smoke testing	Developer, DevOps (if applicable)	Deployed Working SystemHosting Information
Documentation	1. Final project report preparation 2. User manual with screenshots 3. Source code summary and documentation	Documentation Team, Analyst	Final Project ReportUser ManualCode Summary

5. SYSTEM ANALYSIS

5.1 Detailed SRS

Actor :

- ❖ Super Admin (Platform Owner)
- ❖ HOA Admin (Community Admin)
- ❖ Residents (Users)

❖ Super Admin :

- ➢ Purpose: To manage, monitor, and oversee the entire HOA Connect platform across multiple residential communities.
- ➢ Features :

  - Platform Management :
    - Create, update, and delete residential communities.
    - Assign and replace HOA Admins while creating or managing communities.
    - Define available amenities for each community.
  - Community Oversight:
    - Monitor activity across all communities.
    - Override or replace HOA Admins when required.
  - Reports & Analytics:
    - View global analytics such as complaints, and community activity.
    - Generate consolidated reports across all communities.
  - Notification Management:
    - Send system-wide announcements and notifications.
  - Security & Control:
    - Ensure role-based access control and data isolation between communities.

❖ HOA Admin :

- Purpose: To manage and administer a specific residential community and handle its day-to-day operations.
- Features :

  - Community Management :
    - Manage residents within the assigned community (view, add, remove).
    - Access all features strictly limited to the assigned community.
  - Amenity Management :
    - Manage existing amenities defined by Super Admin (view, activate, deactivate).
    - Monitor amenity availability and bookings.
  - Complaint Management :
    - View, update, and resolve complaints raised by residents.
  - Announcements & Notifications :
    - Create announcements visible to community residents.
    - Send community-wide notifications.
  - Payments Management :
    - Manage and monitor maintenance payments for the community.
  - Meetings & Polls :
    - Organize meetings (agenda, date, mode).
    - Create and manage polls for community decisions.
  - Document Management :
    - Upload and share important documents such as rules, notices, and meeting minutes.
  - Admin Replacement:
    - Replace themselves with another resident as HOA Admin when required.
  - Reports & Analytics:
    - View community-level analytics including payments, complaints, and bookings.

❖ Resident :

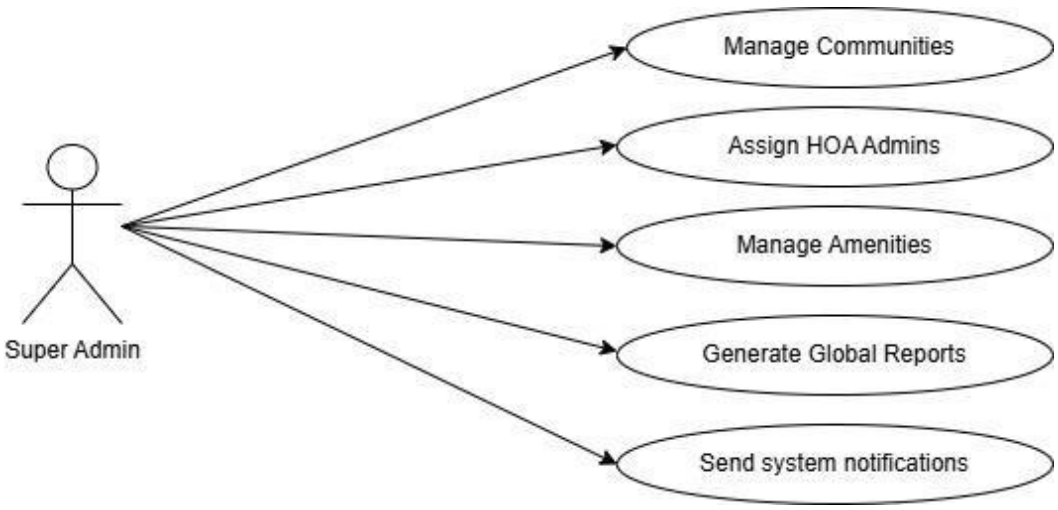
- Purpose: To participate in community activities and access HOA services conveniently.
- Features :

  - Self-Service Dashboard :
    - View announcements and community updates.
    - Manage personal profile details.
  - Complaint Management :
    - Raise new complaints and track their resolution status.
  - Payments :
    - Make online maintenance payments.
    - View payment history and receipts.
  - Amenity Booking:
    - View available amenities and book slots.
  - Meetings & Polls :
    - View meeting details.
    - Participate in polls and voting processes.
  - Documents Access:
    - Download community documents and notices.
  - Notifications:
    - Receive alerts related to payments, complaints, meetings, and announcements.

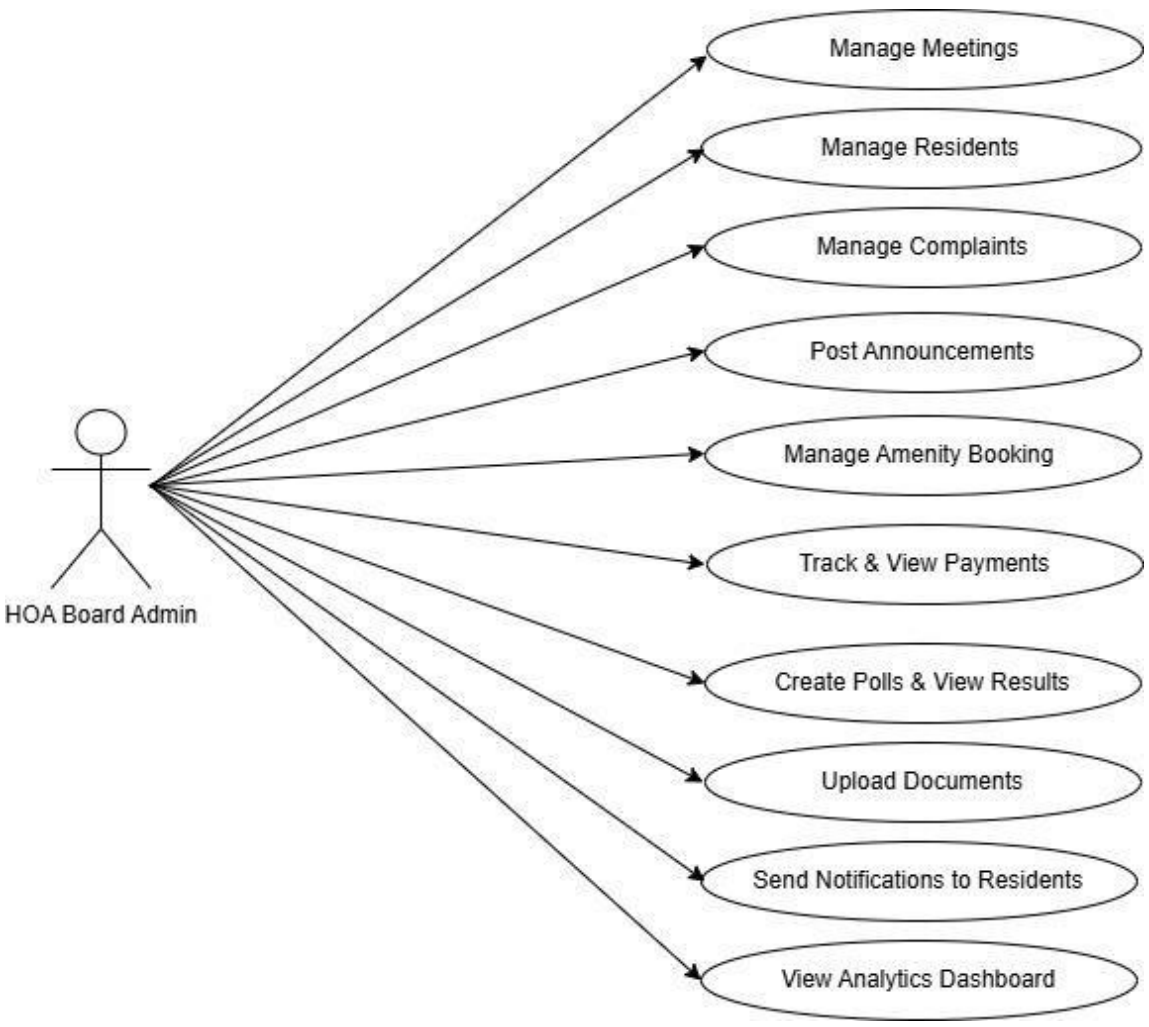
5.2 UML Diagram

5.2.1 Use Case Diagram:

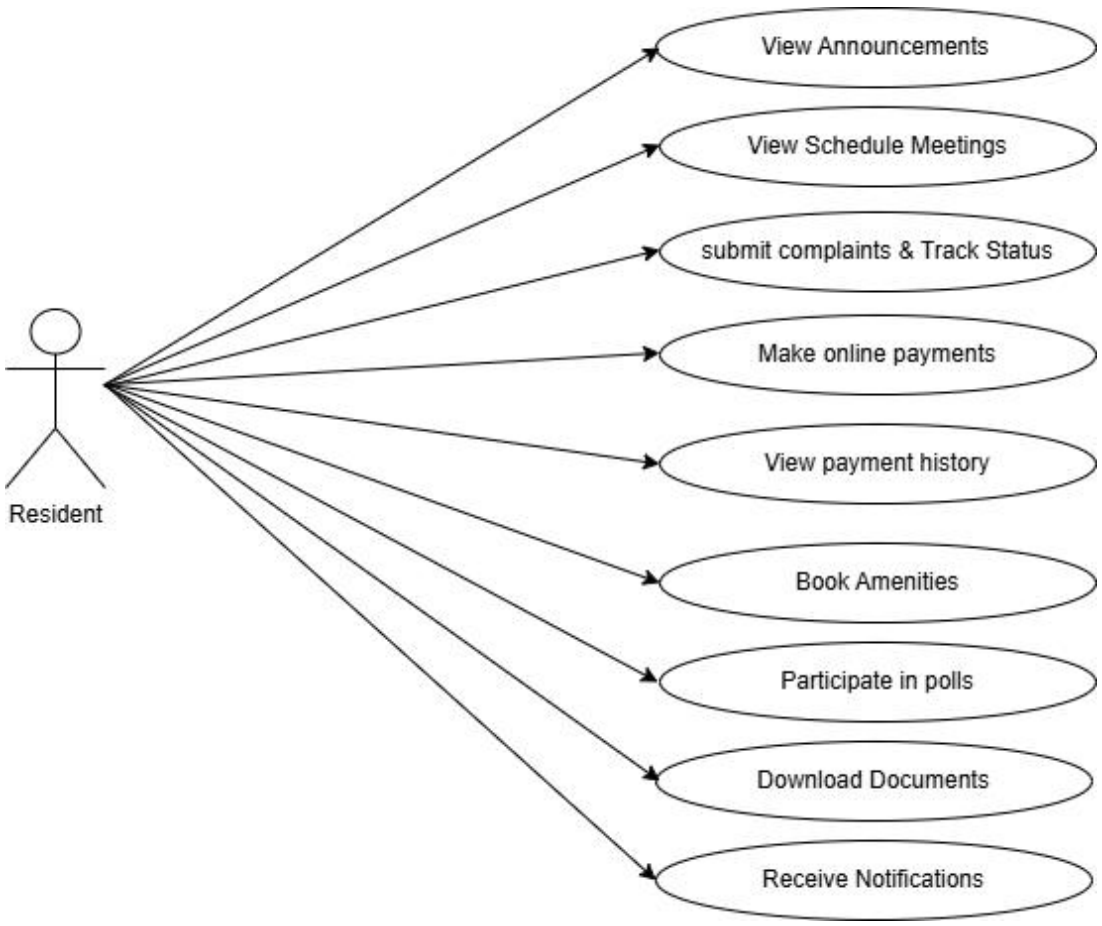
- **Super Admin:**



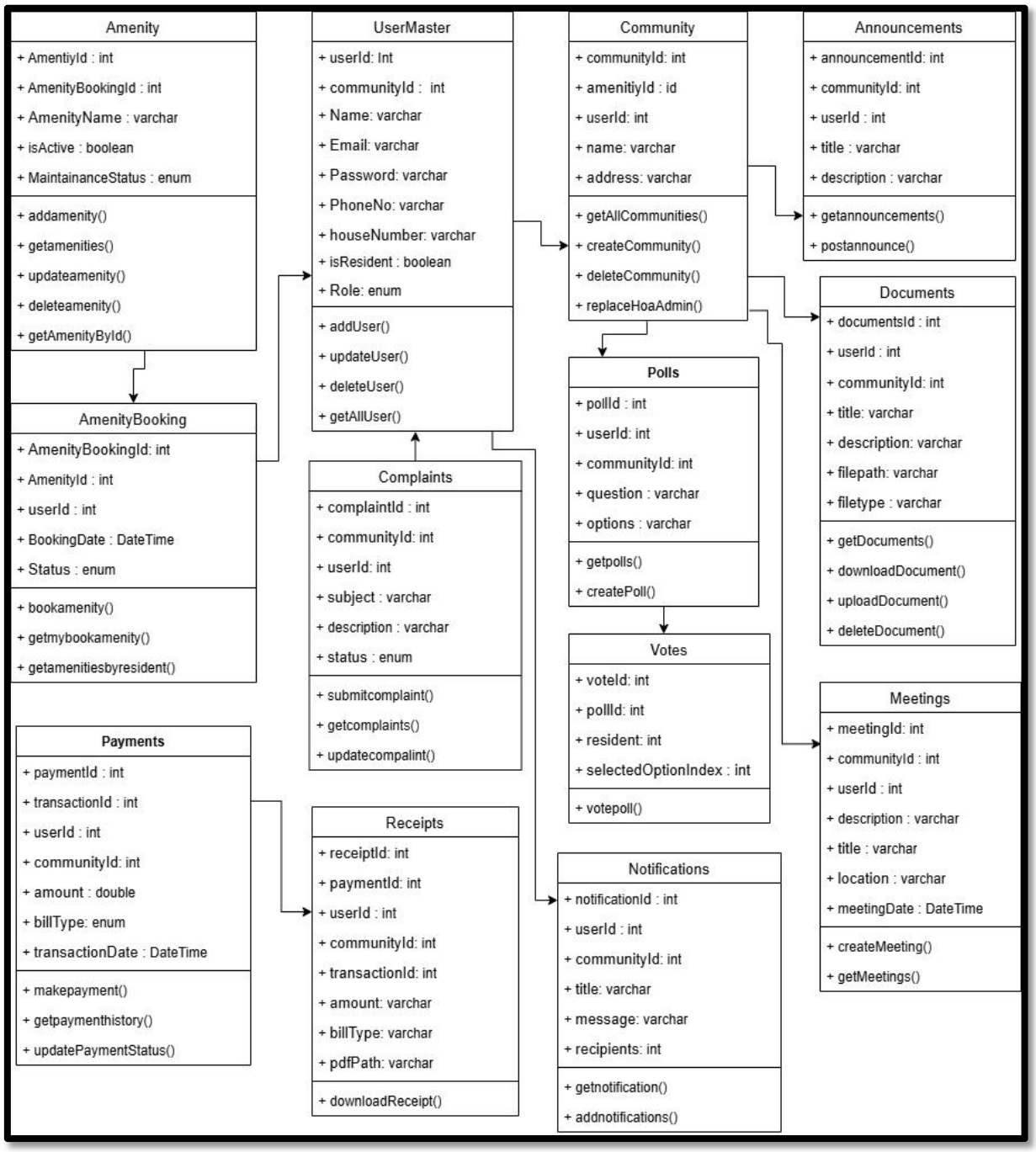
❖ **HOA Admin:**


HOA Board Admin
Manage Meetings
Manage Residents
Manage Complaints
Post Announcements
Manage Amenity Booking
Track & View Payments
Create Polls & View Results
Upload Documents
Send Notifications to Residents
View Analytics Dashboard

❖ **Resident:**

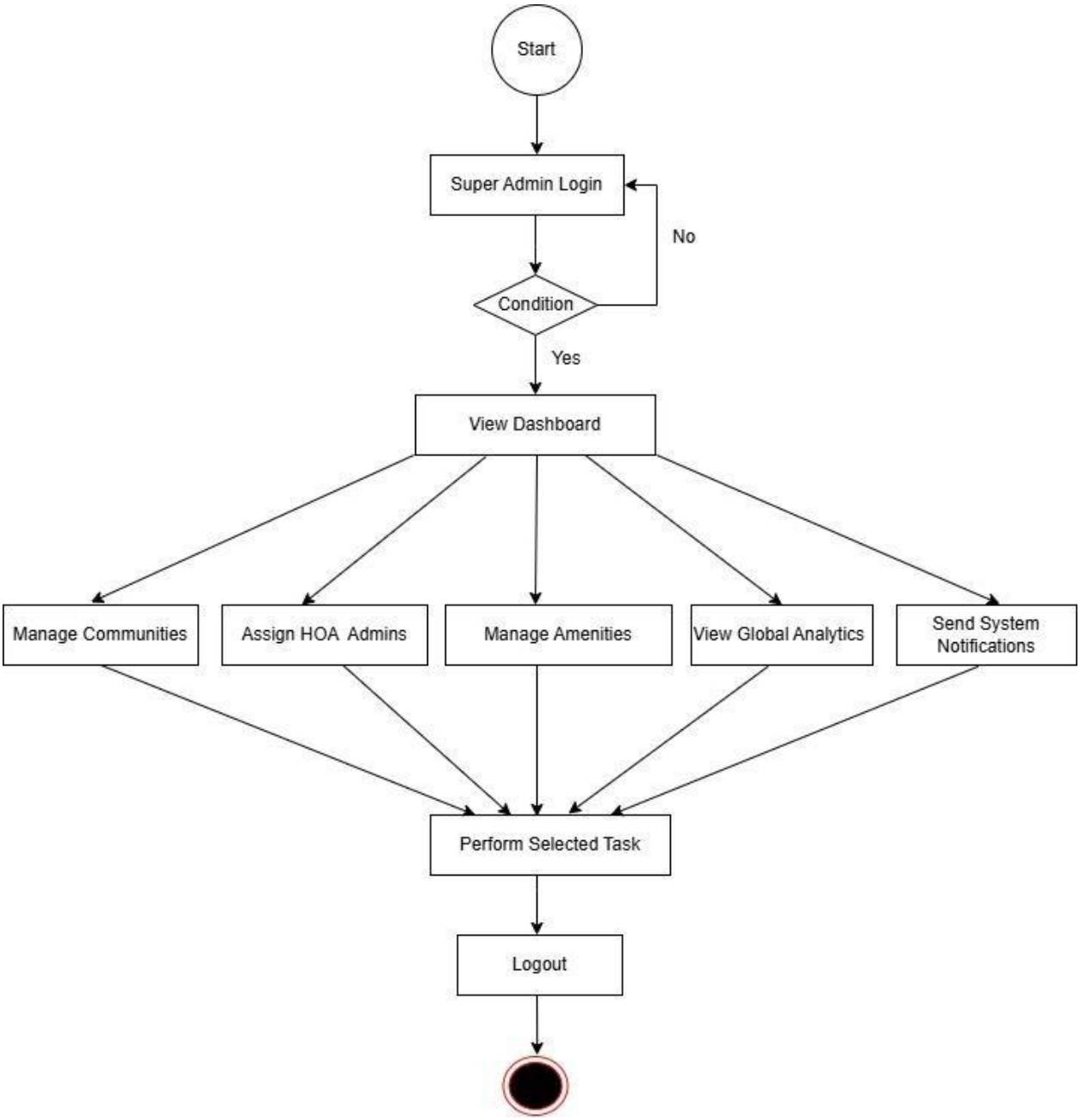

View Announcements
View Schedule Meetings
submit complaints & Track Status
Make online payments
View payment history
Book Amenities
Participate in polls
Download Documents
Receive Notifications
Resident

5.2.2 Class Diagram:

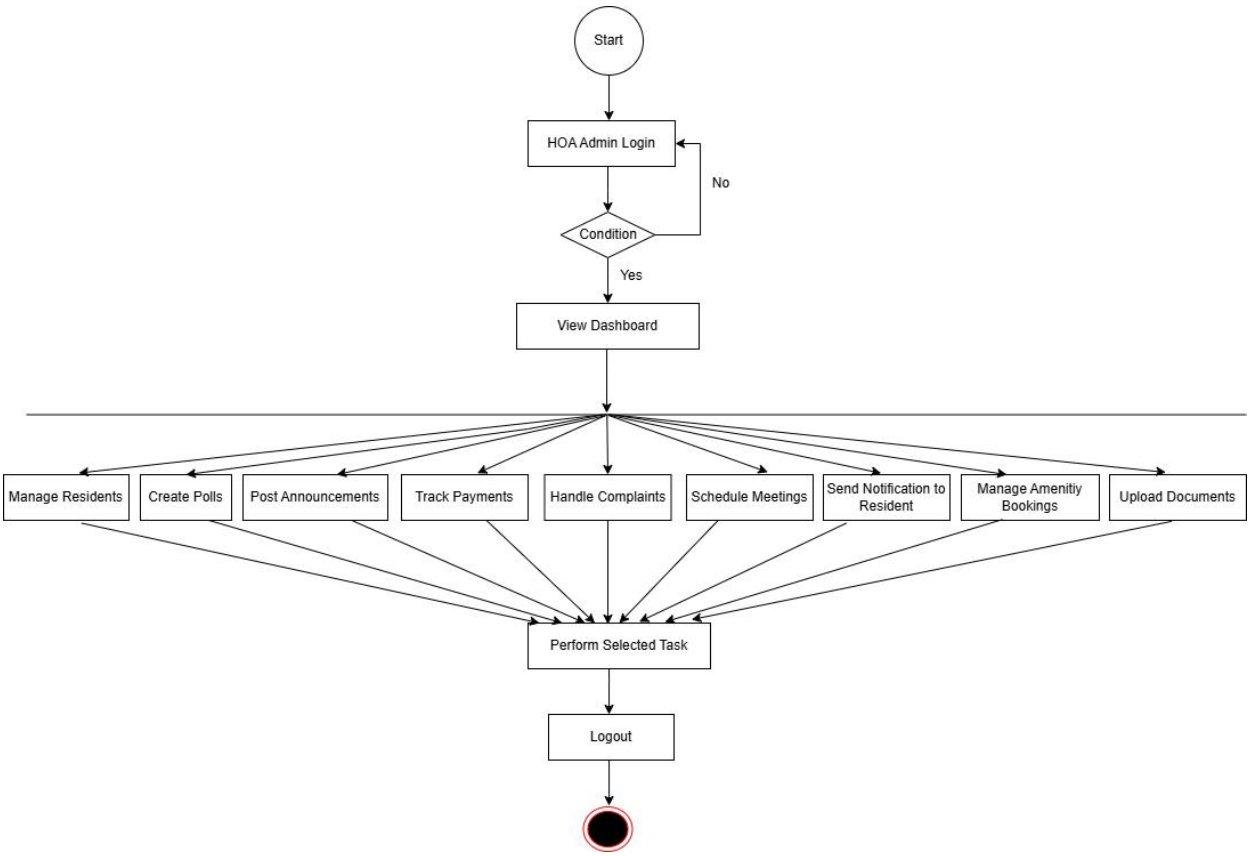


5.2.3 Activity Diagram:

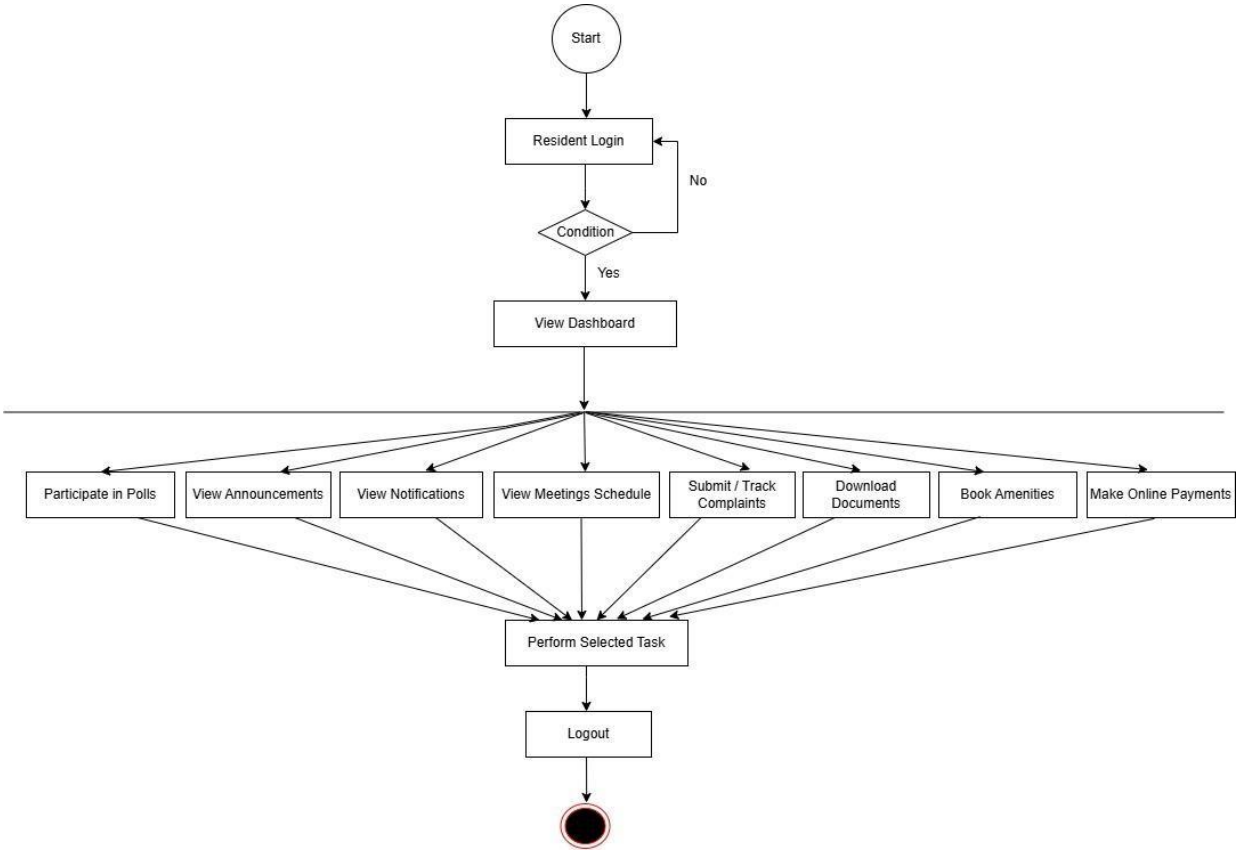
❖ Super Admin:



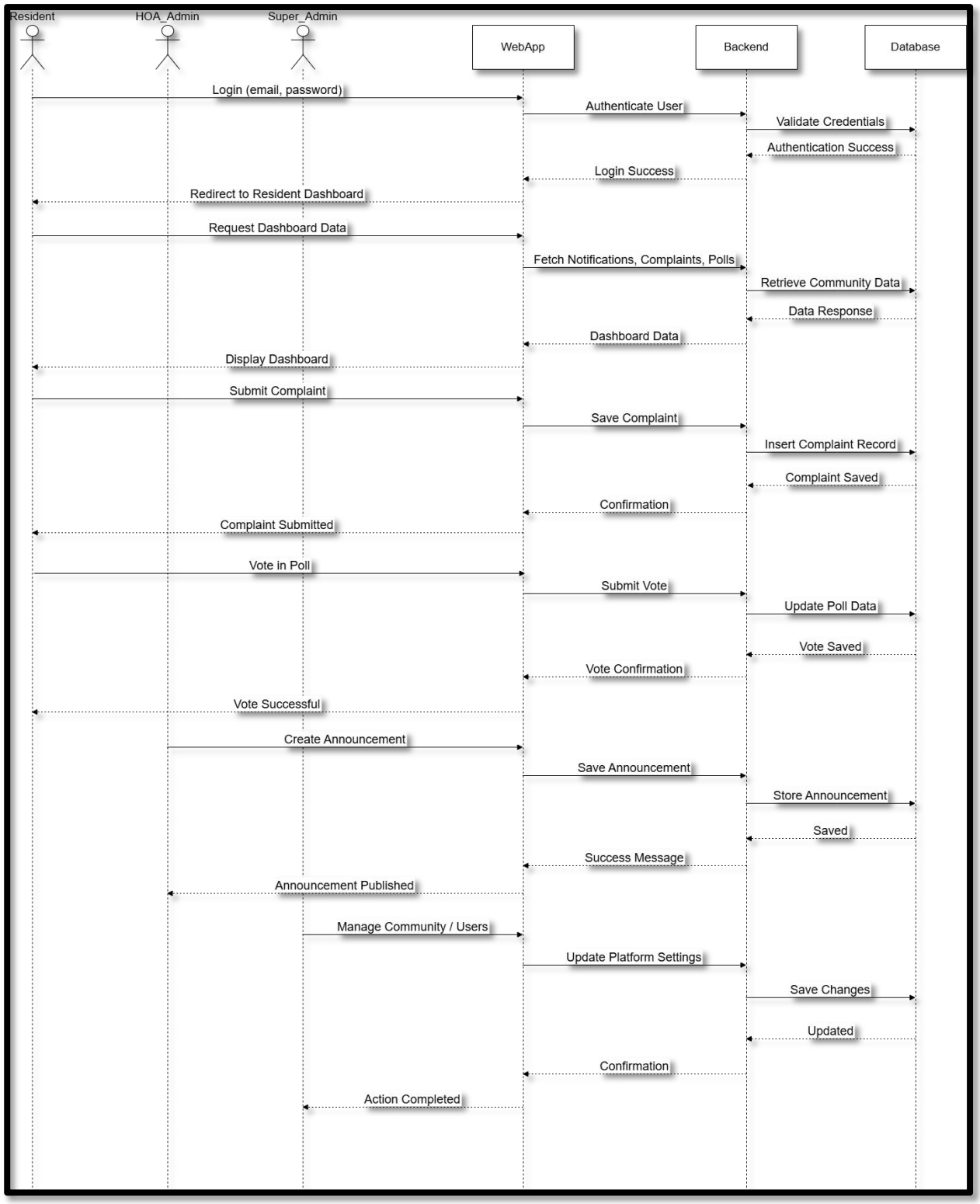
❖ HOA Admin:


Start
HOA Admin Login
No
Condition
Yes
View Dashboard
Manage Residents
Create Polls
Post Announcements
Track Payments
Handle Complaints
Schedule Meetings
Send Notification to Resident
Manage Amenitiy Bookings
Upload Documents
Perform Selected Task
Logout

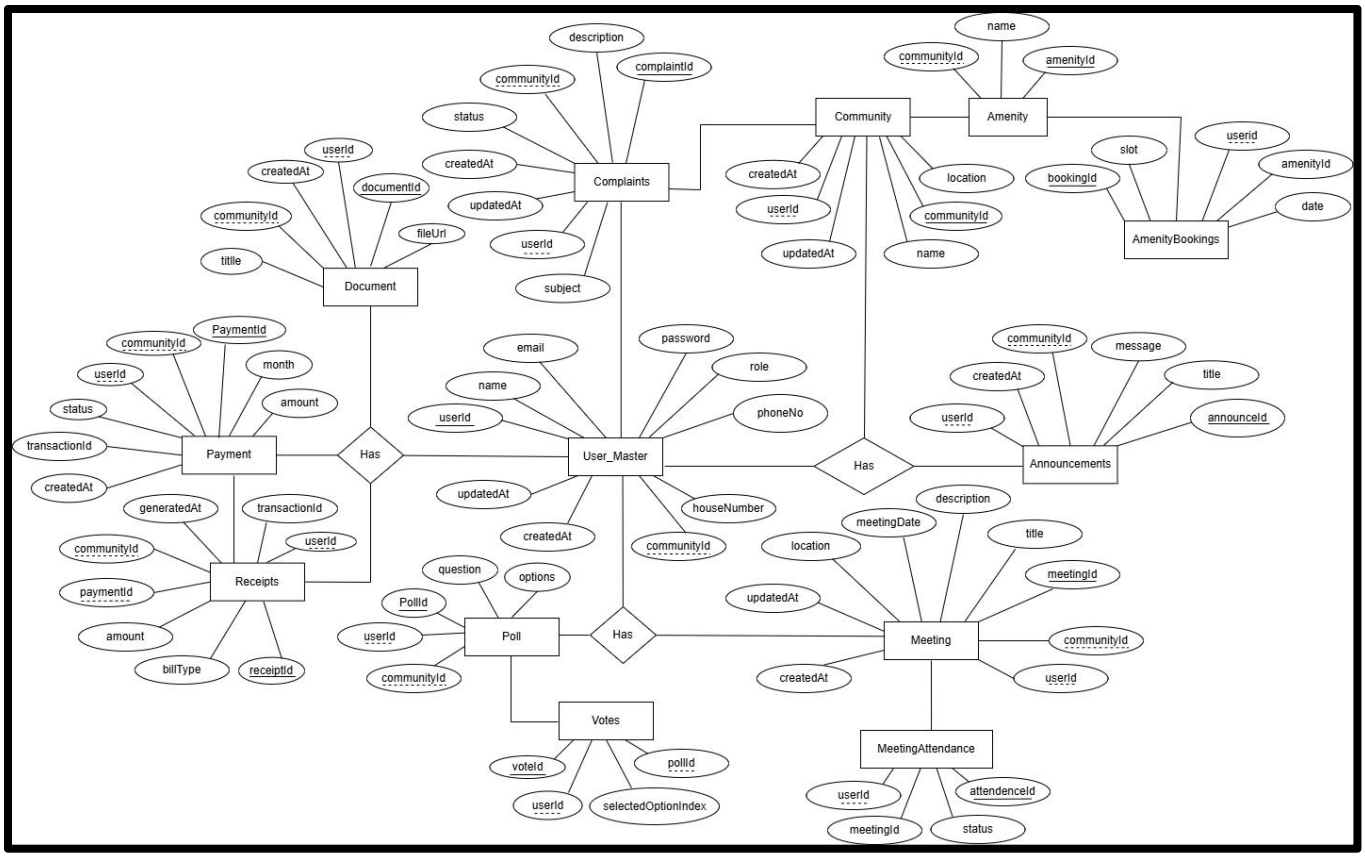
❖ **Resident:**


Start
Resident Login
No
Condition
Yes
View Dashboard
Participate in Polls
View Announcements
View Notifications
View Meetings Schedule
Submit / Track Complaints
Download Documents
Book Amenities
Make Online Payments
Perform Selected Task
Logout

5.2.4 Sequence Diagram:



5.3 ER Diagram



6. SOFTWARE DESIGN

6.1 Database Design

1. User :

Column Name	Data Type	Size	Constraints
userId	int		Primary Key
name	varchar	50	
email	varchar	100	
mobileNo	varchar	100	
password	varchar	255	
role	enum		
houseNo	varchar	100	
communityId	int		Foreign Key
isResident	boolean		

2. Amenity :

Column Name	Data Type	Size	Constraints
amenityId	int		Primary Key
name	varchar	50	
description	varchar	100	
isActive	boolean		
bookingId	int		Foreign Key

3. AmenityBooking :

Column Name	Data Type	Size	Constraints
bookingId	int		Primary Key
bookingDate	date		
status	enum	100	
isActive	boolean		
amenityId	int		Foreign Key
userId	int		Foreign Key

4. Announcement :

Column Name	Data Type	Size	Constraints
announcementId	int		Primary Key
title	varchar	100	
description	varchar	150	
amenityId	int		Foreign Key
userId	int		Foreign Key

5. Community :

Column Name	Data Type	Size	Constraints
communityId	int		Primary Key
name	varchar	100	
address	varchar	150	
amenityId	int		Foreign Key
userId	int		Foreign Key

6. Complaint :

Column Name	Data Type	Size	Constraints
complaintId	int		Primary Key
subject	varchar	100	
description	varchar	150	
status	enum		
communityId	int		Foreign Key
userId	int		Foreign Key

7. Document :

Column Name	Data Type	Size	Constraints
documentId	int		Primary Key
title	varchar	100	
filePath	varchar	255	
fileType	varchar	100	
communityId	int		Foreign Key
userId	int		Foreign Key

8. Meeting:

Column Name	Data Type	Size	Constraints
meetingId	int		Primary Key
title	varchar	100	
description	varchar	200	
meetingDate	date		
location	varchar	100	
communityId	int		Foreign Key
userId	int		Foreign Key

9. Notification :

Column Name	Data Type	Size	Constraints
notificationId	int		Primary Key
title	varchar	100	
message	varchar	150	
communityId	int		Foreign Key
userId	int		Foreign Key

10. Payment:

Column Name	Data Type	Size	Constraints
paymentId	int		Primary Key
amount	double		
method	enum		
status	enum		
billType	enum		
transactionId	int		Unique
transactionDate	date		
communityId	int		Foreign Key
userId	int		Foreign Key

11. Poll :

Column Name	Data Type	Size	Constraints
pollId	int		Primary Key
question	varchar	100	
options	enum		
communityId	int		Foreign Key
userId	int		Foreign Key

12. Reciept :

Column Name	Data Type	Size	Constraints
recieptId	int		Primary Key
amount	double		
billType	varchar	100	
pdfPath	varchar	150	
transactionId	int		
paymentId	int		Foreign Key
communityId	int		Foreign Key
userId	int		Foreign Key

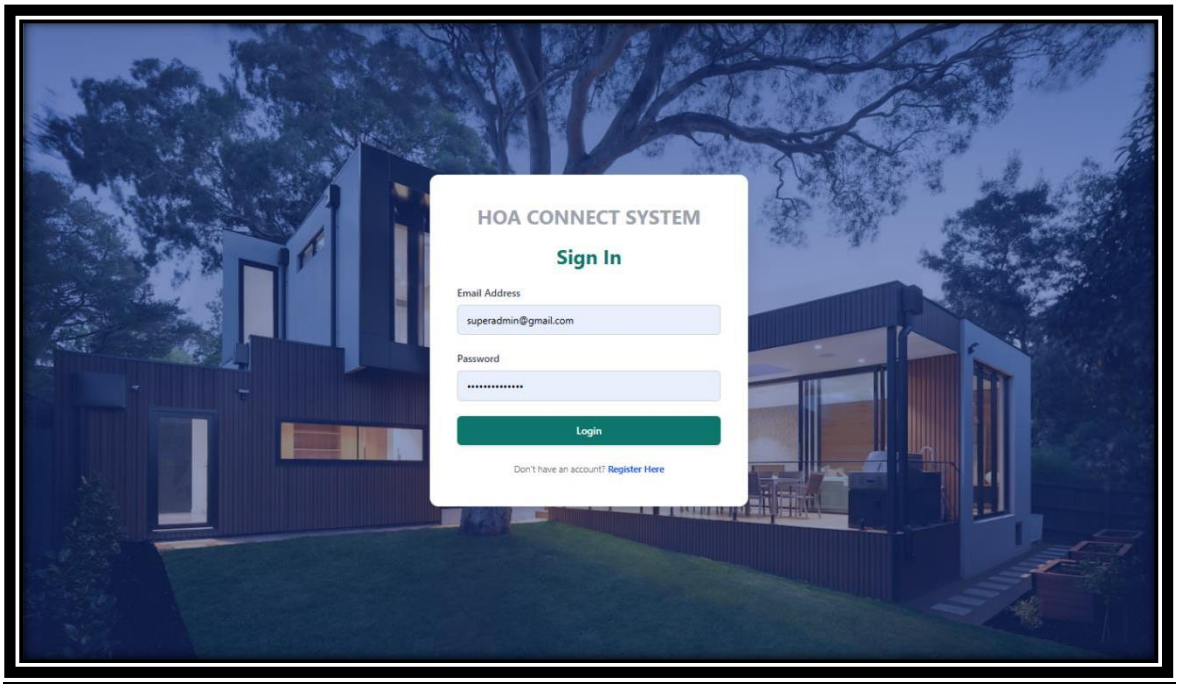
13. Vote :

Column Name	Data Type	Size	Constraints
voteId	int		Primary Key
selectedOption	varchar	100	
pollId	int		Foreign Key
userId	int		Foreign Key

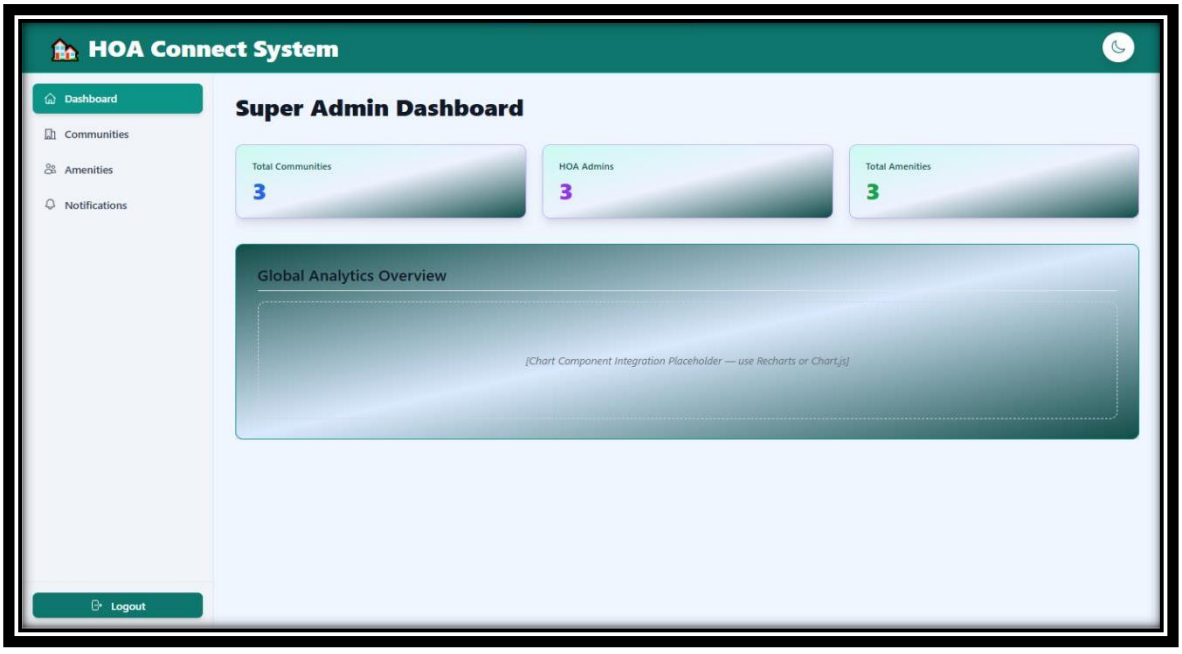
6.2 Interface Design

❖ Super Admin :

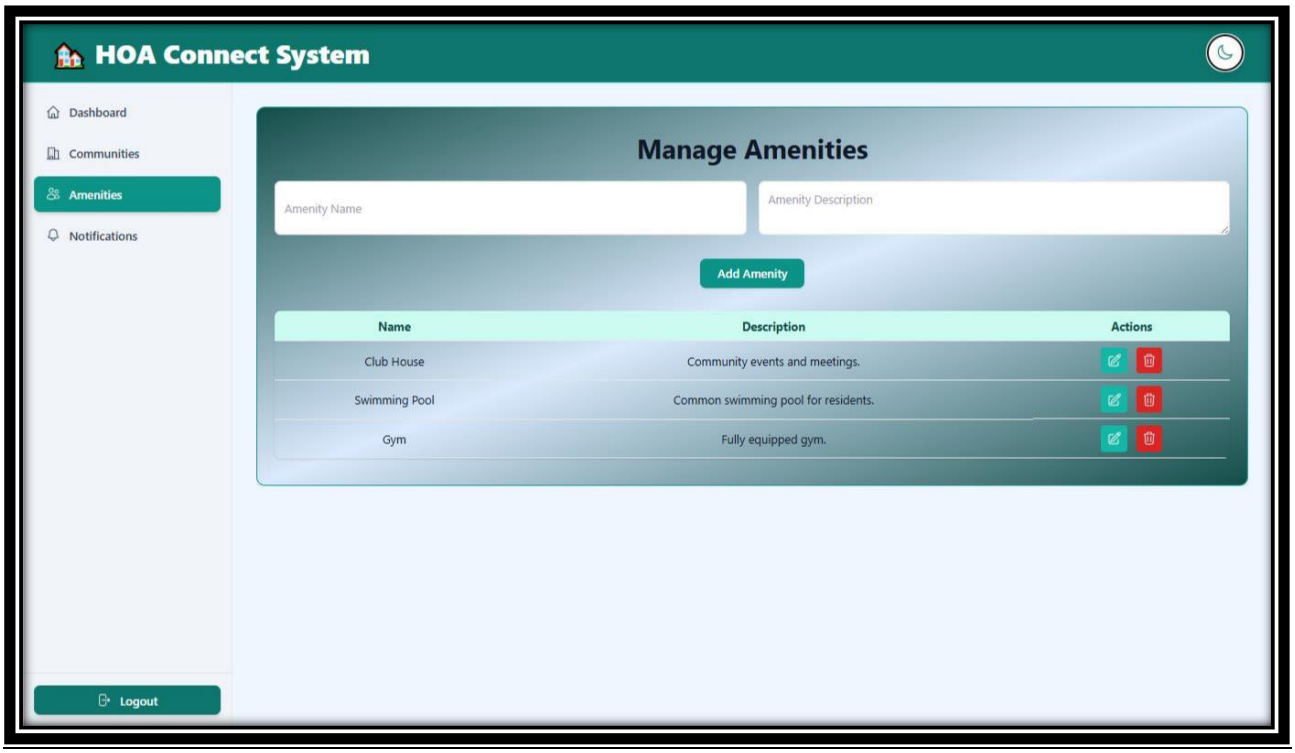
➢ Login :-



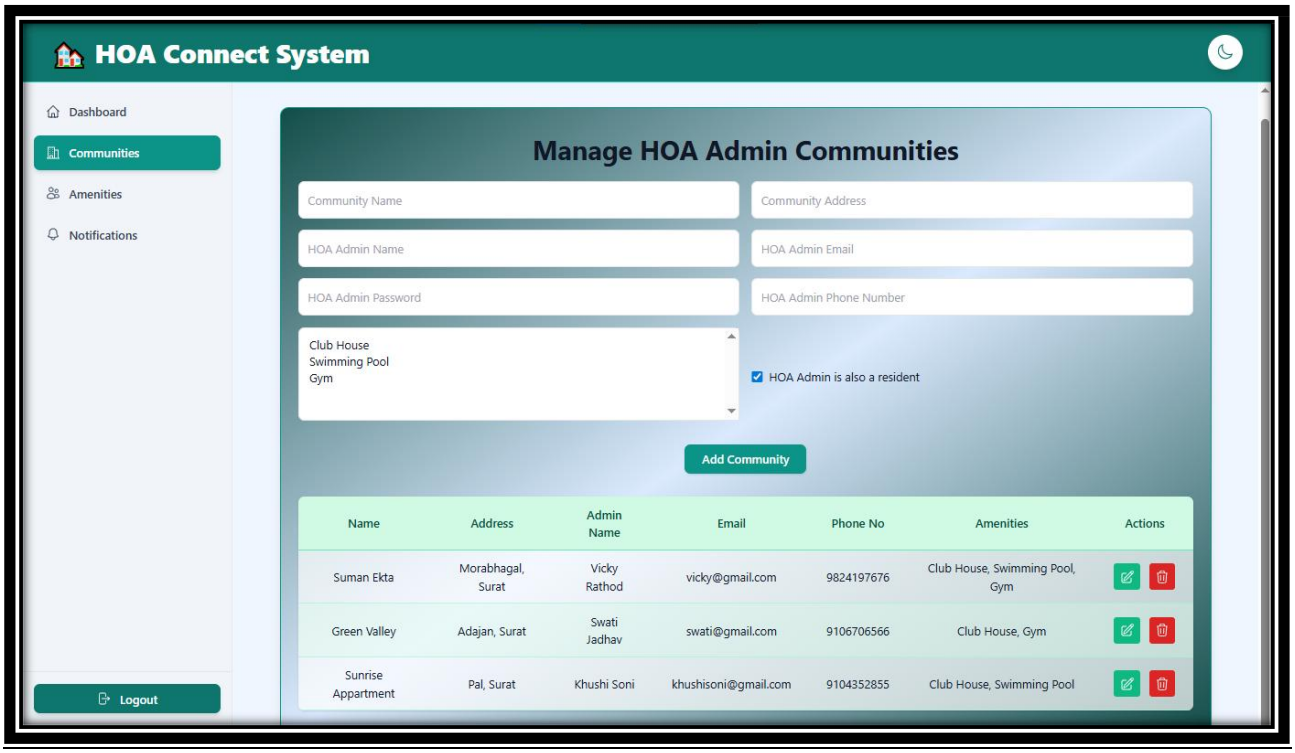
➢ Dashboard :-



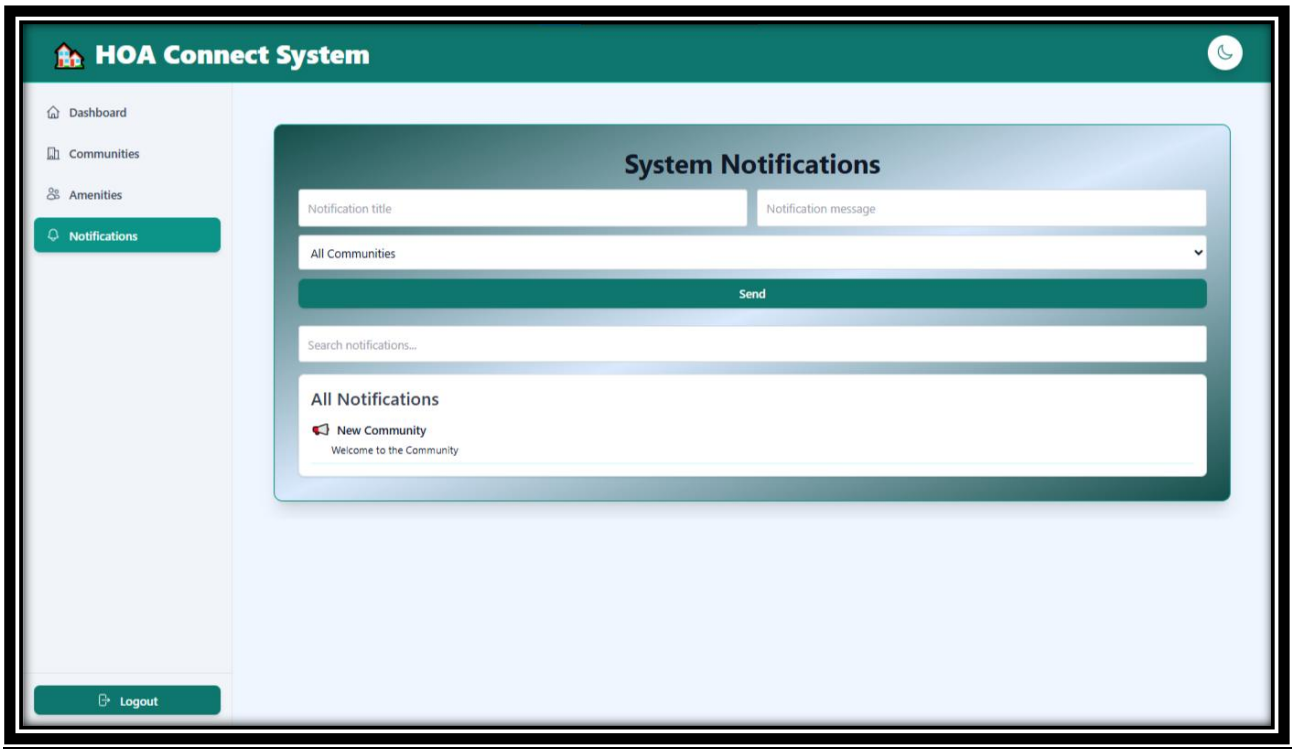
➢ Manage Amenity :-



➢ Manage Community and HOA Admin :-

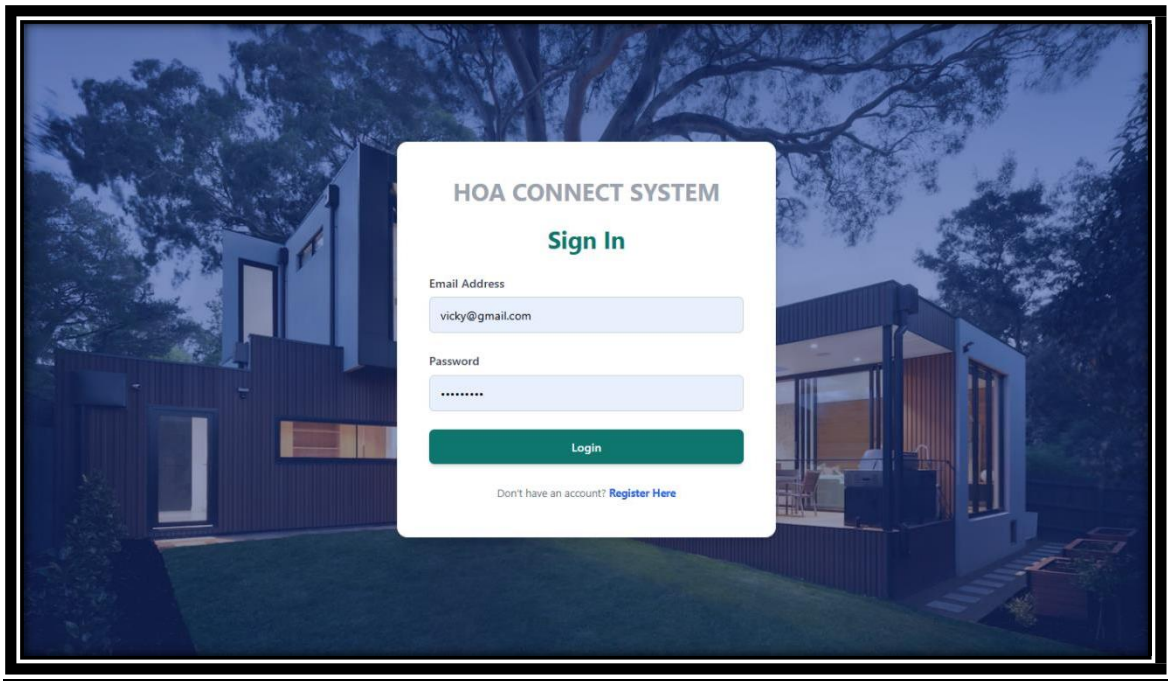


- **Manage System-wide Notifications:-**

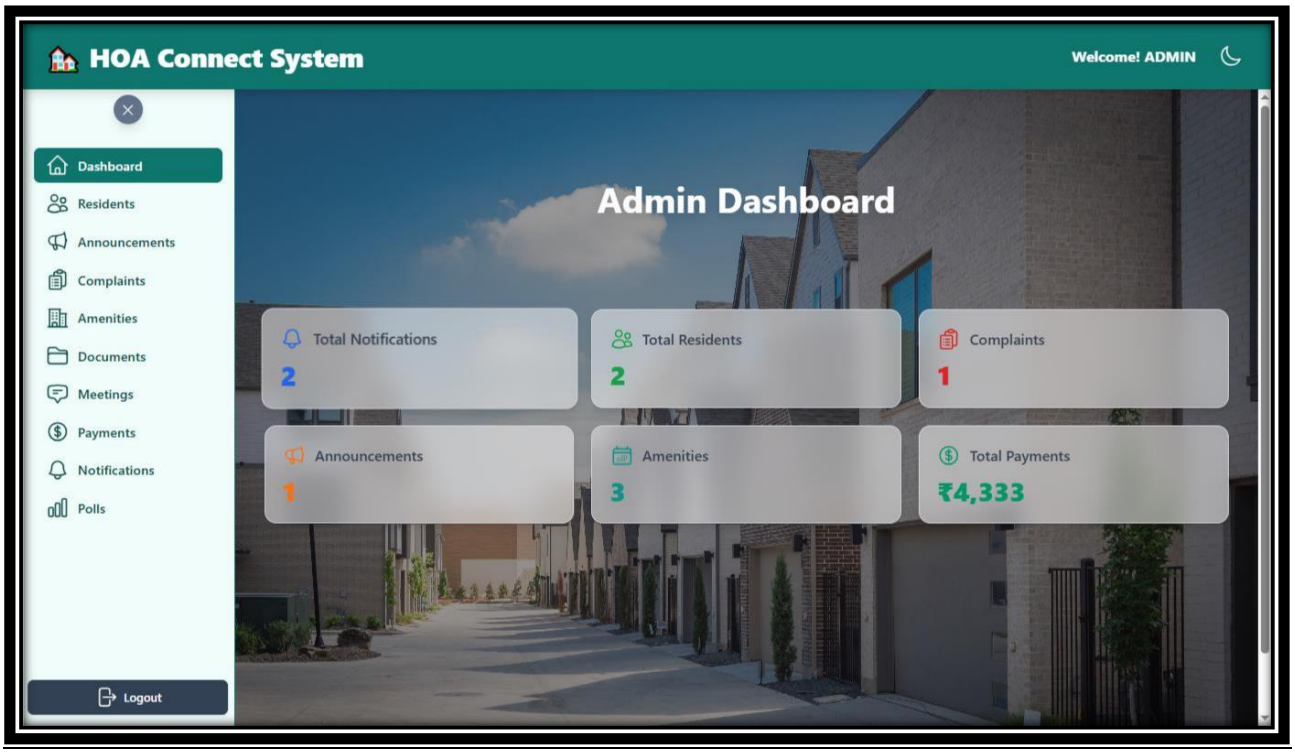


❖ HOA Admin :

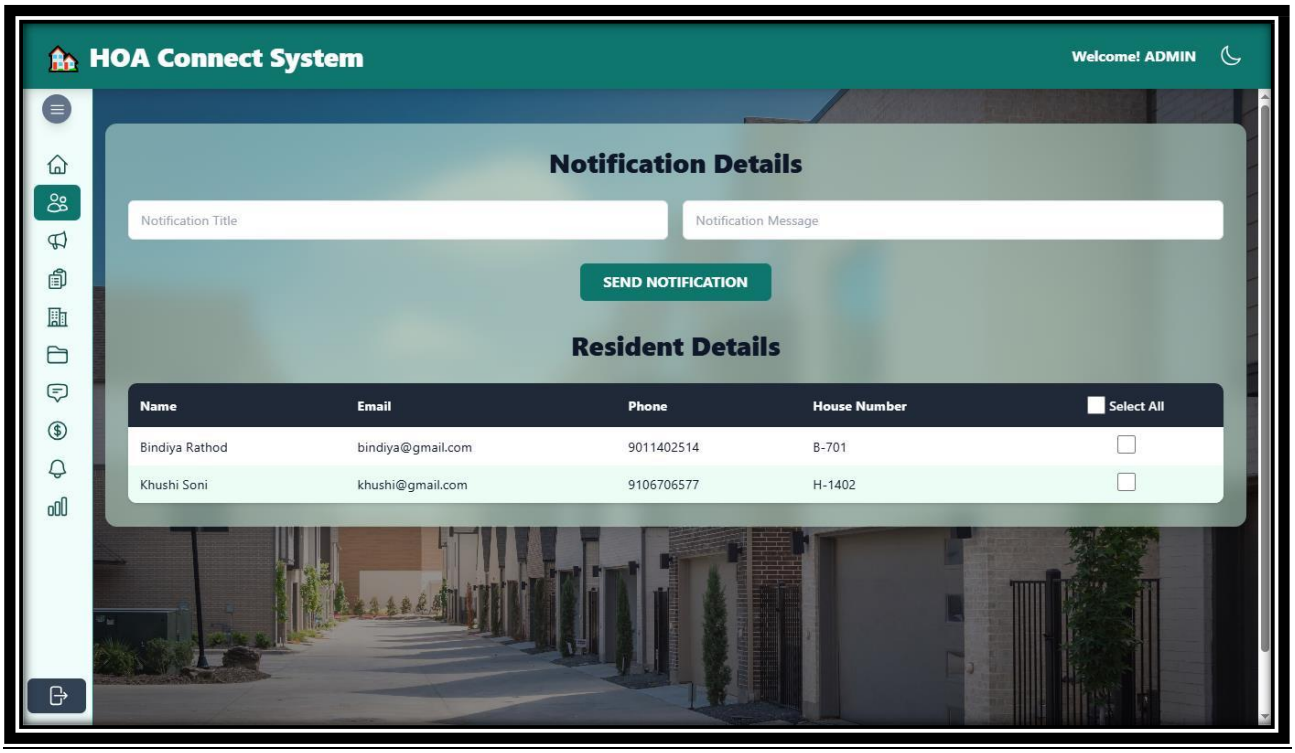
- **Login :-**



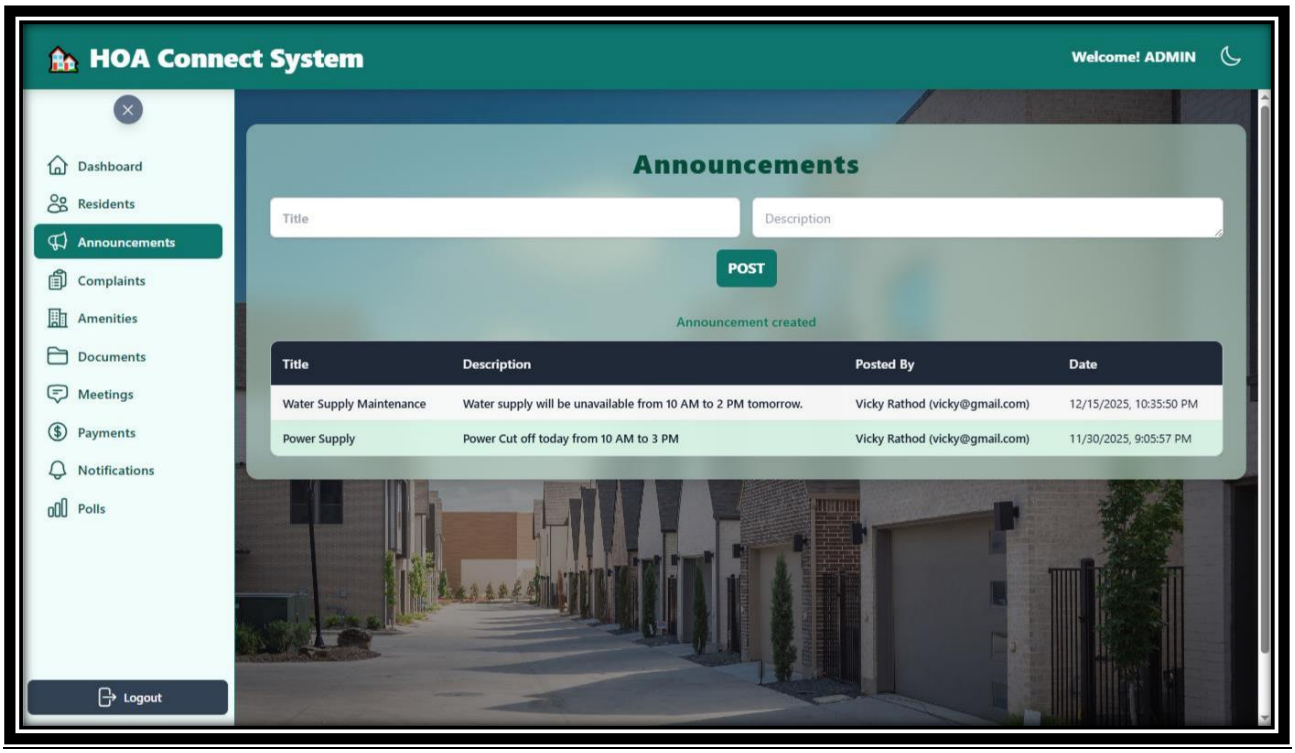
➢ Dashboard:-



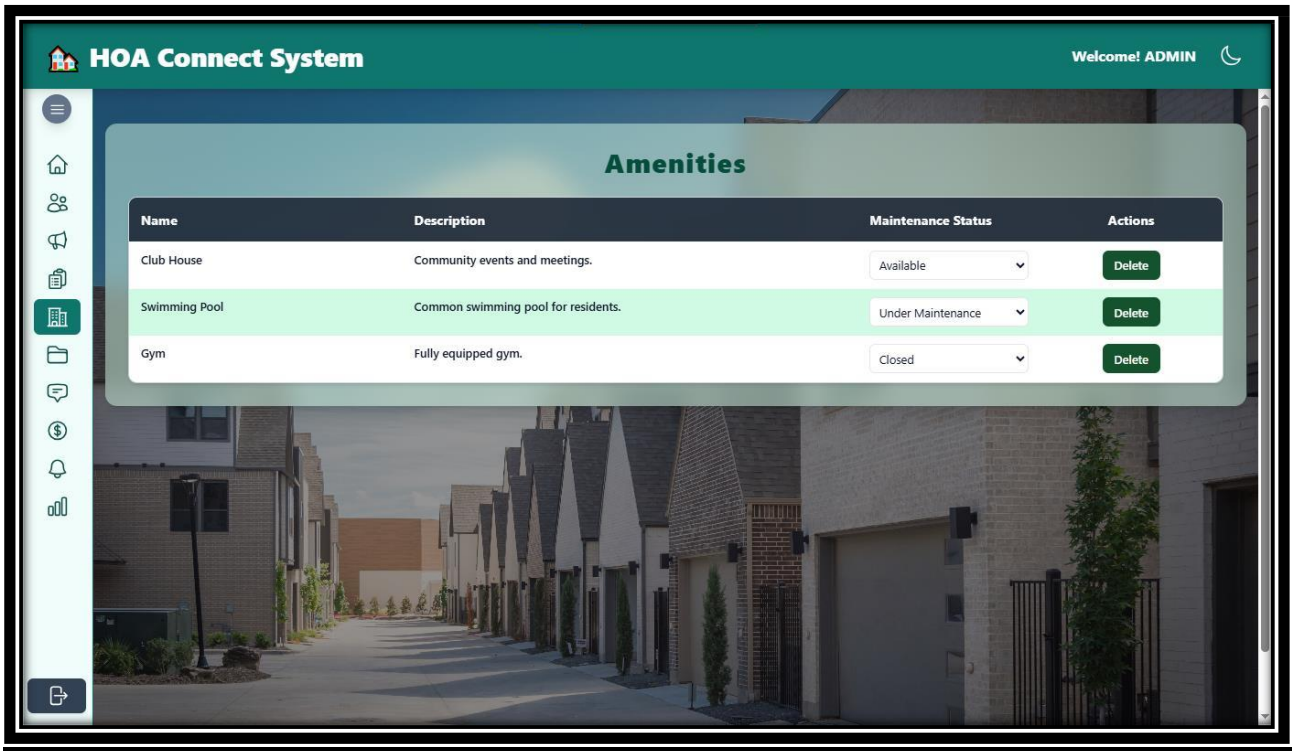
➢ Manage Notification:-



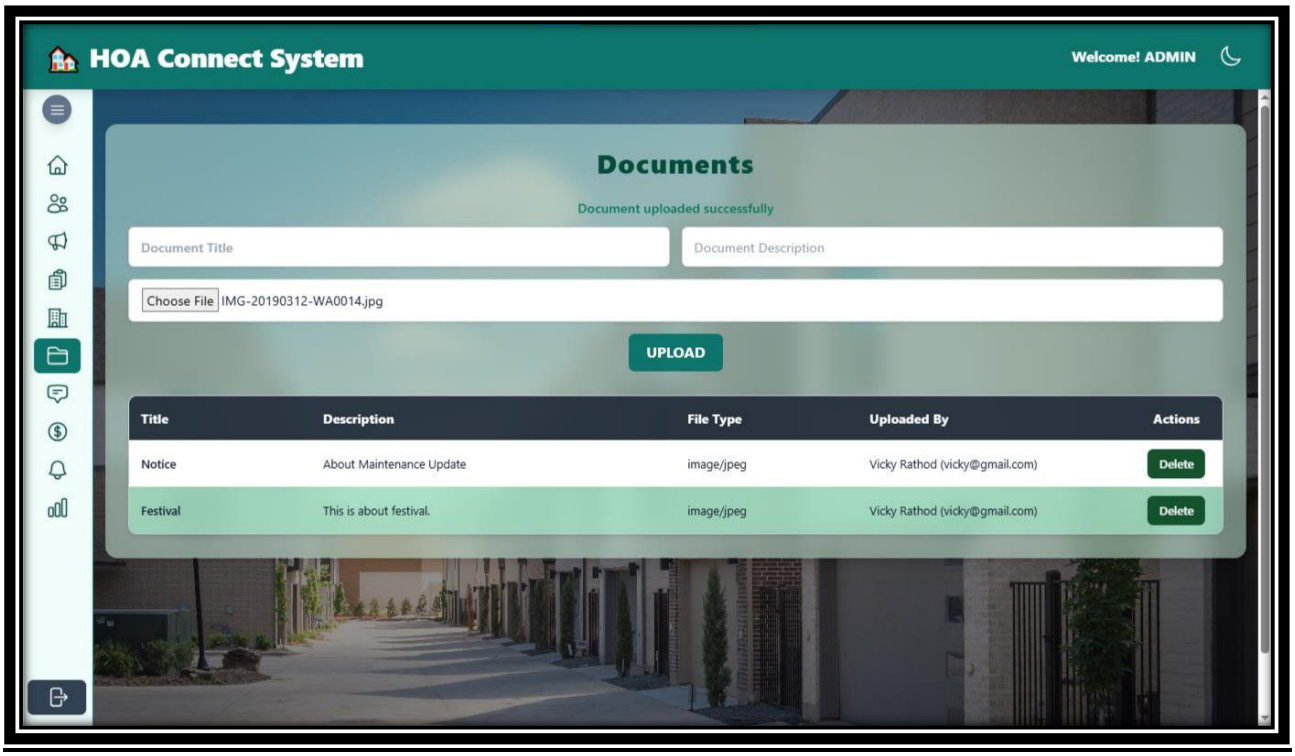
➢ Manage Announcement:-



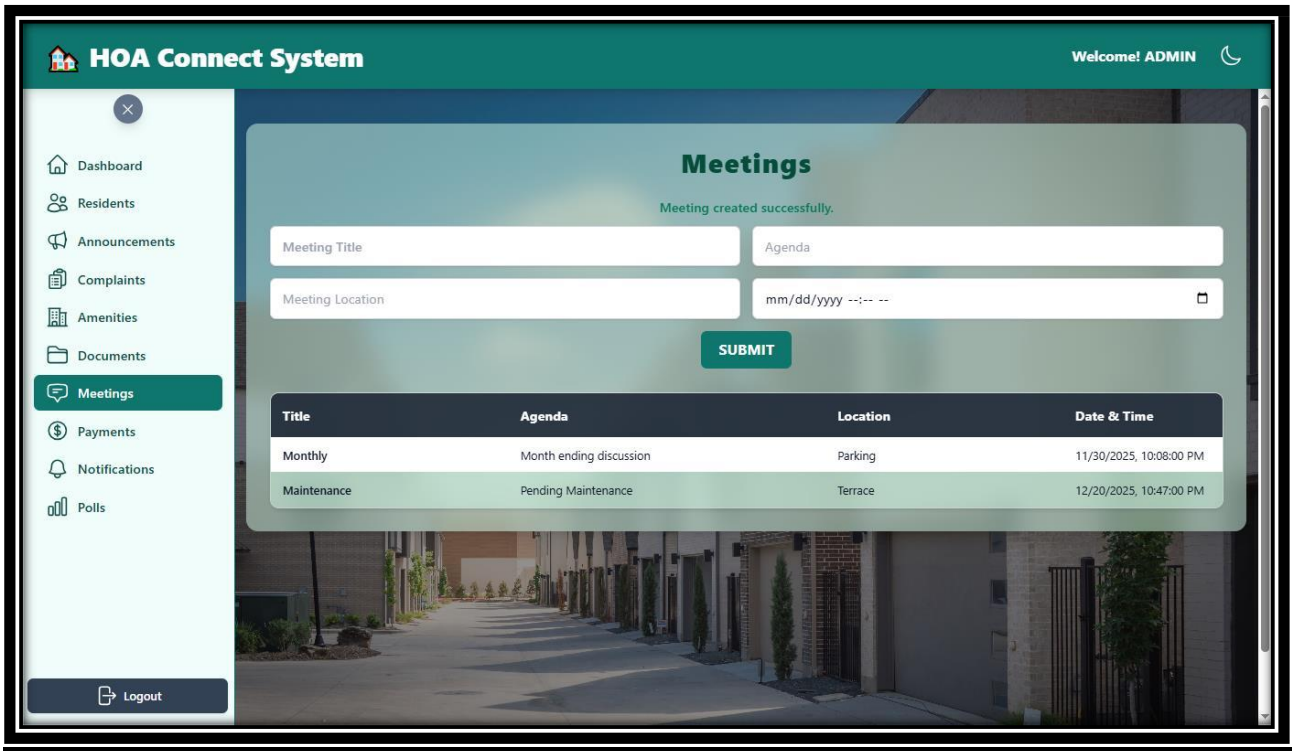
➢ Manage Amenities:-



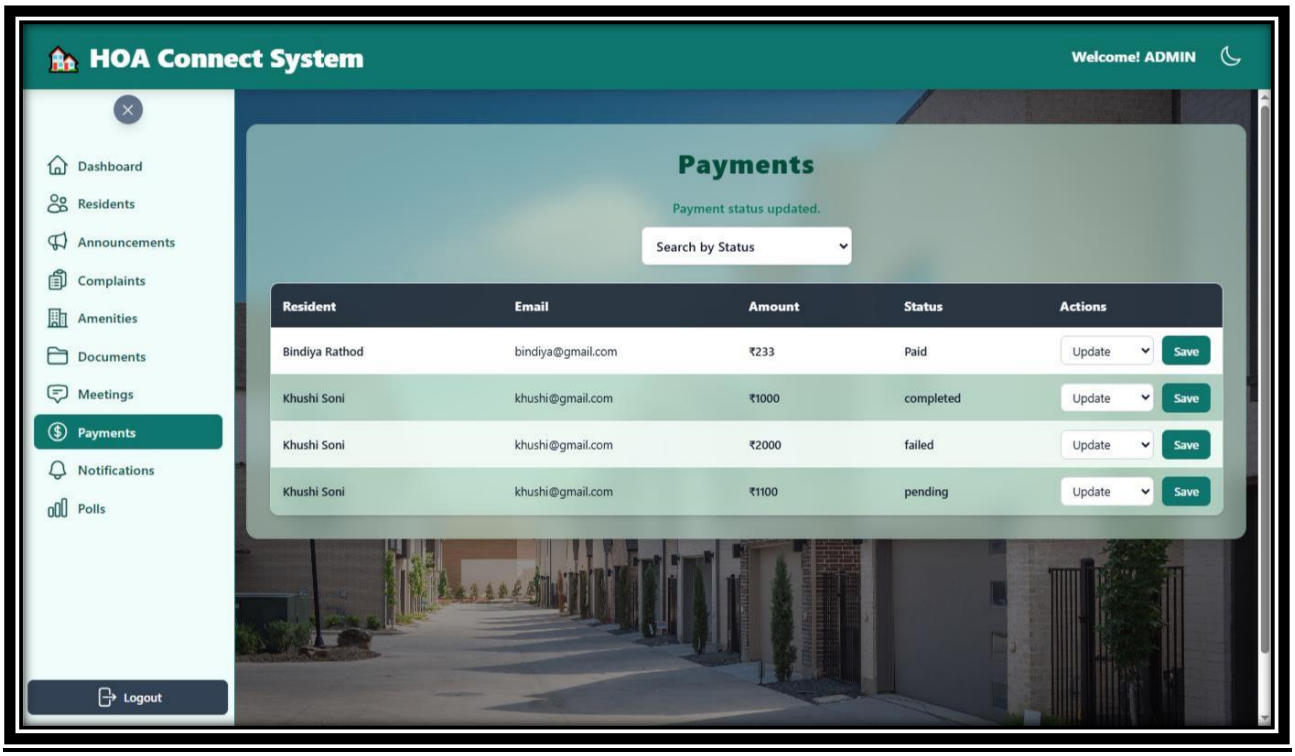
➢ Manage Documents:-



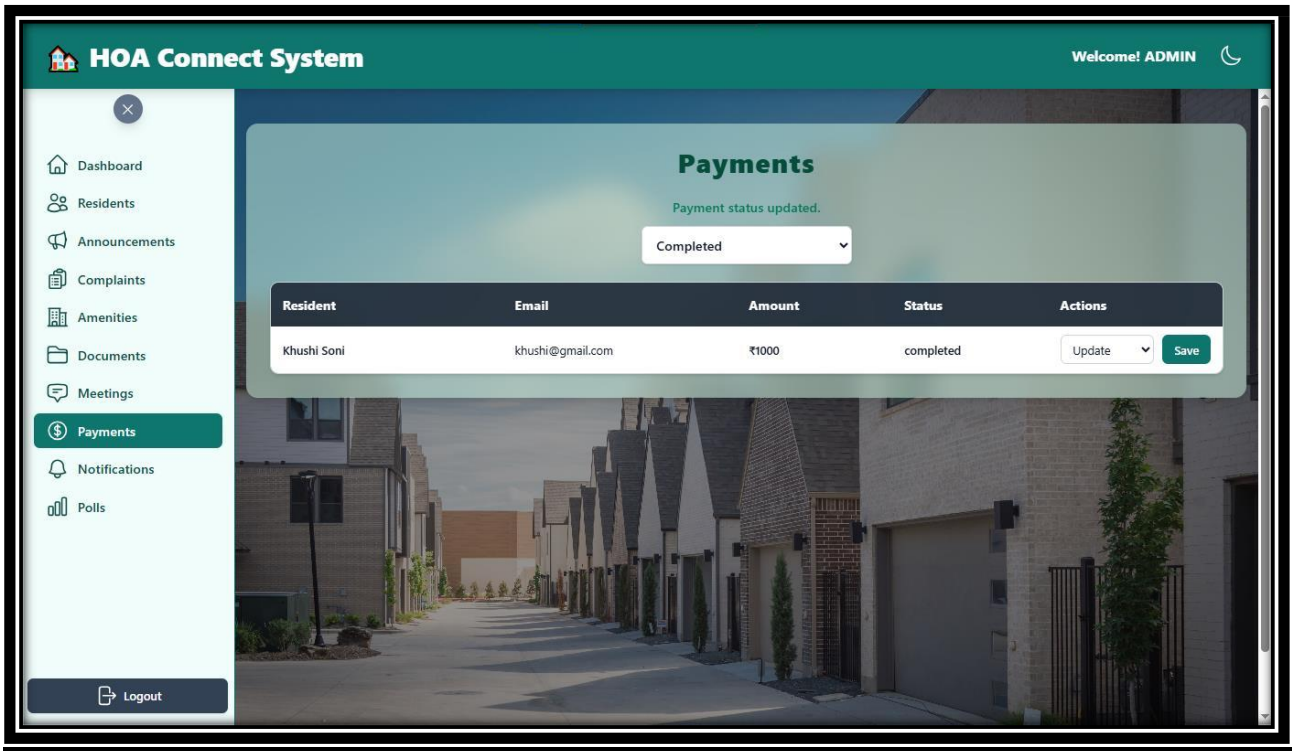
➢ Manage Meeting:-



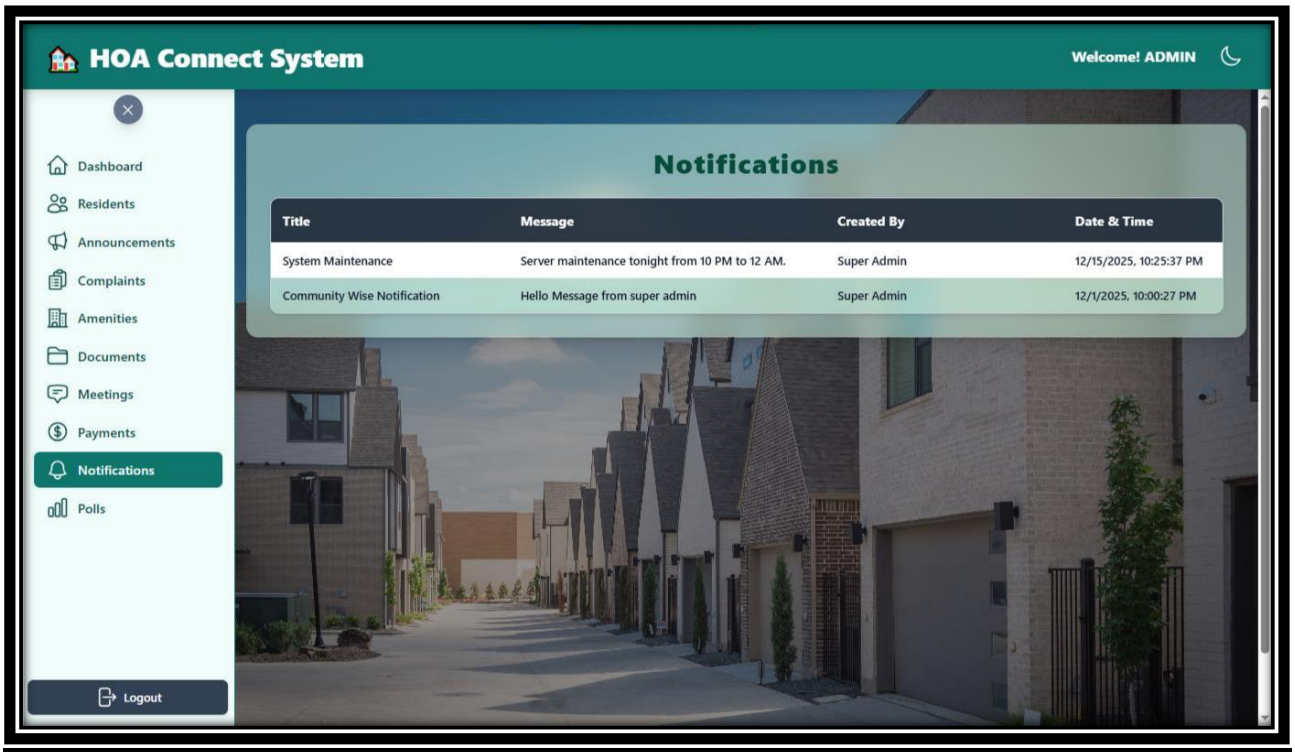
➢ Manage Payments:-



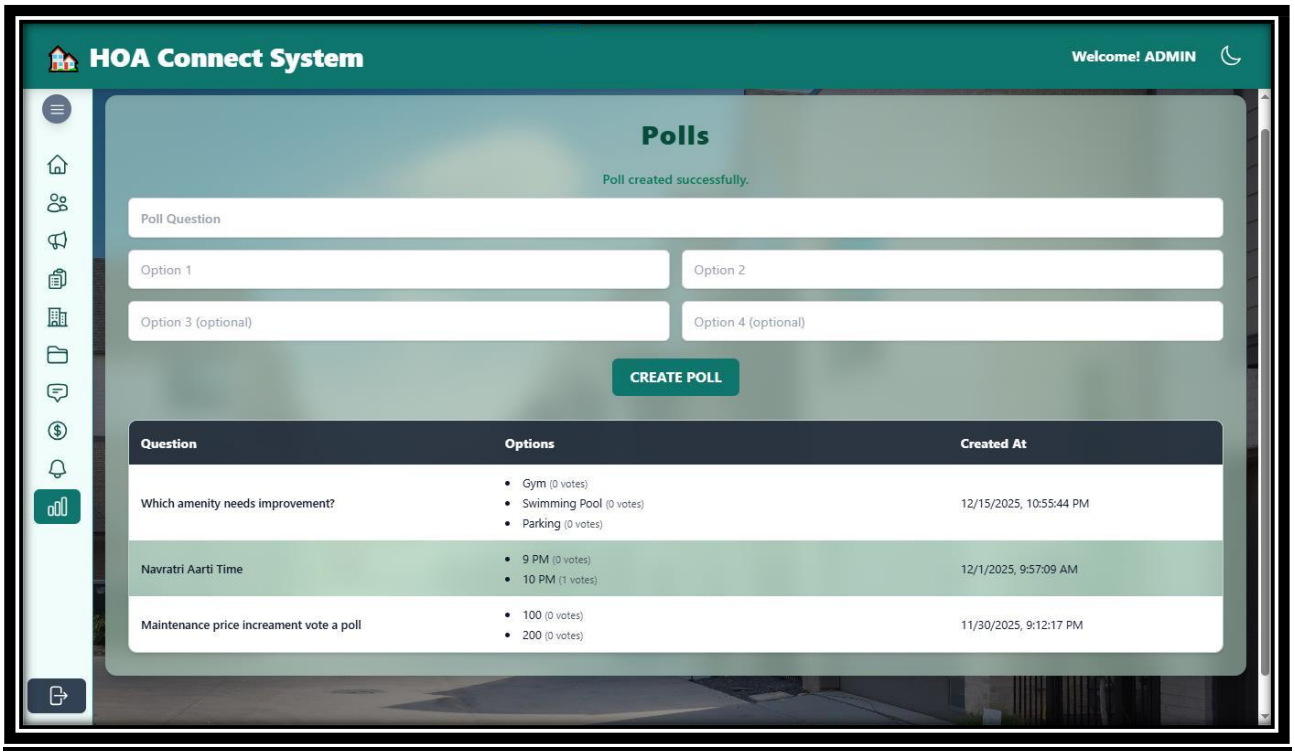
➢ Search Payment Status:-



➢ View Notifications:-

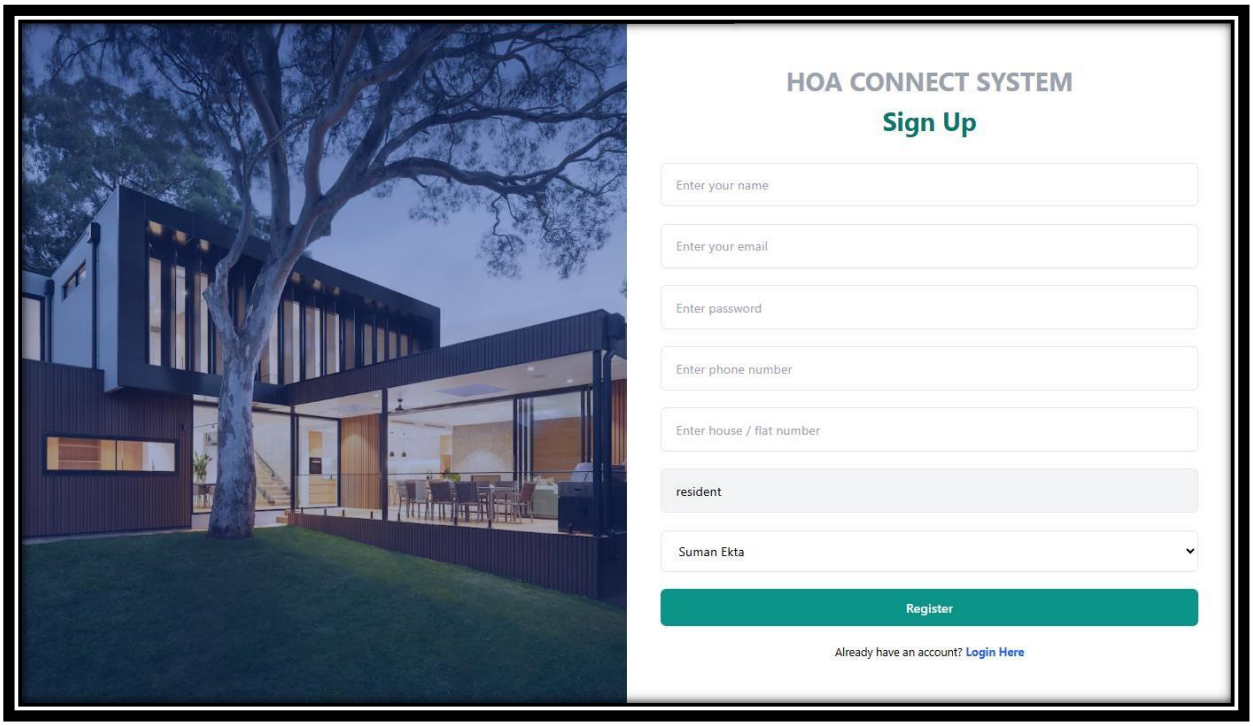


➢ Manage Polls:-

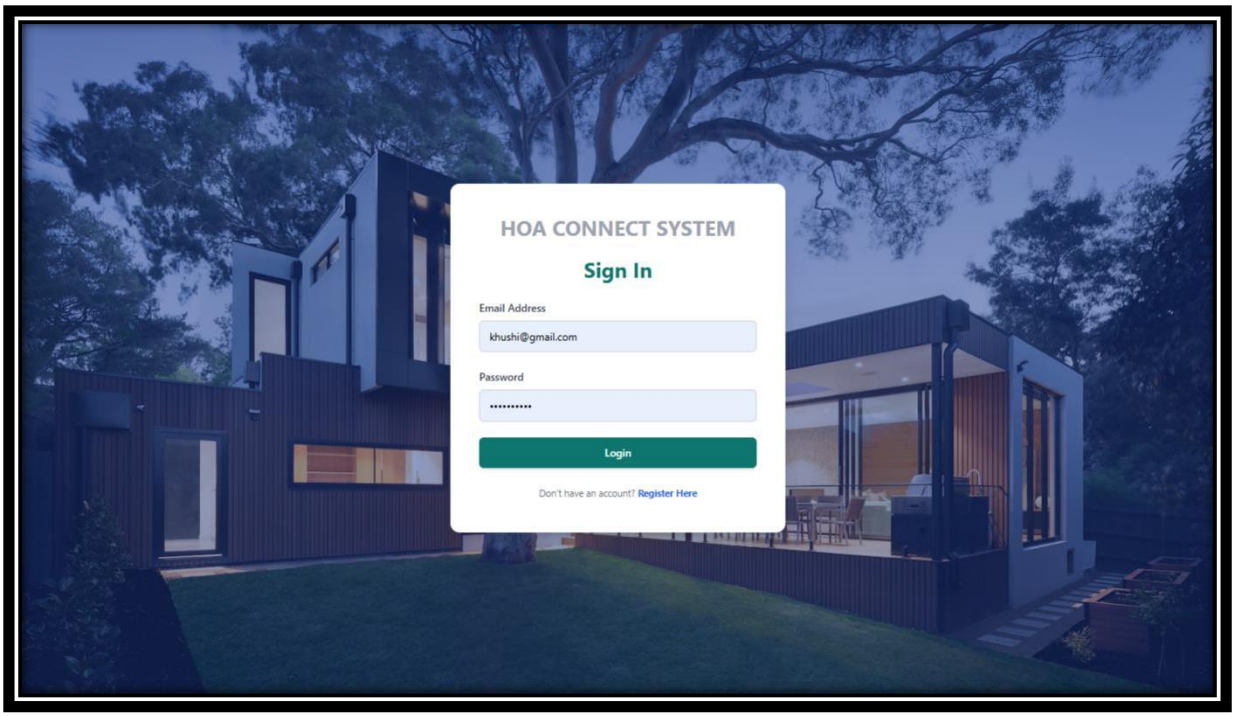


❖ Resident:

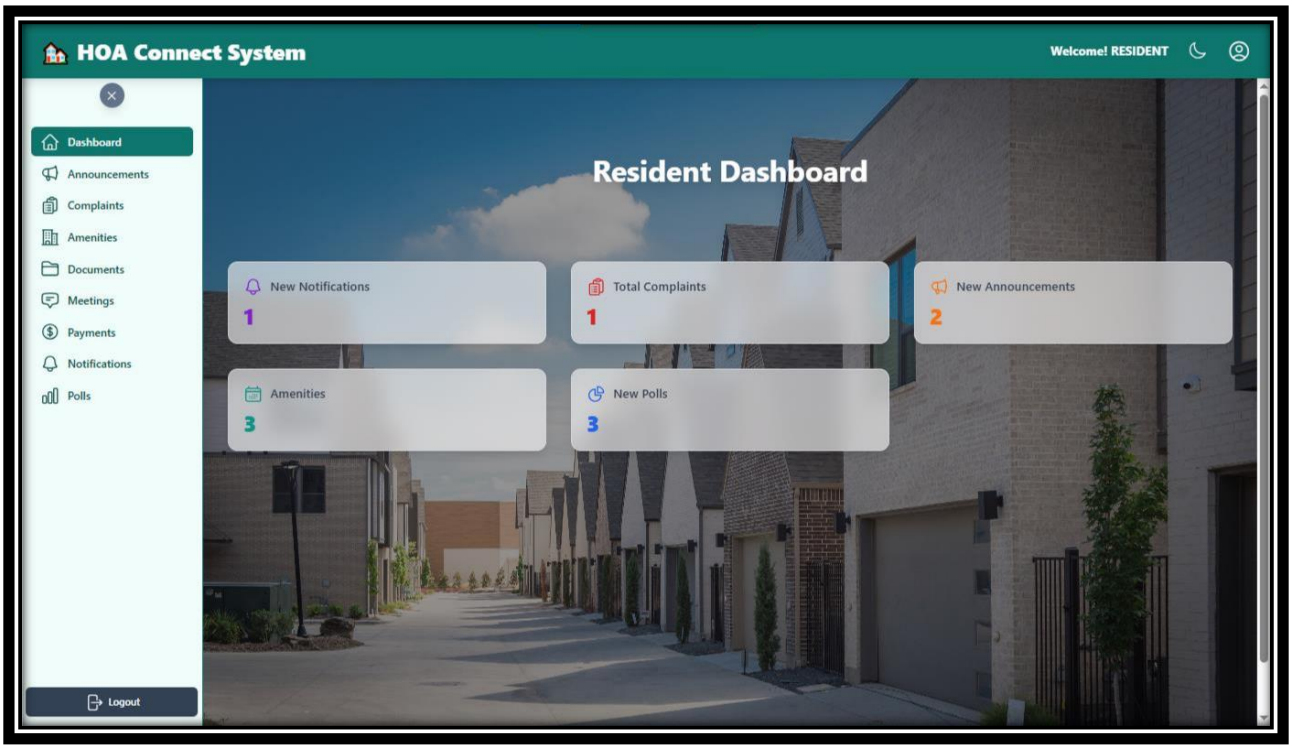
➢ Sign Up:-



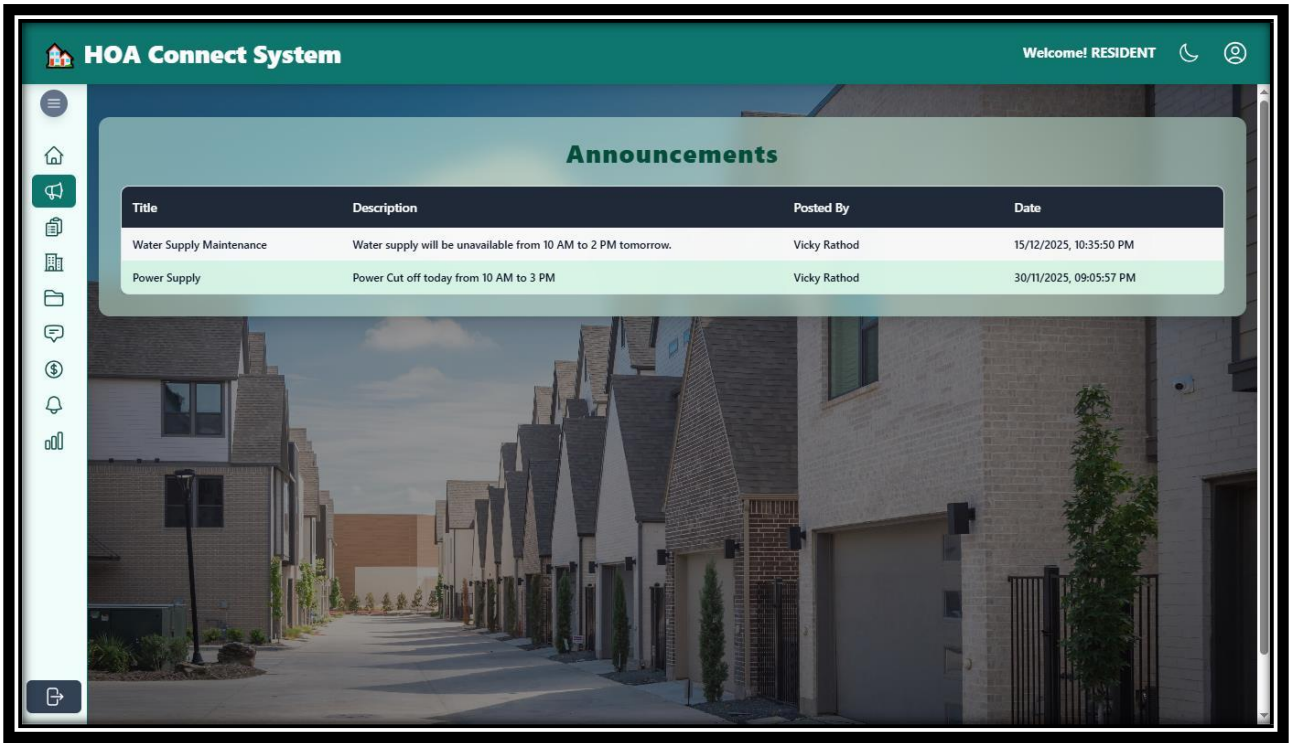
➢ Sign In:-



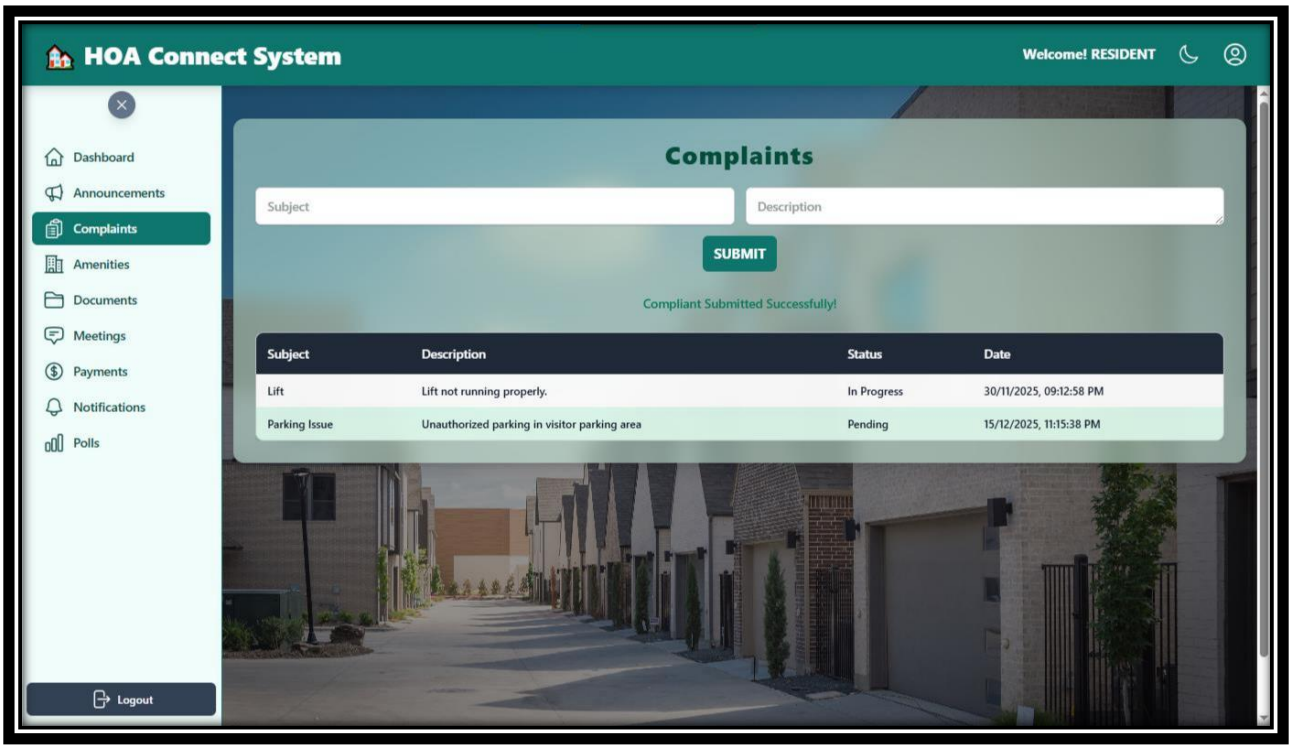
➢ Dashboard:-



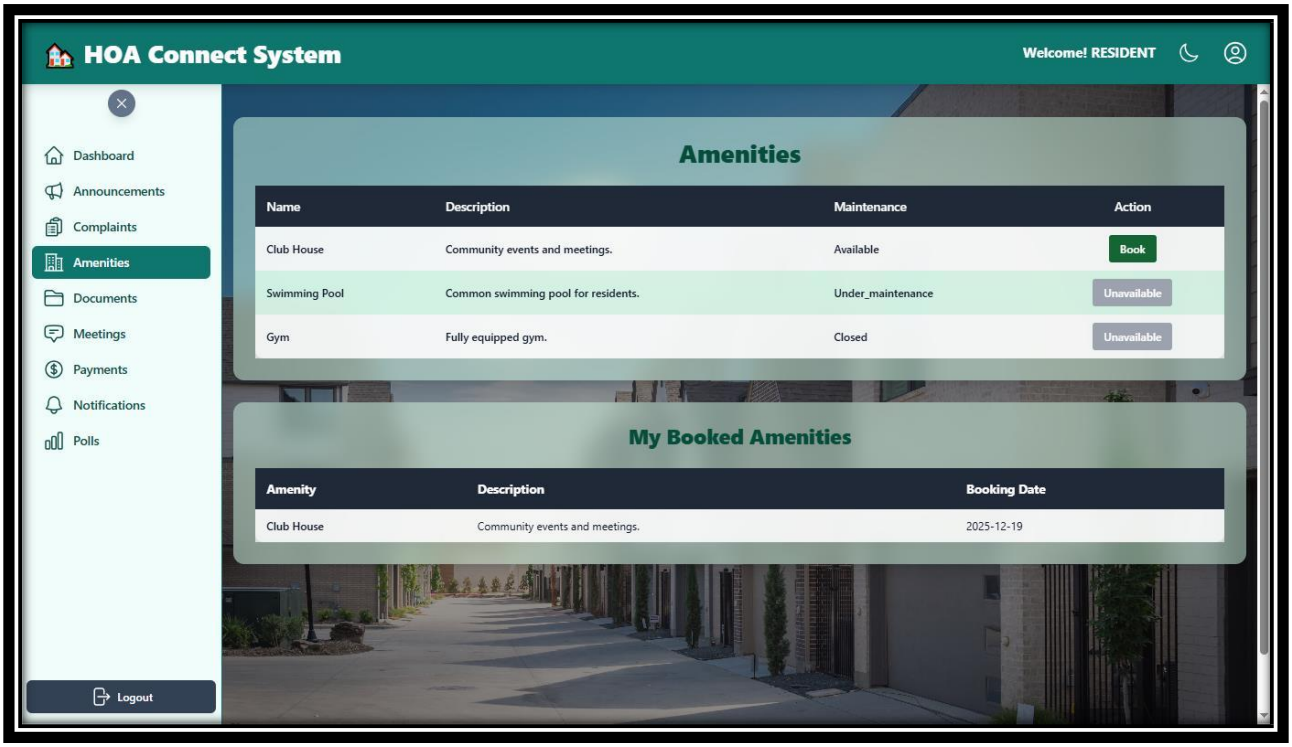
➢ View Announcements:-



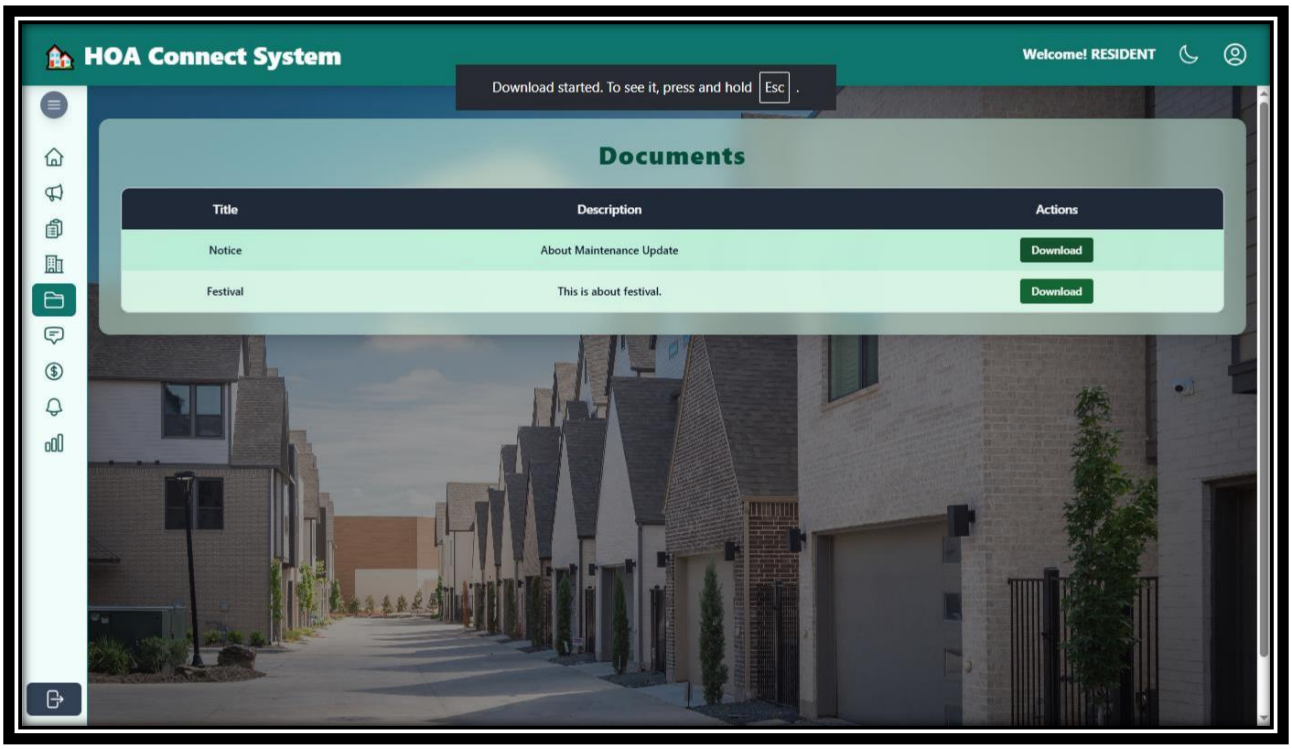
- **Add Complaint:-**



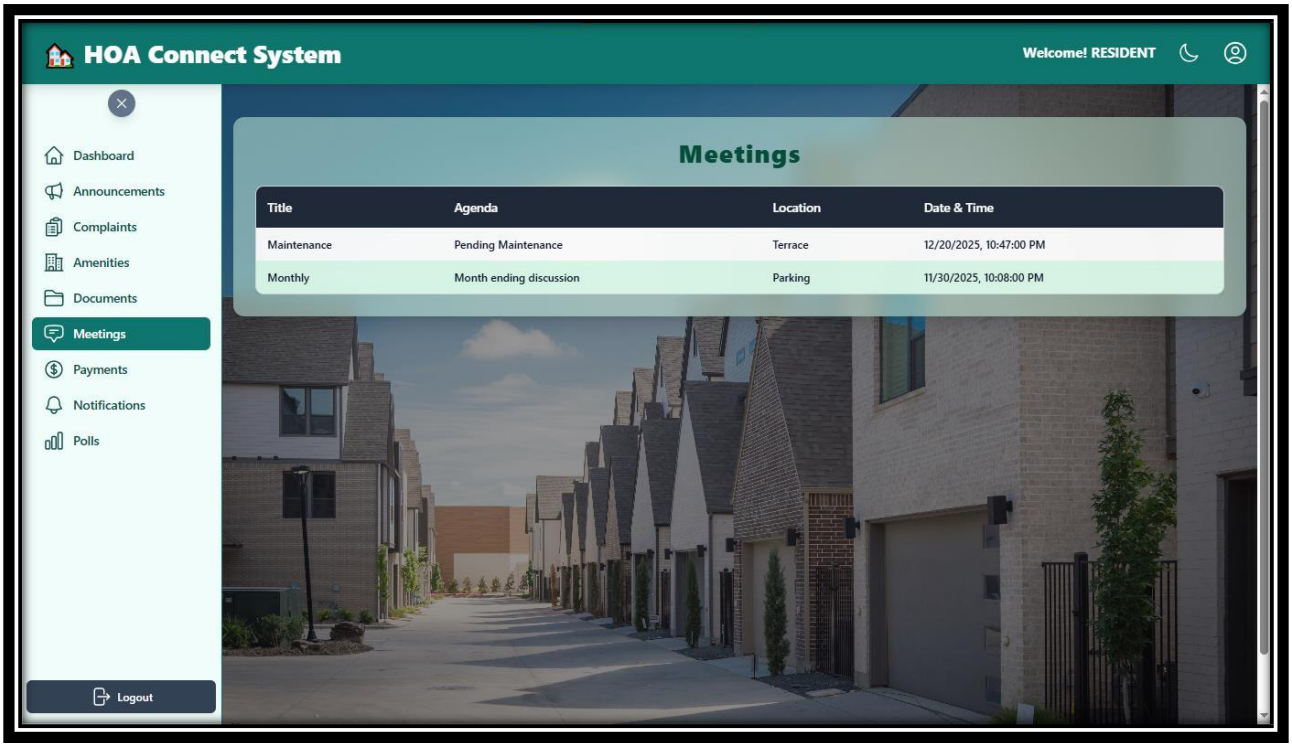
- **Book Amenities:-**



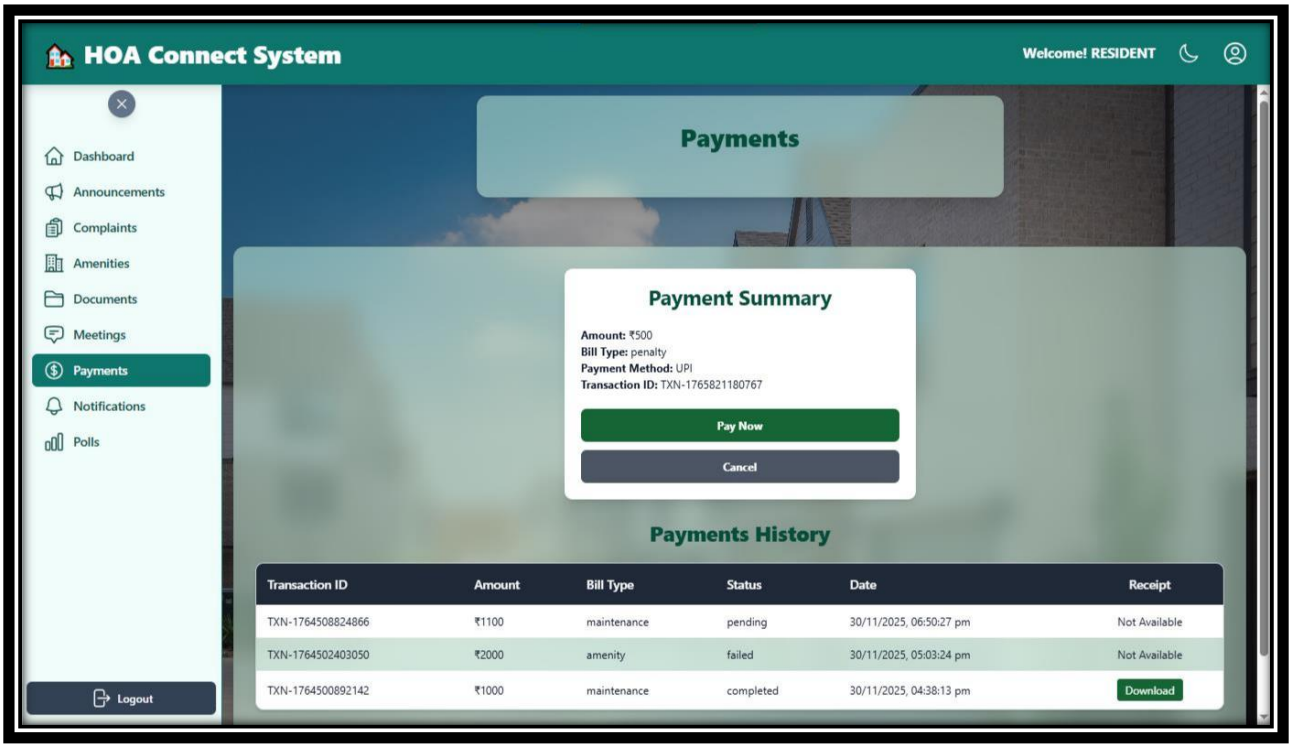
- **Documents Download:-**



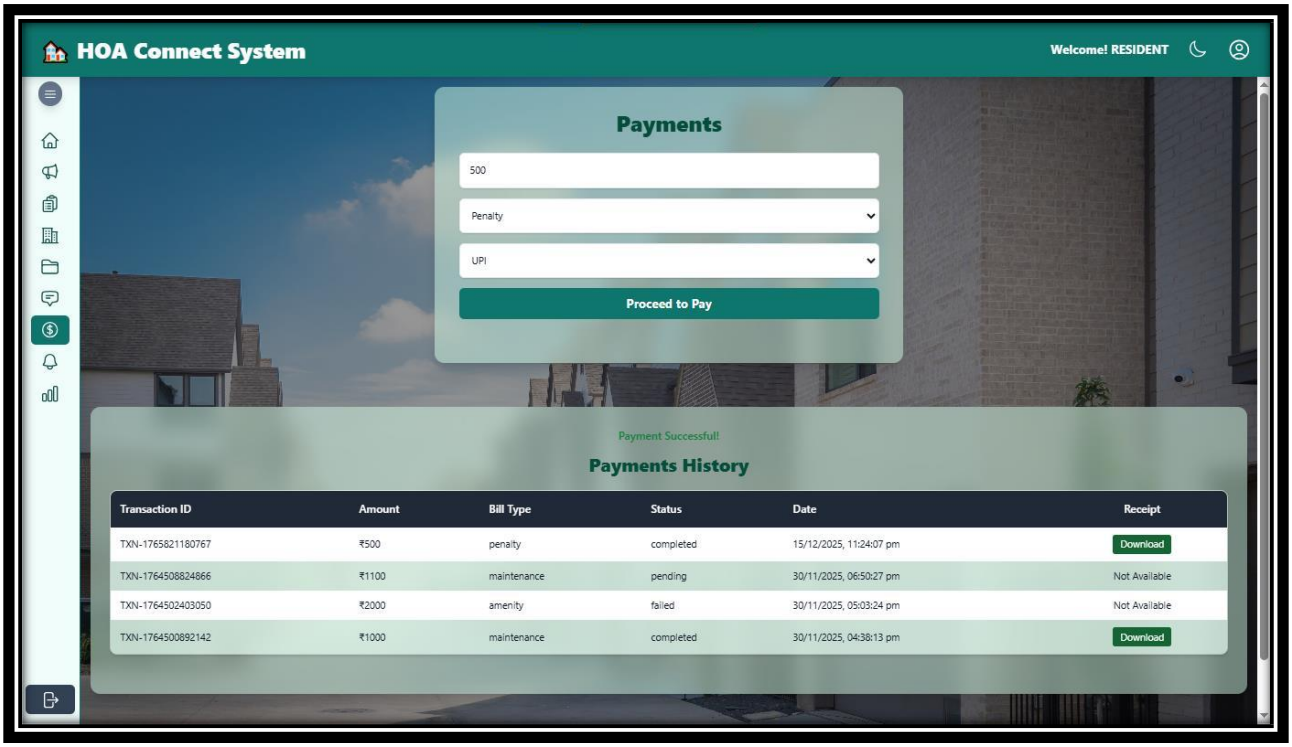
- **Show and Attend Meeting:-**



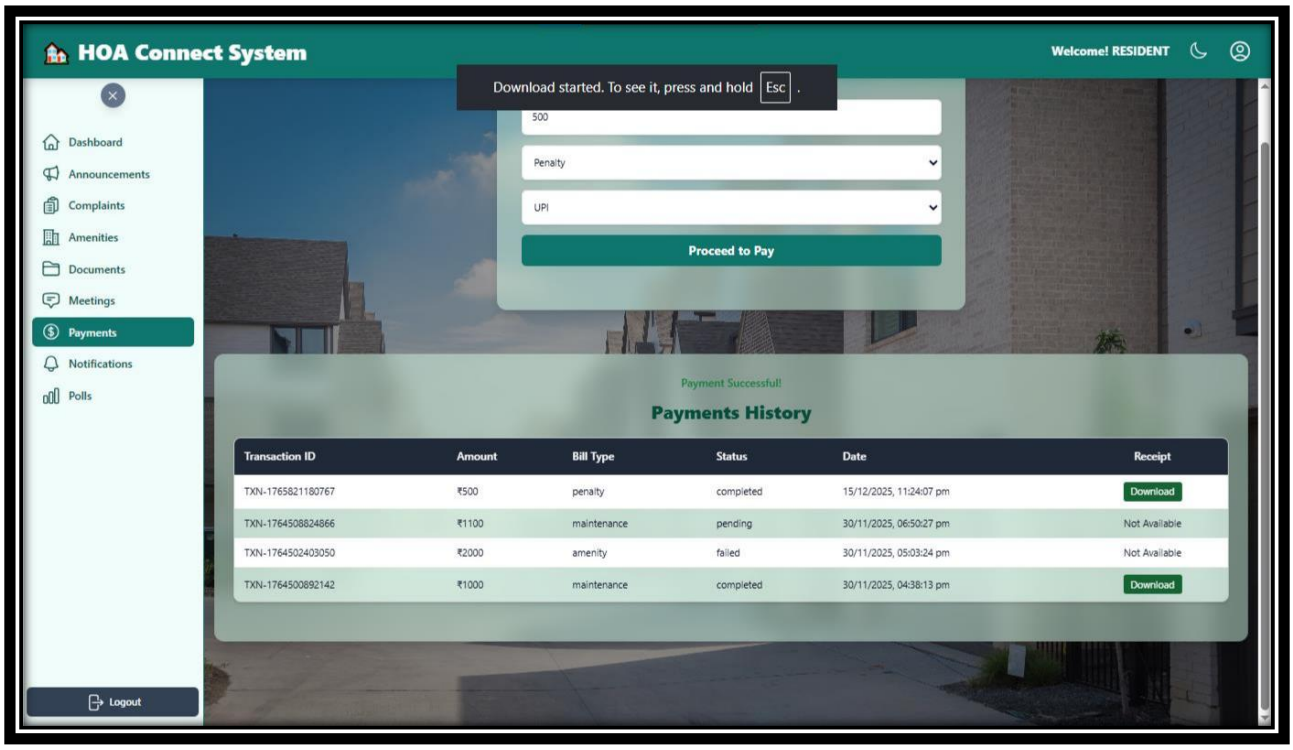
- **Make Payment:-**



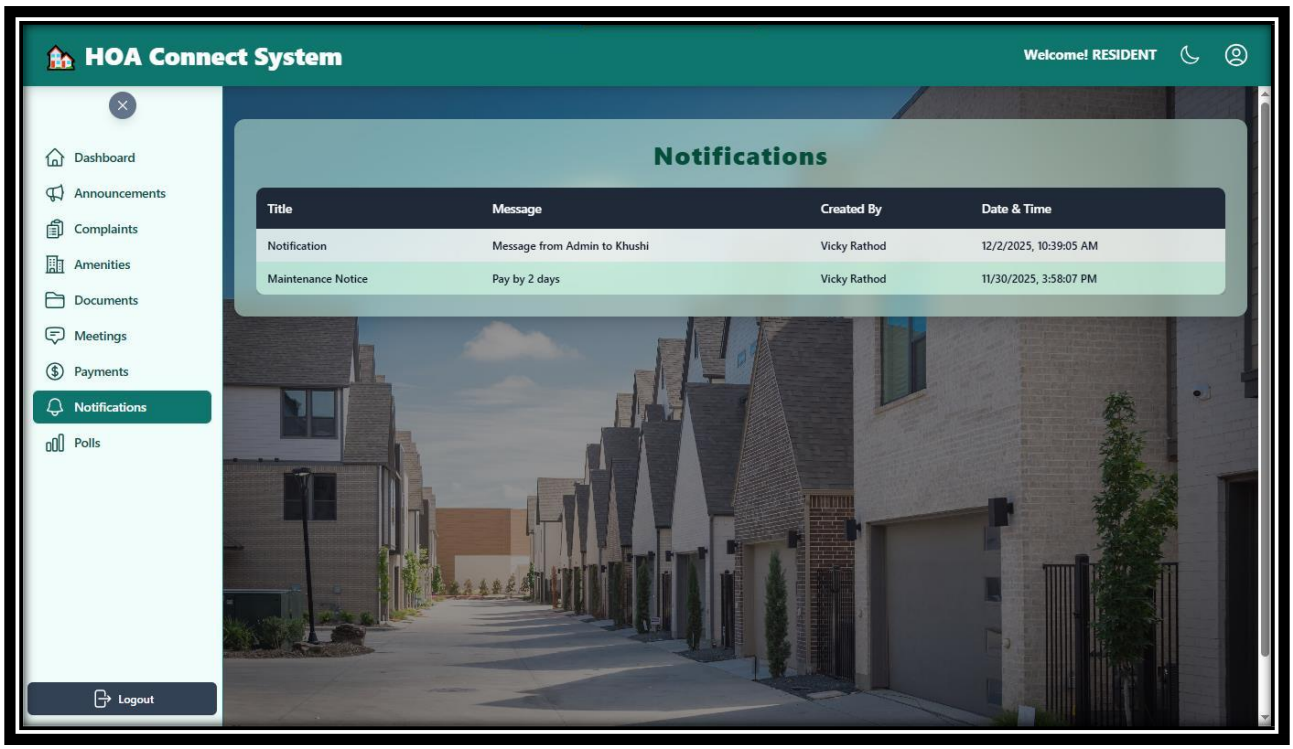
- **View Payment History:-**



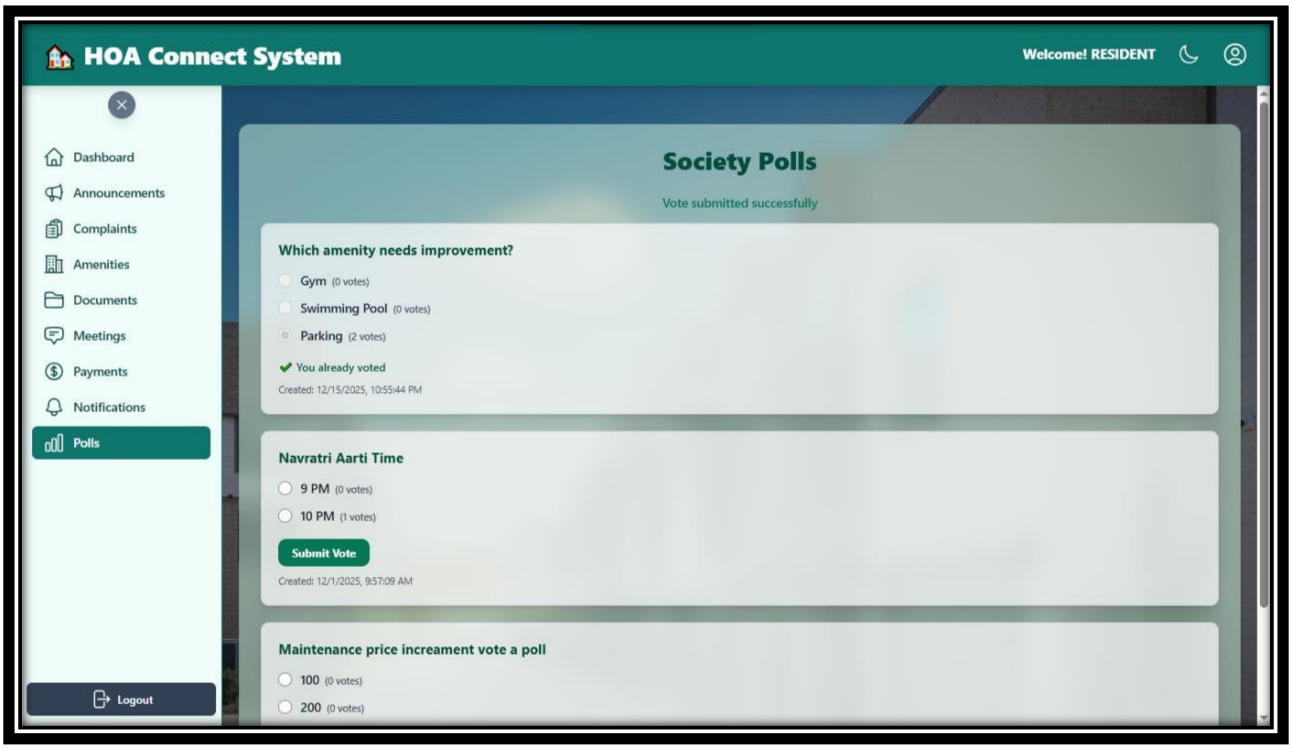
➢ Download Payment Reciept:-

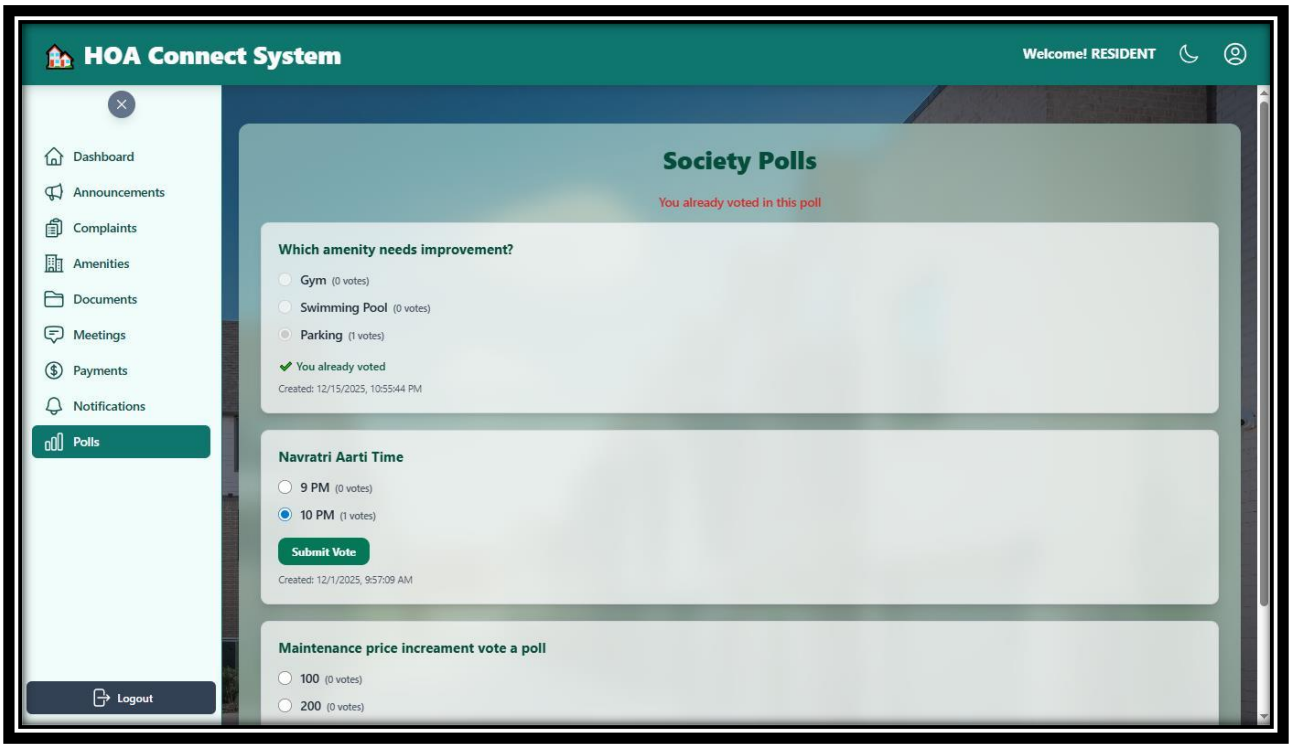


➢ Show User Notification:-



- **Vote Poll:-**


HOA Connect System
Welcome! RESIDENT
Dashboard
Announcements
Complaints
Amenities
Documents
Meetings
Payments
Notifications
Polls
Logout
Society Polls
Vote submitted successfully
Which amenity needs improvement?
Gym (0 votes)
Swimming Pool (0 votes)
Parking (2 votes)
You already voted
Created: 12/15/2025, 10:55:44 PM
Navratri Aarti Time
9 PM (0 votes)
10 PM (1 votes)
Submit Vote
Created: 12/1/2025, 9:57:09 AM
Maintenance price increament vote a poll
100 (0 votes)
200 (0 votes)


HOA Connect System
Welcome! RESIDENT
Dashboard
Announcements
Complaints
Amenities
Documents
Meetings
Payments
Notifications
Polls
Logout
Society Polls
You already voted in this poll
Which amenity needs improvement?
Gym (0 votes)
Swimming Pool (0 votes)
Parking (1 votes)
You already voted
Created: 12/15/2025, 10:55:44 PM
Navratri Aarti Time
9 PM (0 votes)
10 PM (1 votes)
Submit Vote
Created: 12/1/2025, 9:57:09 AM
Maintenance price increament vote a poll
100 (0 votes)
200 (0 votes)

6.3 Architecture Design

- A **3-Tier Application Architecture** is a modular client–server architecture that separates an application into three logical layers: **Presentation Tier**, **Application Tier**, and **Data Tier**. Each tier has a distinct responsibility and can be developed, maintained, and scaled independently.

❖ Presentation Tier:

- It handles user input such as login, payments, complaints, amenity bookings, and displays responses received from the application tier. The presentation tier communicates with the application tier through secure API calls.

❖ Application Tier:

- The application tier, also known as the business logic layer, contains the core logic of the HOA Connect system. This layer processes user requests, enforces business rules, validates data, and manages role-based access.
- It handles operations such as community management, admin replacement, complaint resolution, payment processing, meeting and poll management, and notification handling. The application tier communicates with both the presentation tier and the data tier.

❖ Data Tier:

- The data tier is responsible for storing, retrieving, and managing application data. This layer contains the database where all information related to users, communities, amenities, complaints, payments, meetings, polls, and documents is stored securely.

❖ Example of a 3-Tier Architecture in HOA Connect :

- **Presentation Tier:** A web application developed using HTML, CSS, JavaScript, and React.js to provide interactive dashboards and user interfaces.
- **Application Tier:** A Node.js and Express.js based backend that handles business logic, authentication, authorization, and API communication.
- **Data Tier:** A MongoDB database used to store and retrieve application data efficiently and securely.

7. TESTING

7.1 Unit Testing

Unit testing ensures that individual components of the HOA Connect system function correctly in isolation. It is performed on small, testable units such as functions, APIs, services, or individual modules.

Key Aspects for HOA Connect:

- **User Authentication Tests** – Verify login, registration, JWT token generation, and role-based access.
- **Complaint Management Tests** – Ensure correct creation, status updates, and resolution handling.
- **Payment Processing Tests** – Validate payment records, transaction status, and history generation.
- **Amenity Booking Tests** – Check booking availability, date validation, and booking limits.
- **Community Management Tests** – Verify creation and assignment of communities and admins.
- **Exception Handling Tests** – Ensure proper error handling for invalid inputs, unauthorized access, and server errors.

7.2 Integration Testing

Integration testing evaluates how different modules of the HOA Connect system work together to ensure seamless functionality between the **frontend (React.js)** and backend **(Node.js/Express).**

Key Integration Areas for HOA Connect:

- **Authentication Flow** – Validate login, authorization, and session handling across roles.
- **Community-wise Data Isolation** – Ensure admins and residents access only their assigned community data.
- **Payment Integration Workflow** – Verify data flow between frontend payment actions and backend payment storage.
- **Amenity Booking Workflow** – Ensure proper interaction between amenities, users, and booking records.
- **Database Connectivity** – Confirm accurate data storage and retrieval from MongoDB.
- **Notification System** – Ensure residents receive real-time or scheduled notifications for announcements and updates.

8. FUTURE ENHANCEMENT

- **Advanced Security and Authentication**
  Future versions of HOA Connect can implement enhanced security features such as multi-factor authentication (MFA), refresh tokens, and device-based login verification to further strengthen data protection.

- **Real-Time Notifications and Updates**
  Integrating real-time communication using Web-Sockets or similar technologies can enable instant notifications for announcements, complaints updates, meeting reminders, and payment confirmations.

- **Mobile Application Support**
  Developing a mobile application for Android and iOS will improve accessibility and allow residents and admins to manage community activities on the go.

- **Enhanced Analytics and Reporting**
  Future enhancements may include detailed dashboards with visual graphs for payments, complaints, amenity usage, and community engagement, enabling better decision-making.

- **Additional Community Modules**
  New modules such as parking management, vendor services, visitor management, and maintenance tracking can be added to expand system functionality.

9. GLOSSARY

- **API :** Enables communication between frontend (React.js) and backend (Node.js/Express).
- **Authentication :** Verifies user identity for secure system access..
- **Authorization** : Controls access based on user roles (Super Admin, HOA Admin, Resident).
- **Backend** : Server-side logic handling requests, validations, and business rules.
- **Community Management** : Managing residential communities and their assigned admins.
- **Complaint Management** : System for raising, tracking, and resolving resident complaints.
- **CRUD Operations** : Create, Read, Update, and Delete operations on data.
- **Database** : MongoDB used for storing user, community, and transaction data.
- **Frontend** : User interface built using React.js.
- **Integration Testing** : Ensures different modules work together correctly.
- **JWT (JSON Web Token)** : Used for secure authentication and authorization.
- **Payment Management** : Handles maintenance payments and transaction history.
- **Risk Analysis** : Identification and evaluation of project risks.
- **Unit Testing** : Testing individual modules independently.

10. REFERENCE

- www.chatgpt.com
- www.perplexityAI.com
- https://vite.dev
- https://nodejs.org