INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT (IACSD), AKURDI, PUNE

Documentation On
(Group no. 98)

Society Management System (SMS.com)

PG-DAC March 2023

Group no: 98

UNDER THE GUIDANCE OF:
Mrs. Monika Sindhikar
Mr. Harshal Waghchaure
Faculty Member

SUBMITTED BY:
Kunal (233165)
Siddhant Malik (233215)

ACKNOWLEDGEMENT

We would like to extend our heartfelt gratitude and acknowledgment to the individuals who played a pivotal role in the successful development and completion of the Society Management System project.

First and foremost, we express our gratitude to Mr. Rohit Purnaik, the Center Coordinator, for his continuous support and guidance throughout the project's duration. His insights and motivation have been instrumental in steering us in the right direction.

Our appreciation also goes to Mr. Narendra Pawar, the Course Coordinator, for his valuable inputs and encouragement that pushed us to excel in our project. His expertise and guidance have been invaluable in shaping the project's outcome.

We would also like to extend our thanks to our Project Guide, Mr. Harshal Waghchaure, for his unwavering support and mentorship. His expert advice and feedback have greatly enriched the project and our learning experience.

Furthermore, we extend our gratitude to Miss Monika for her contributions to the project. Her insights and dedication have added significant value to our efforts.

Last but not least, a special mention goes to our friends Abhishek, Ayush, Anant, Prateek for their remarkable contributions and

collaboration. Their dedication and teamwork have been crucial to the project's success.

Together, this exceptional team effort has culminated in the development of the Society Management System project. Each individual's unique skills and commitment have combined to create a robust and functional system that addresses the needs of our community.

Once again, we express our heartfelt gratitude to everyone involved for their dedication, support, and contributions to this project. Your efforts have truly made a difference.

Thank you.

Sincerely,

Kunal

Siddhant Malik

CERTIFICATE

This is to certify that the project work under the title 'Society Management System' is done by Kunal and Siddhant Malik in partial fulfillment of the requirement for award of Diploma in Advanced Computing Course.

Mr. Harshal Waghchaure

Mrs. Monika — Project Guide

Mr. Narendra — Course Coordinator

Date: 30/07/2023

INDEX

- INTRODUCTION TO PROJECT
- OBJECTIVE:
- FUNCTIONAL REQUIREMENTS
- NON-FUNCTIONAL REQUIREMENTS
- DIAGRAMS
  - USE CASE DIAGRAM
  - CLASS DIAGRAM
  - ADMIN ACITIVITY DIAGRAM
  - OWNER ACITIVITY DIAGRAM
  - SECURITY GUARD ACITIVITY DIAGRAM
  - 0 LEVEL DFD
  - 1 LEVEL DFD FOR ADMIN
  - 1 LEVEL DFD FOR OWNER
  - 1 LEVEL DFD FOR SECURITY GUARD
  - ER DIAGRAM
- SCREENSHOTS
- DESIGN
- CODING STANDARDS IMPLEMENTED
- CONCLUSION OF THE PROJECT SOCIETY MANAGEMENT SYSTEM
- FUTURE SCOPE OF THE PROJECT
- LIMITATIONS OF THE PROJECT
- REFERENCES

1. INTRODUCTION TO PROJECT

The purpose of this document is to outline the requirements for the development of Society Management System.

Society Management System is a web-based application which can manage the different functionalities of a society online.
This app allows the security guards of the society to store details of visitors in the application.
Allows the house owners to book various facilities of the society such as party halls and meeting halls and see the latest news and upcoming events in the society.
The admin can add, update the details of house owners, the news and events which can be viewed by all the members of the society

2.OBJECTIVE

Key features of a Vehicle Society Management System may include:

●**Visitor details:** The Society Management software empowers security guards to efficiently record visitor details. With a user-friendly interface, guards can swiftly register visitors by capturing essential information such as names, contact details, purpose of visit, and the resident they are visiting.

●**User (house owner) details:** Within the Society Management System, administrators enjoy the convenience of swiftly adding and modifying house owner details. This functionality empowers administrators to maintain an accurate and updated database of residents. They can effortlessly enter new house owner information, such as contact details and occupancy status, and promptly update any changes. This feature ensures that resident records remain current and facilitates effective communication within the community. As a result, administrators can efficiently manage resident information, enhancing the overall efficiency and cohesion of the community management process.

● **News and events:** The Society Management System offers administrators a simple and efficient way to manage news and events. Administrators can easily add new announcements and upcoming events, ensuring residents stay informed about community activities. Additionally, they have the flexibility to modify or update existing news and event details as needed. This feature facilitates seamless communication, enabling administrators to keep residents engaged and up-to-date on the latest happenings within the community.

Booking facilities: In the Society Management System, administrators have the convenience of effortlessly managing party hall and meeting hall facilities. They can promptly add new bookings for these spaces, detailing the event date, time, and purpose. Additionally, administrators can easily modify or update existing bookings to accommodate changes in scheduling. This feature streamlines the reservation process, enabling efficient utilization of community spaces and enhancing resident satisfaction. Overall, administrators can effectively oversee and optimize the availability of party hall and meeting hall facilities, contributing to a well-organized and accommodating community environment.

Requirements

3. FUNCTIONAL REQUIREMENTS

- **3.1 Security Guard Functionality**

  - Record visitor details, including name, contact information, purpose of visit, and check-in/check-out times.

- **3.2 House Owners and Tenants Functionality**

  - Book facilities (party halls, guest houses) based on availability.

  - Access and view upcoming events and news.

- **3.3 Admin Functionality**

  - Manage news and events by adding, updating, and deleting entries.

  - Monitor user activities and manage user accounts.

- **3.4 Performance Requirements**

  - The system should respond to user interactions within 2 seconds.

  - Concurrent user capacity should be at least 100 users.

- **3.5 Design Constraints**

  - The system design should be responsive and mobile-friendly.

- **3.6.1 Usability**

  The user interface should be easy to navigate and understand.

- **3.6.2 Reliability**

  The system should operate without errors and handle user data securely.

- **3.6.3 Security**

  User data should be encrypted and stored securely. User authentication and authorization should be implemented.

- **3.6.4 Availability**

  The system should have at least 99% uptime.

- **3.6.5 Scalability**

  The system should be scalable to accommodate increased user loads.

- **3.6.6 Maintainability**

  The system should be easy to maintain, and updates should not disrupt user access.

- **3.6.7 Portability**

  The system should work across different web browsers and devices.

4. NON FUNCTIONAL REQUIREMENTS

- **4.1 Usability**

  The user interface should be easy to navigate and understand.

- **4.2 Reliability**

  The system should operate without errors and handle user data securely.

- **4.3 Security**

  User data should be encrypted and stored securely. User authentication and authorization should be implemented.

- **4.4 Availability**

  The system should have at least 99% uptime.

- **4.5 Scalability**

  The system should be scalable to accommodate increased user loads.

- **4.6 Maintainability**

  The system should be easy to maintain, and updates should not disrupt user access.

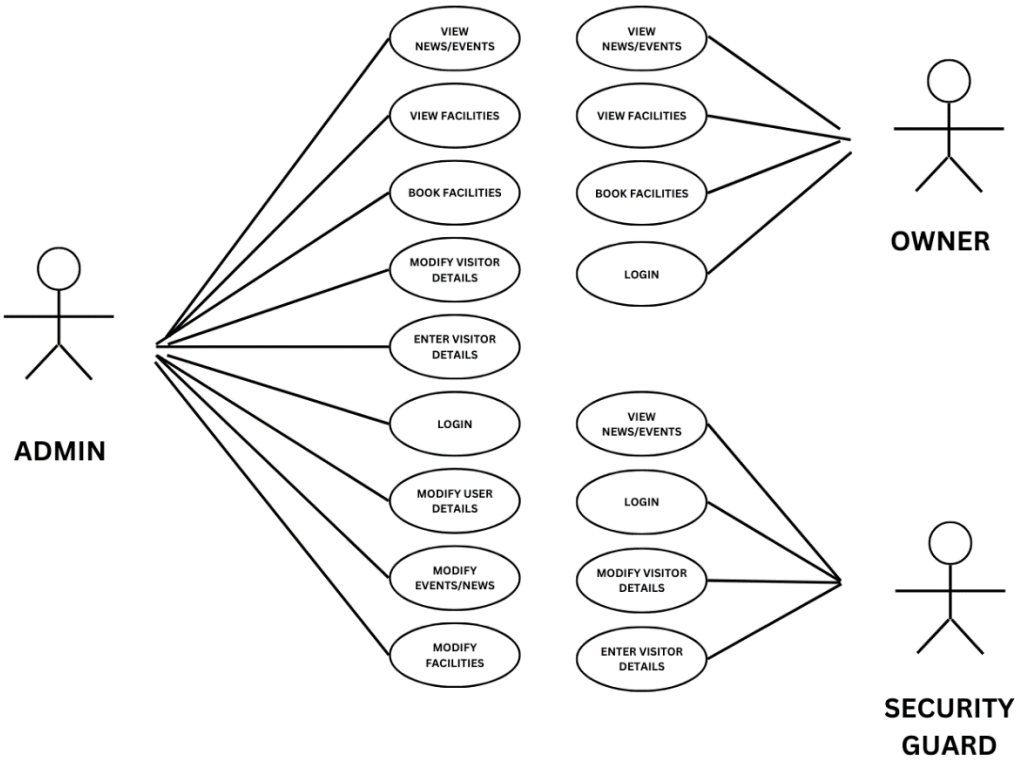
- **4.7 Portability**

  The system should work across different web browsers and devices.

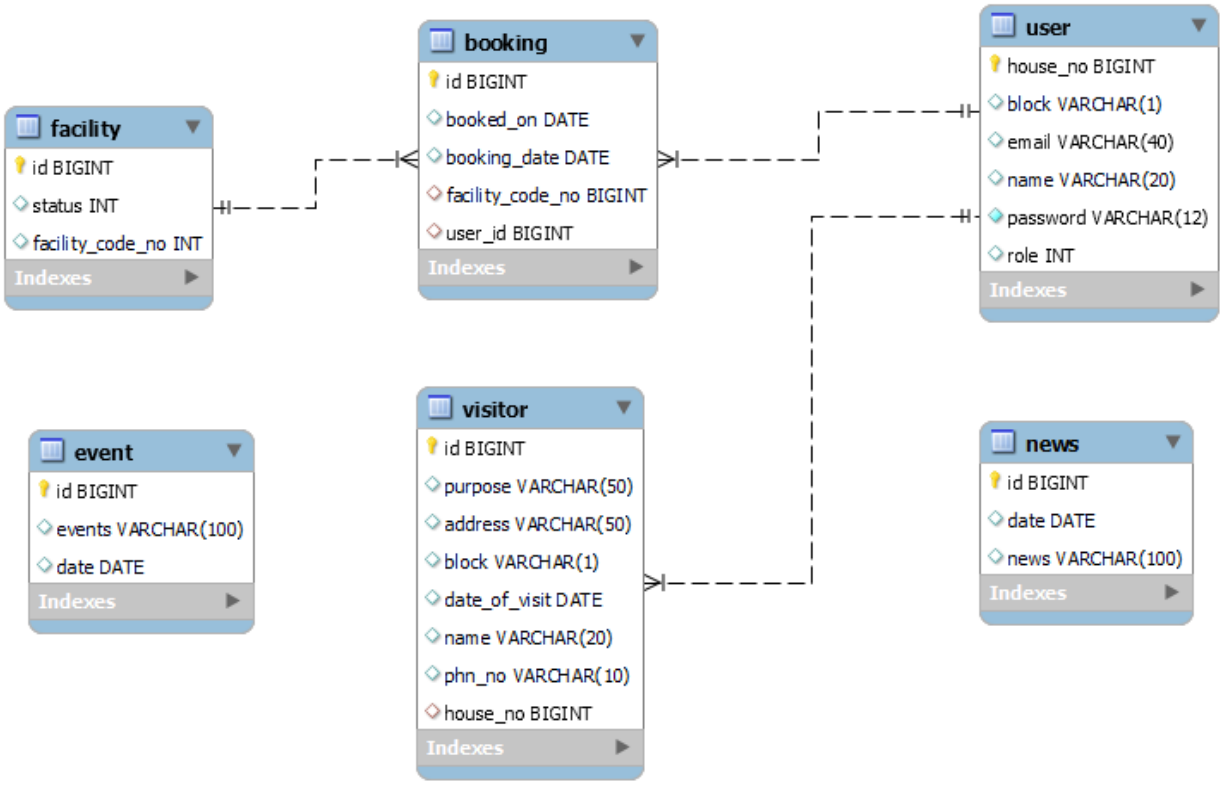
5. DIAGRAMS

Appendix A -

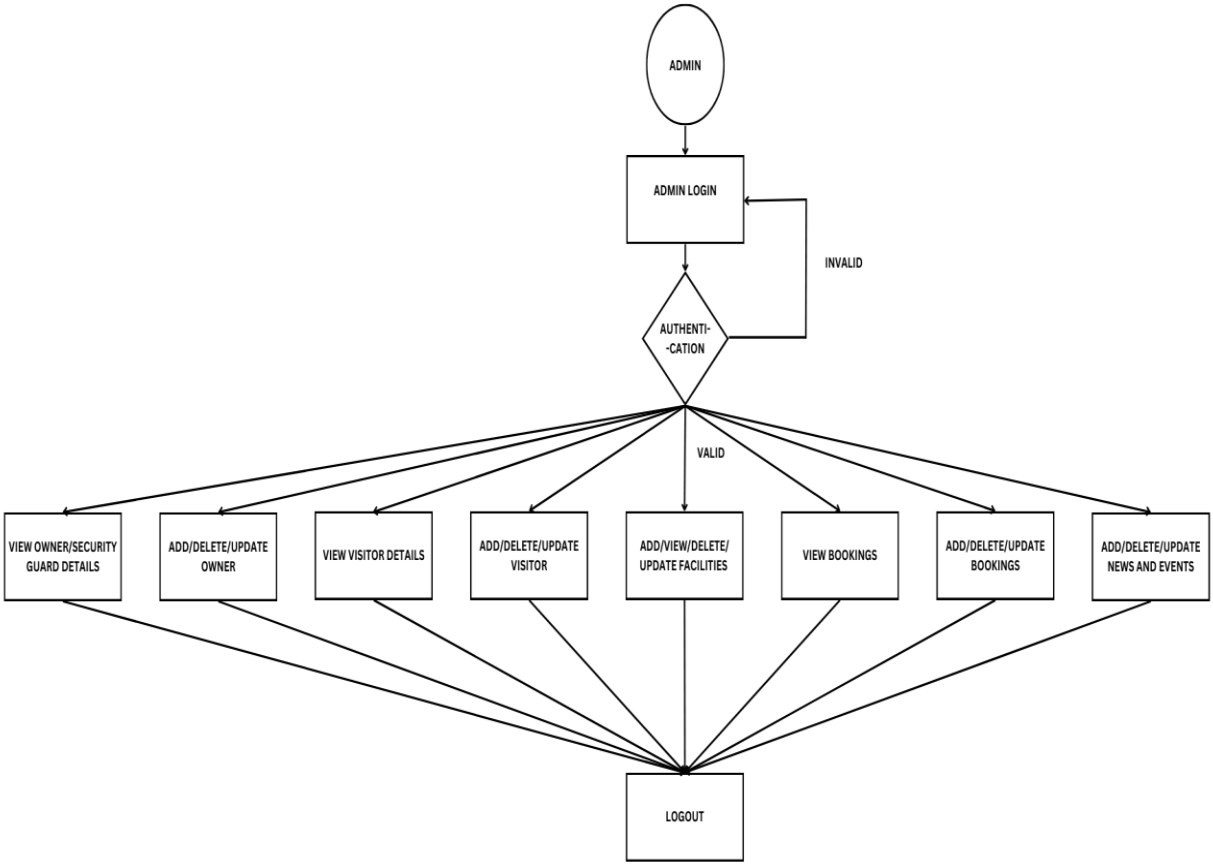
5.1 USE CASE DIAGRAM



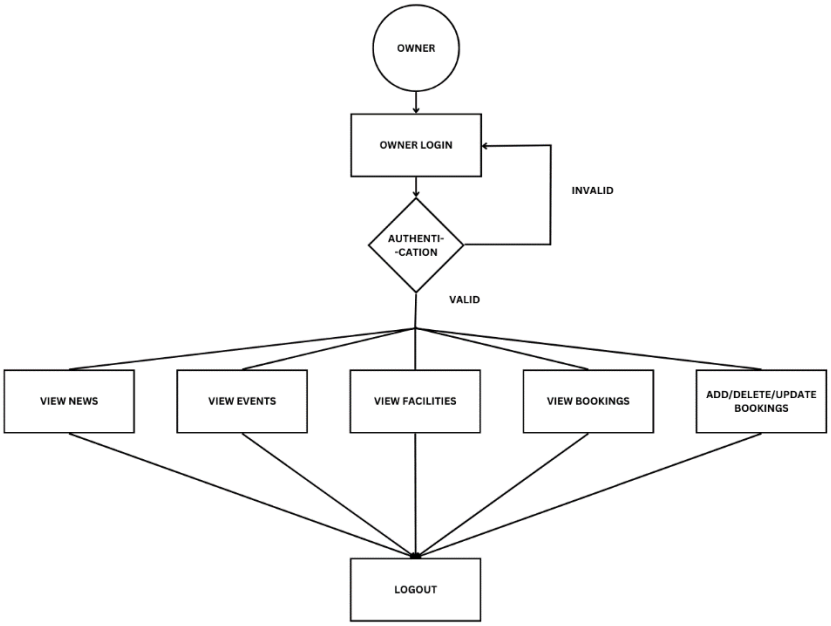
5.2 CLASS DIAGRAM



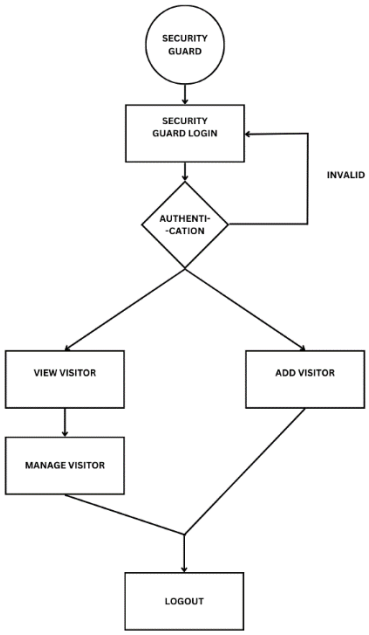
5.3 ADMIN ACITIVITY DIAGRAM



5.4 OWNER ACITIVITY DIAGRAM

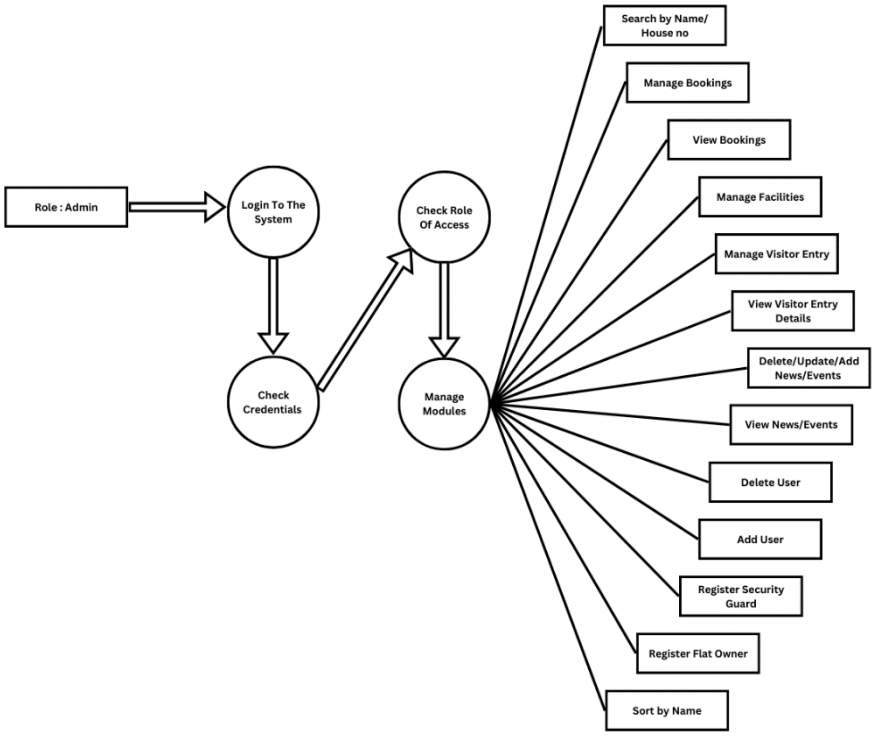


5.5 SECURITY GUARD ACITIVITY DIAGRAM

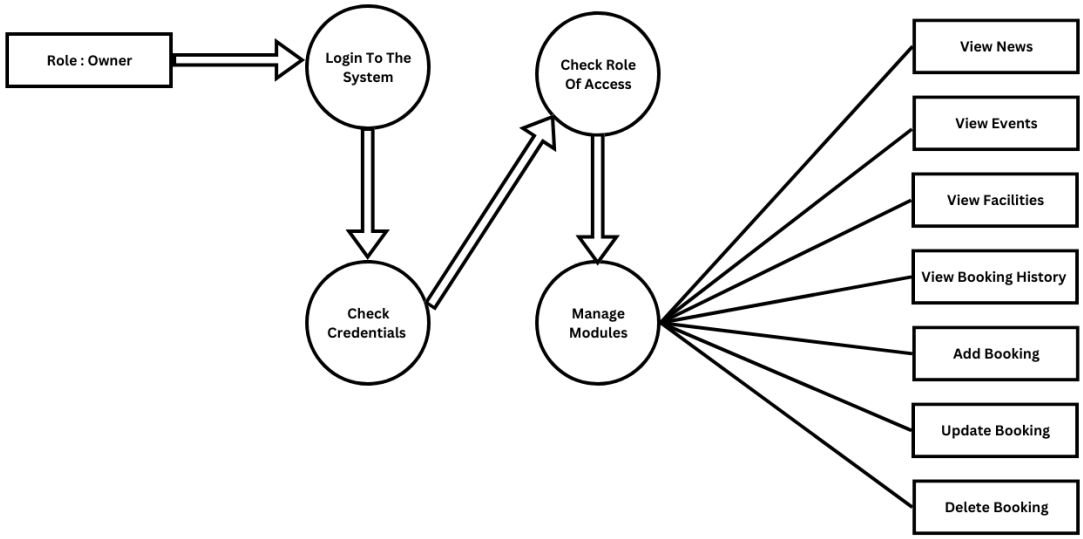


5.6 0 LEVEL DFD

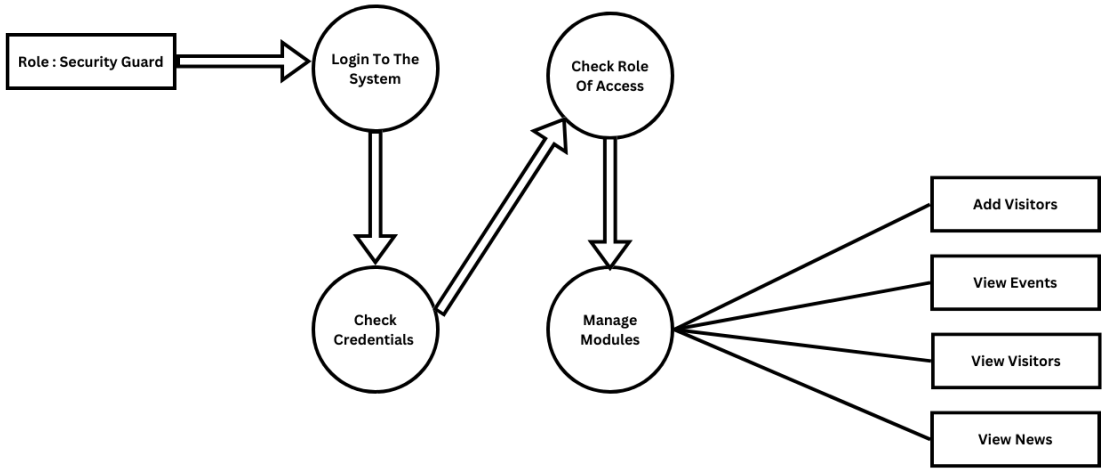
5.7 1 LEVEL DFD FOR ADMIN



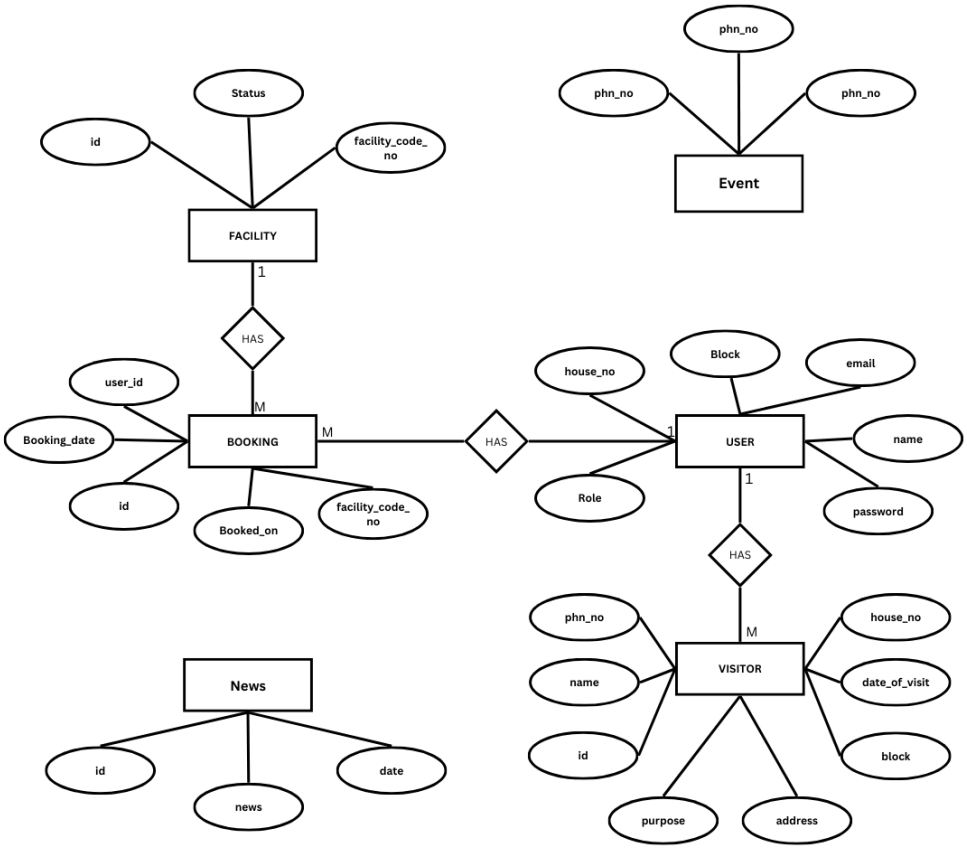
5.8 1 LEVEL DFD FOR OWNER



5.9 1 LEVEL DFD FOR SECURITY GUARD

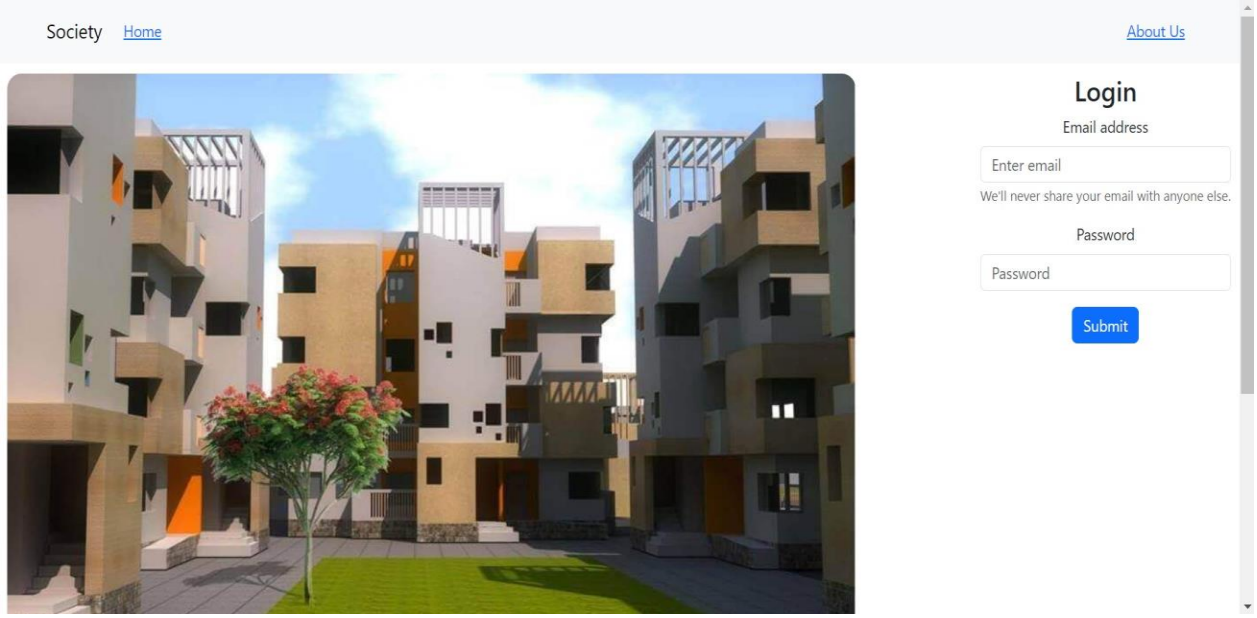


5.10 ER DIAGRAM


phn_no
phn_no
phn_no
Event
Status
id
facility_code_ no
FACILITY
1
HAS
M
user_id
Booking_date
BOOKING
M
id
Booked_on
facility_code_ no
HAS
house_no
Block
email
1
USER
name
Role
password
1
HAS
M
phn_no
name
id
VISITOR
house_no
date_of_visit
block
purpose
address
News
id
news
date

6. SCREENSHOTS

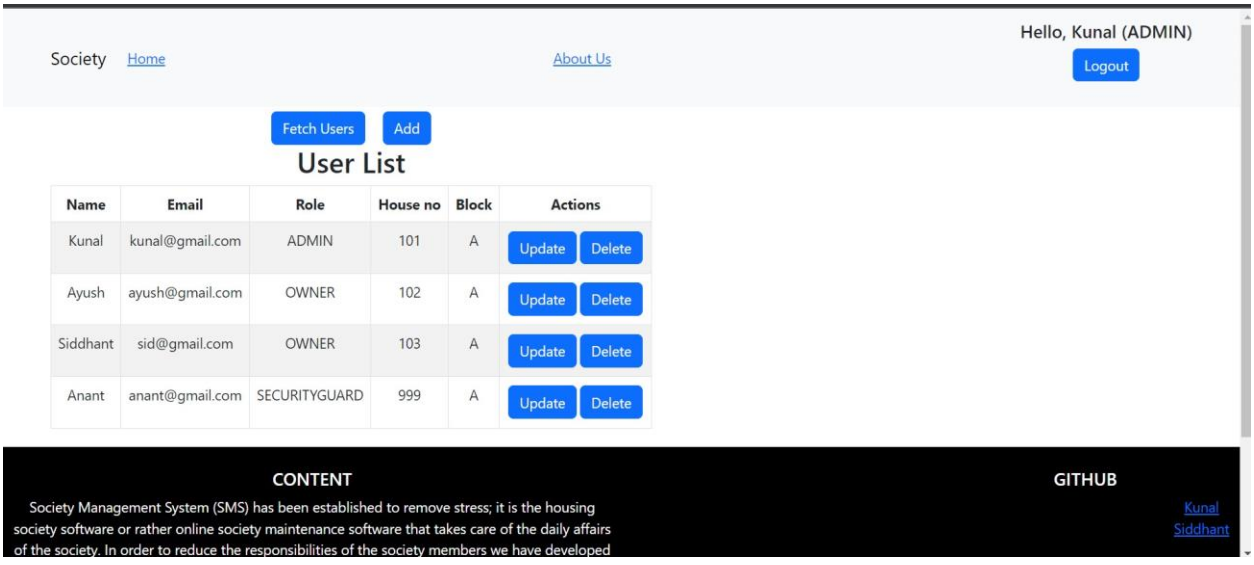
1.Home Page



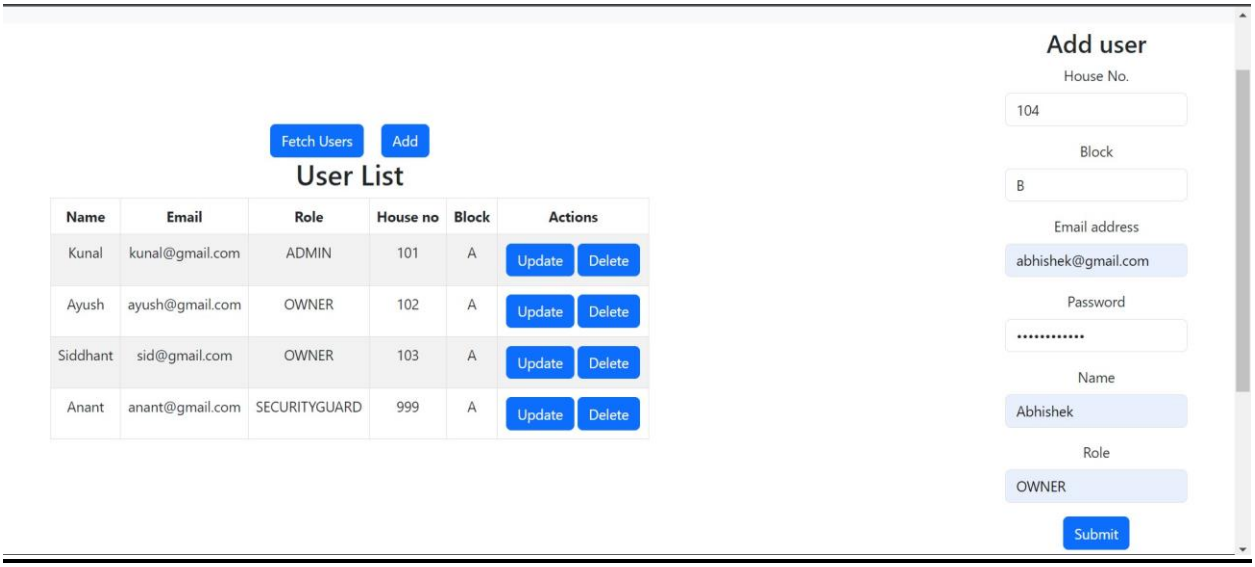
2. Login Id of Admin

3. Admin home page after login

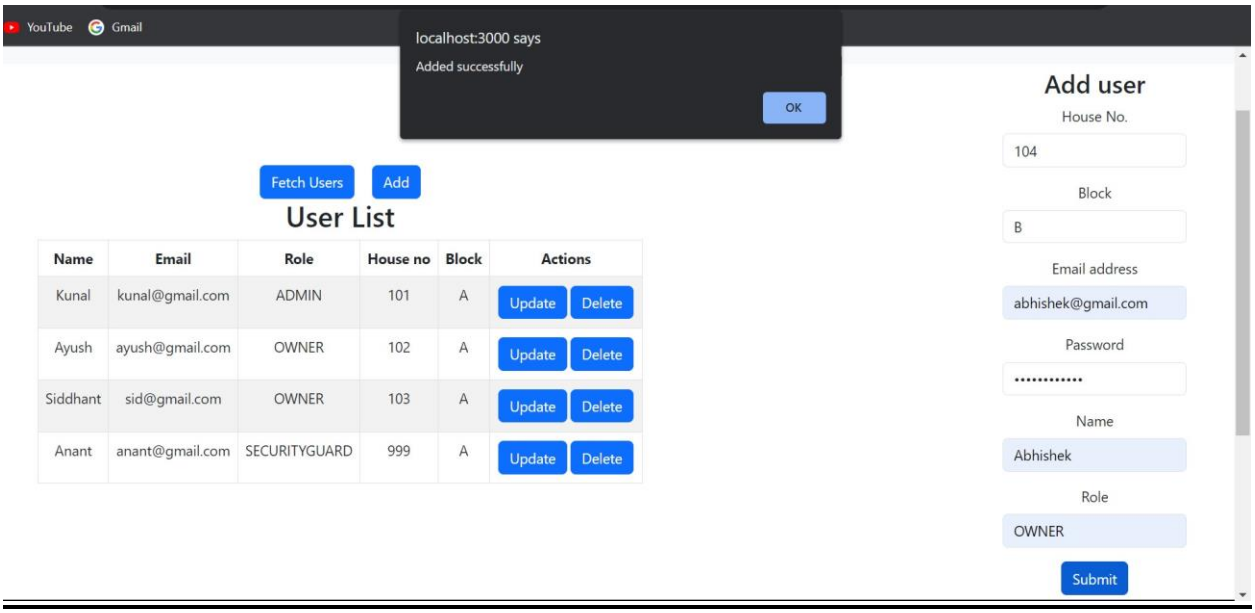
4. User Page



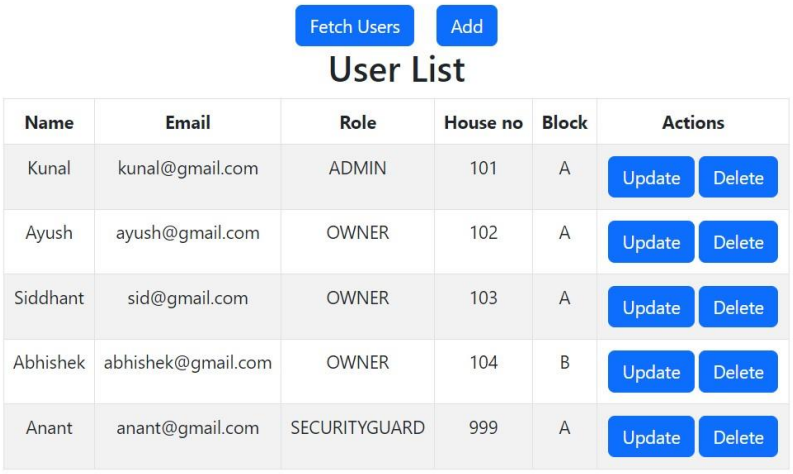
5. Add User



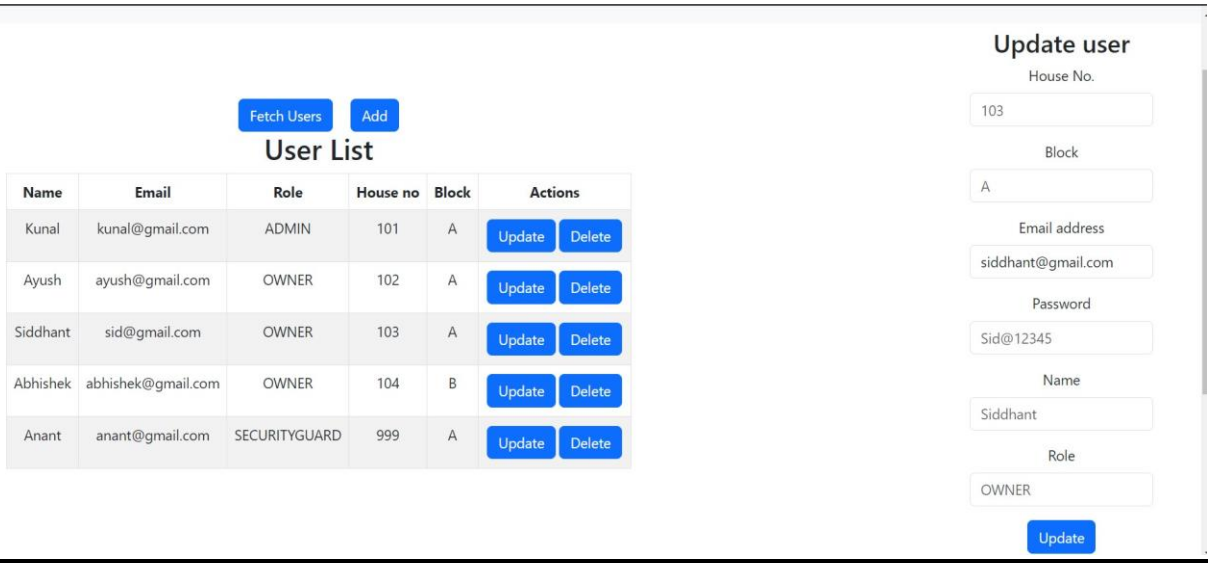
6. User Added



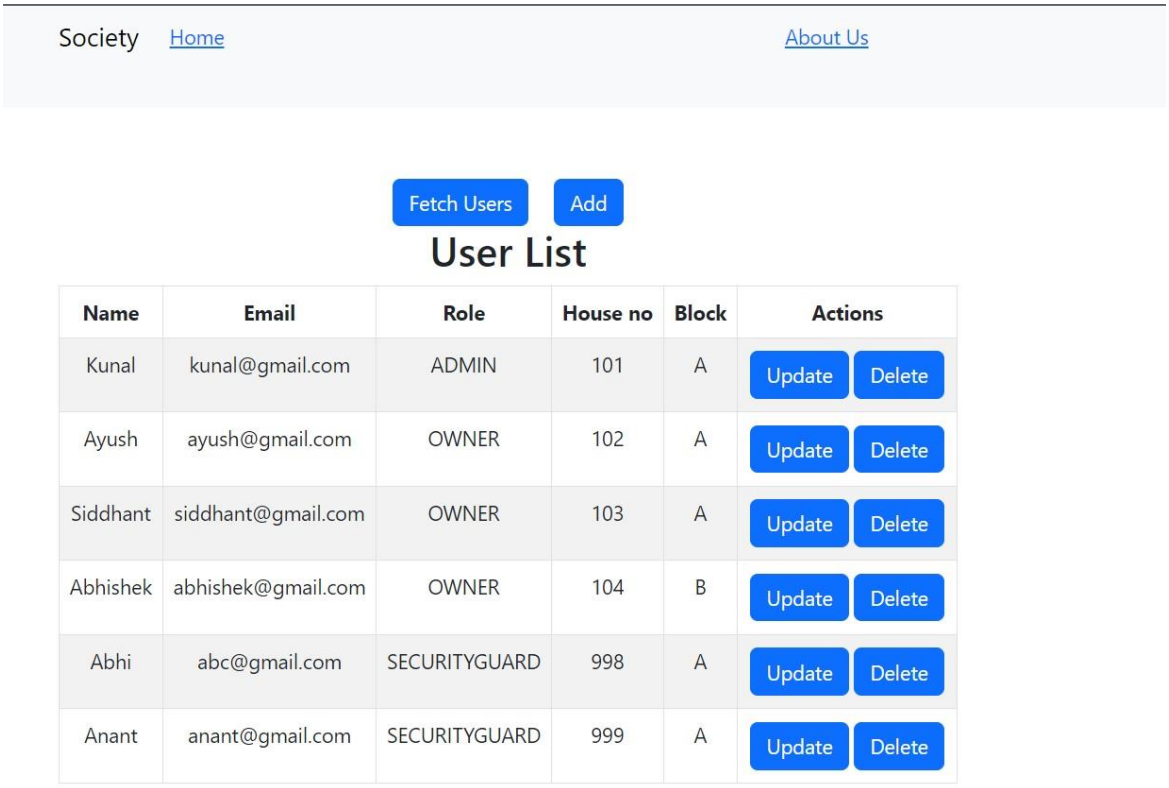
7. Fetch Users



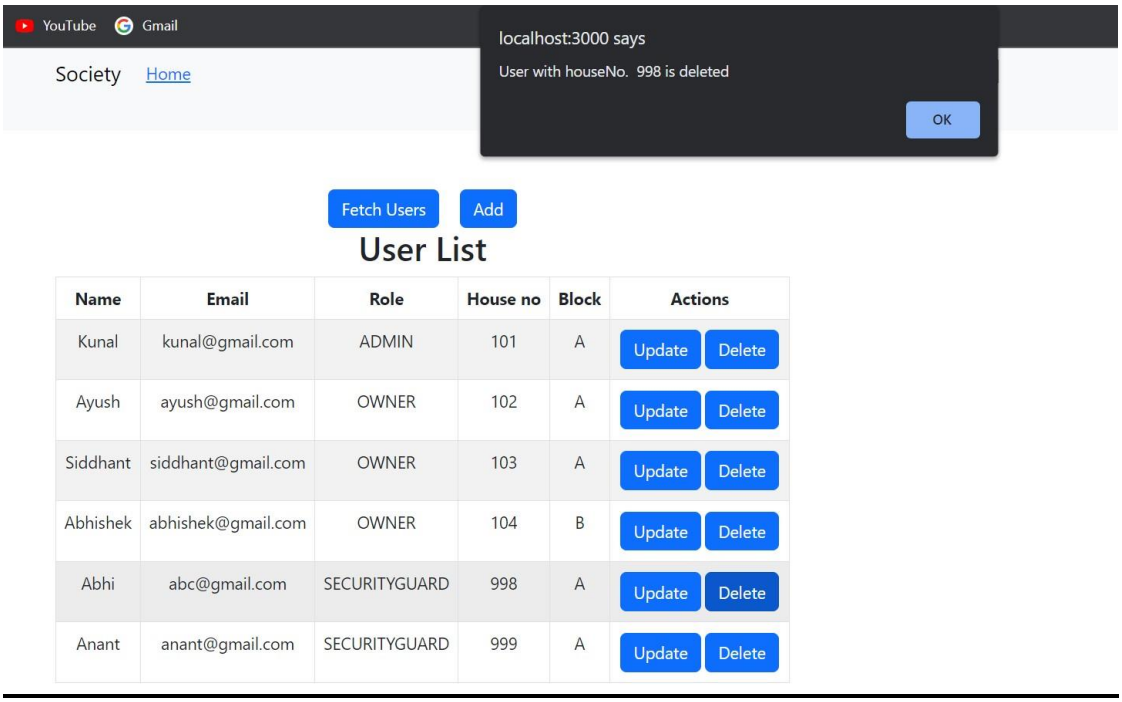
8. Update User



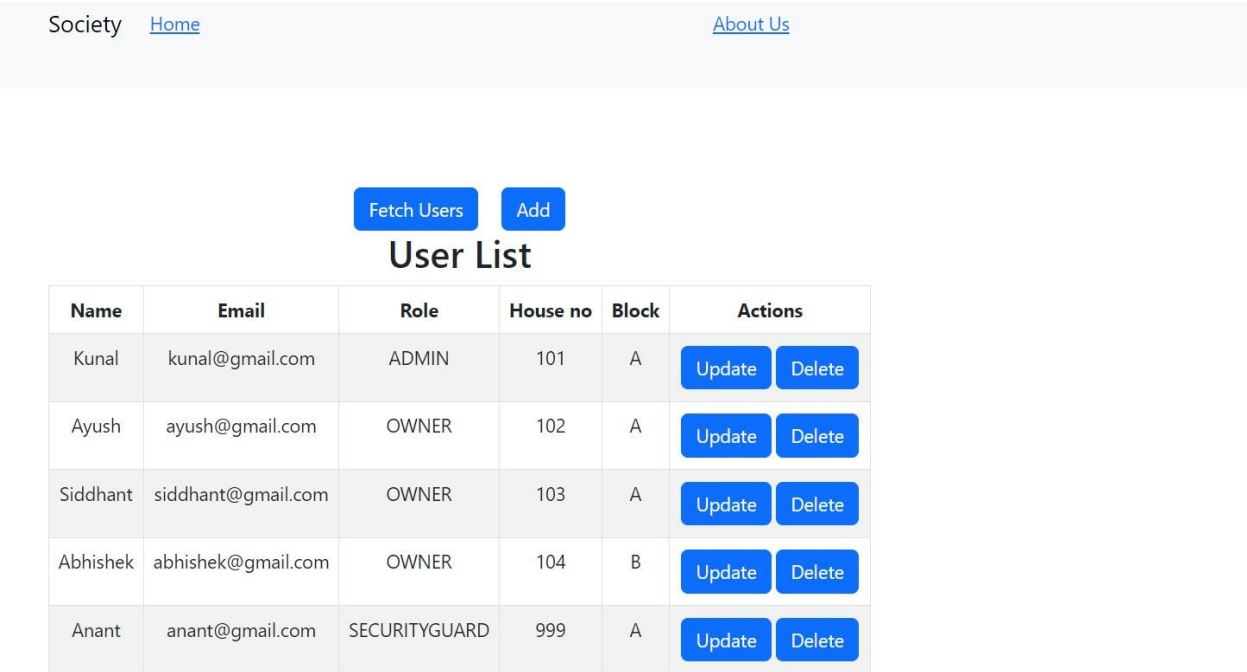
9. Updated User



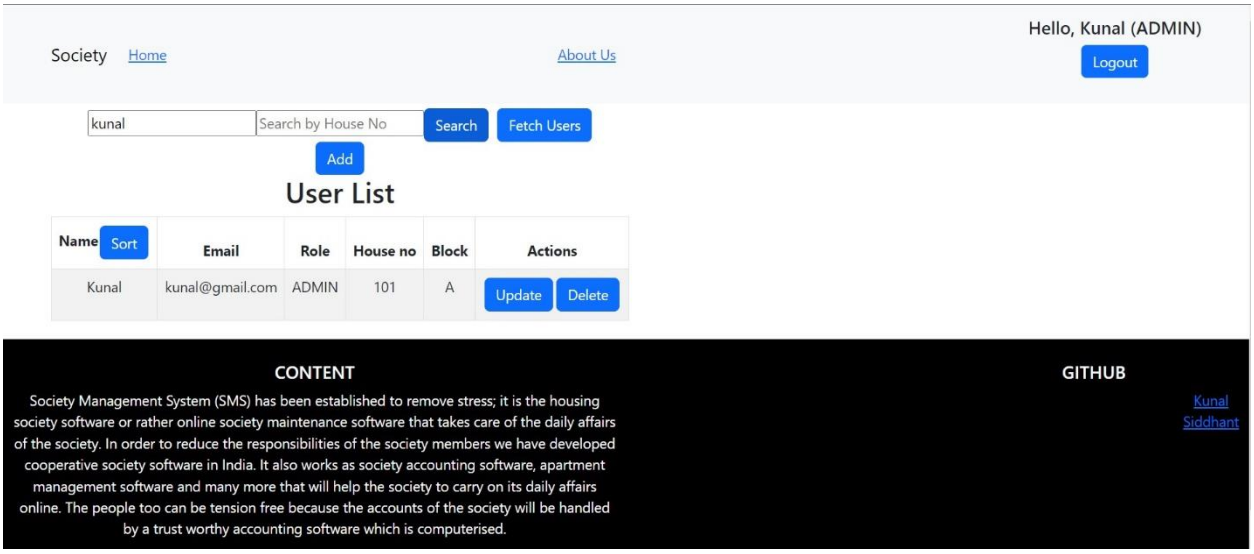
10. Deleted User



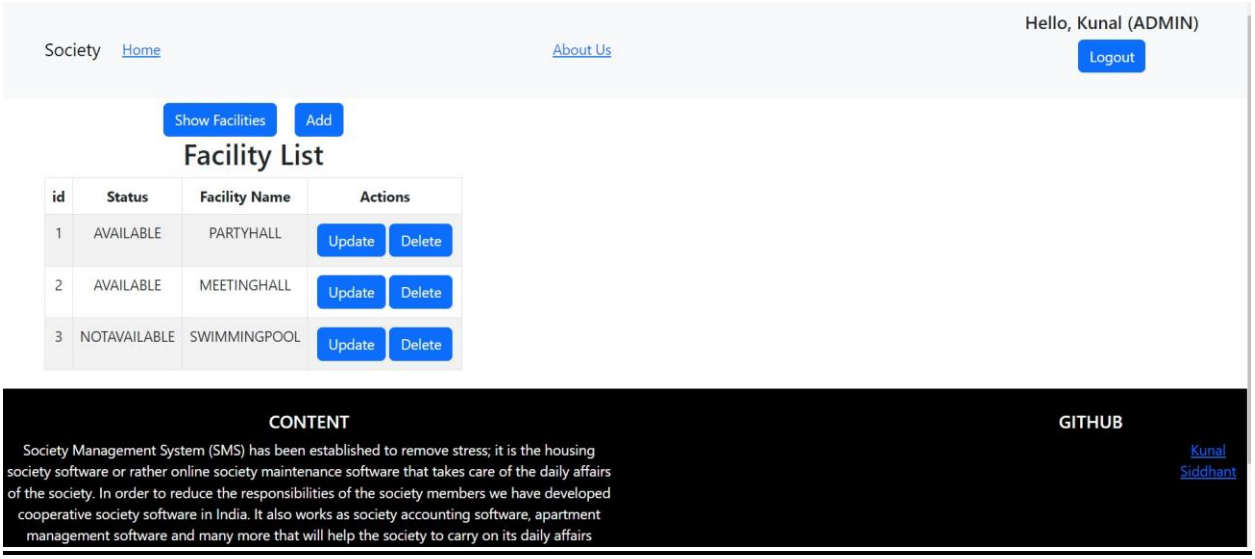
11. User list after deletion



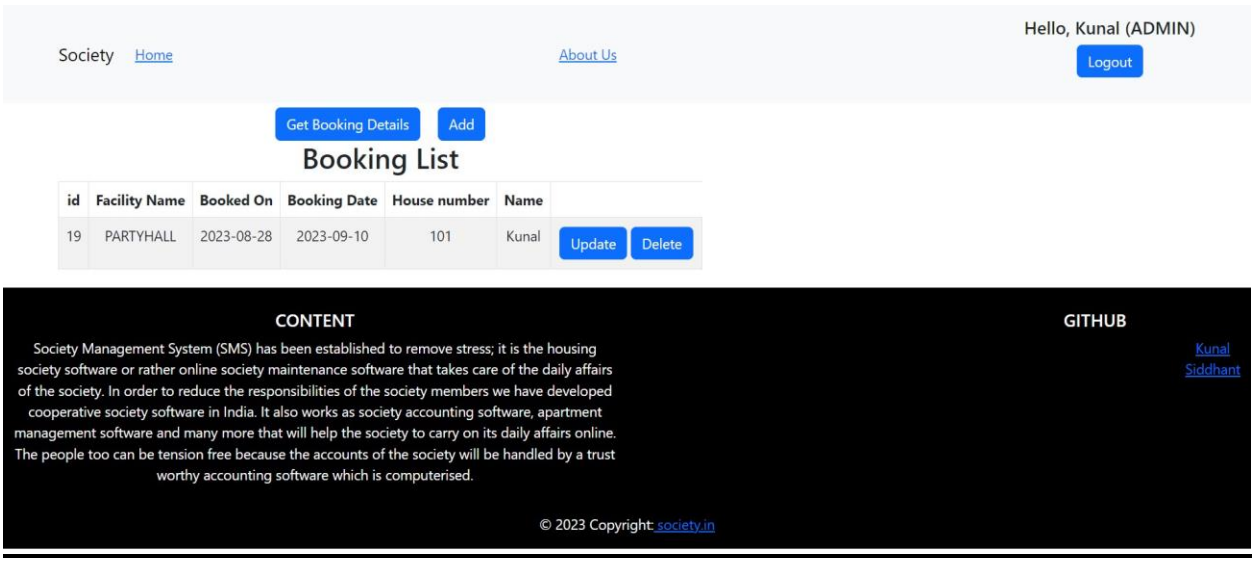
12. Search User by Name



13. Facility page

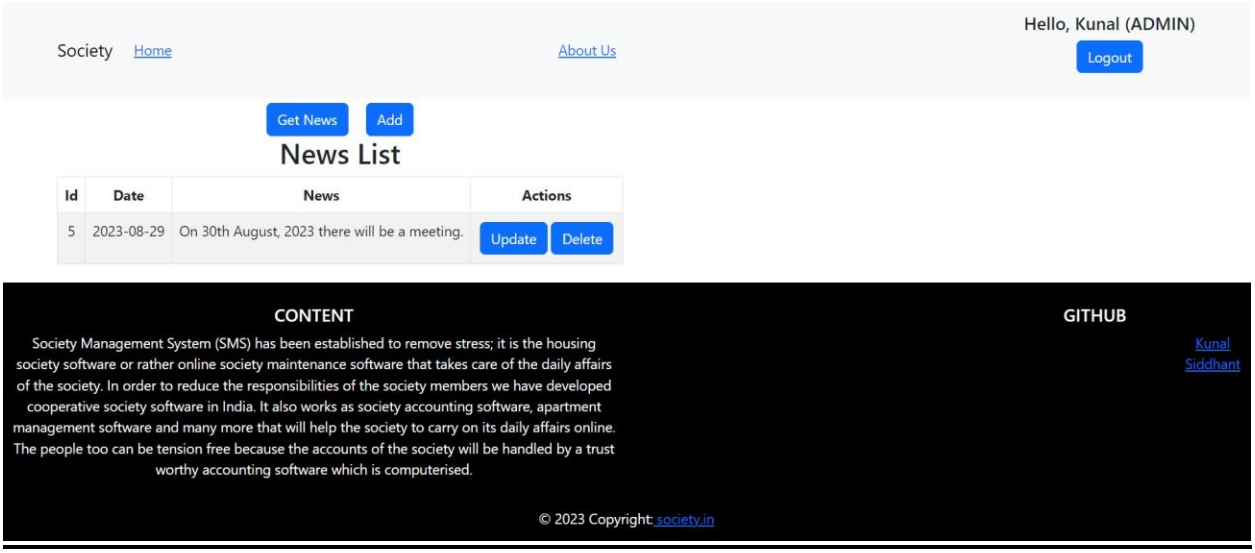


14. Booking page

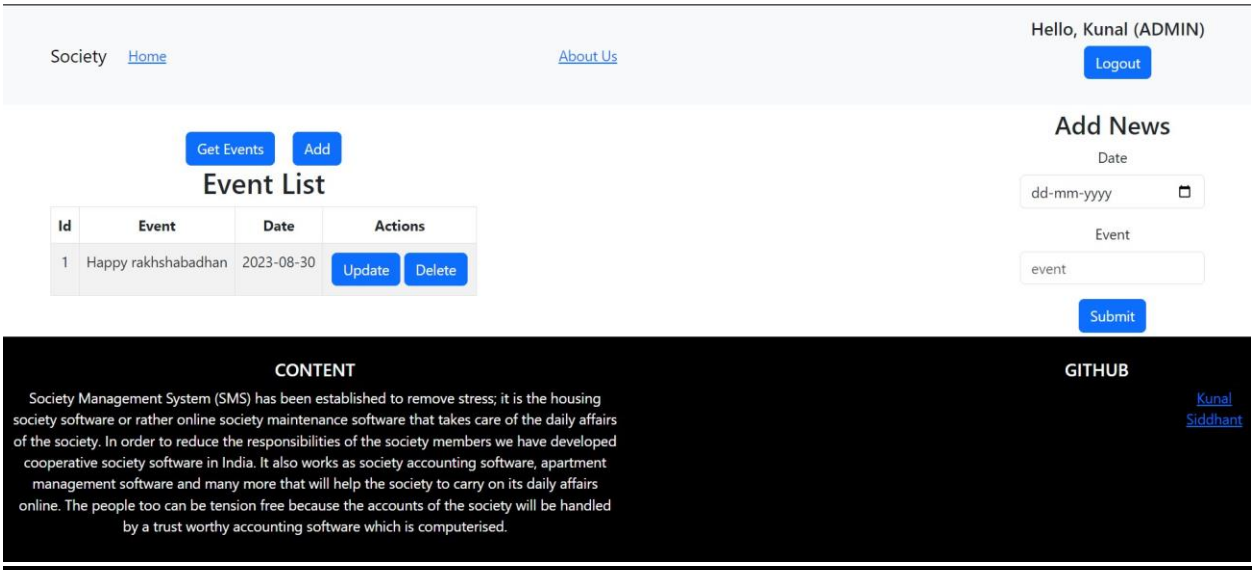


15. News and Event page

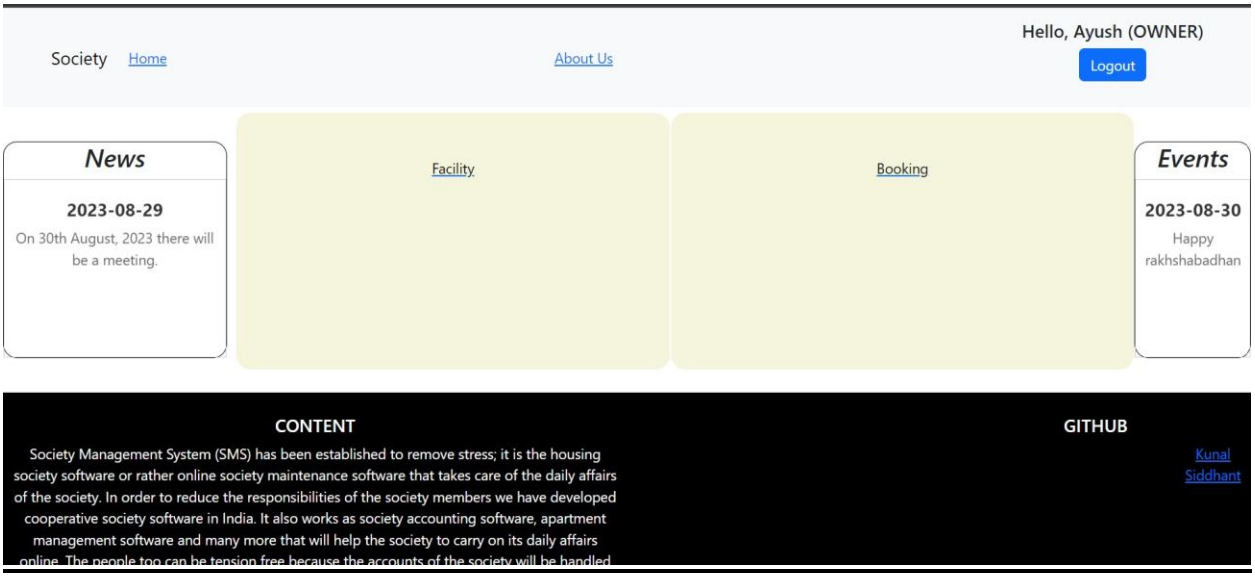
16. News page inside "News and Event page"



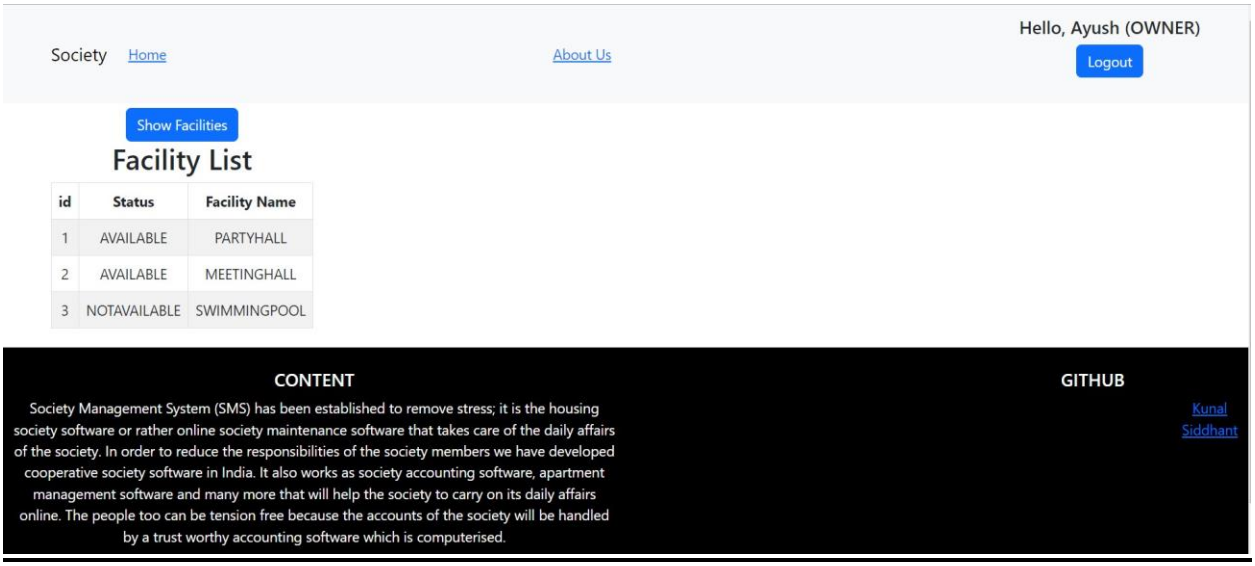
17. Event page inside “News and Event page”



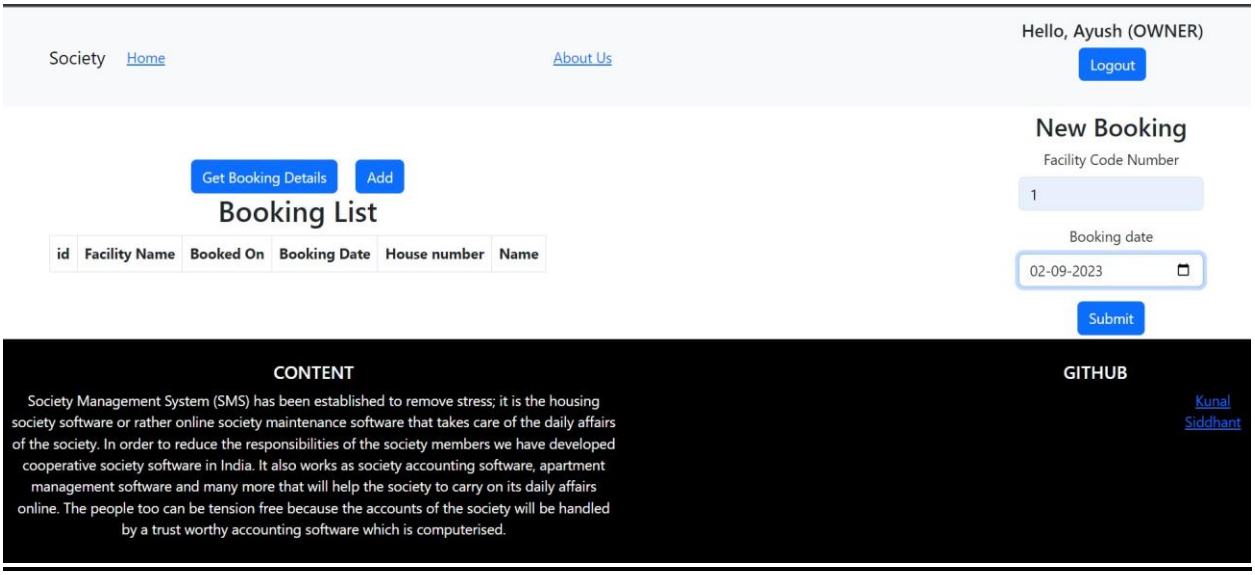
18. Owner Home page



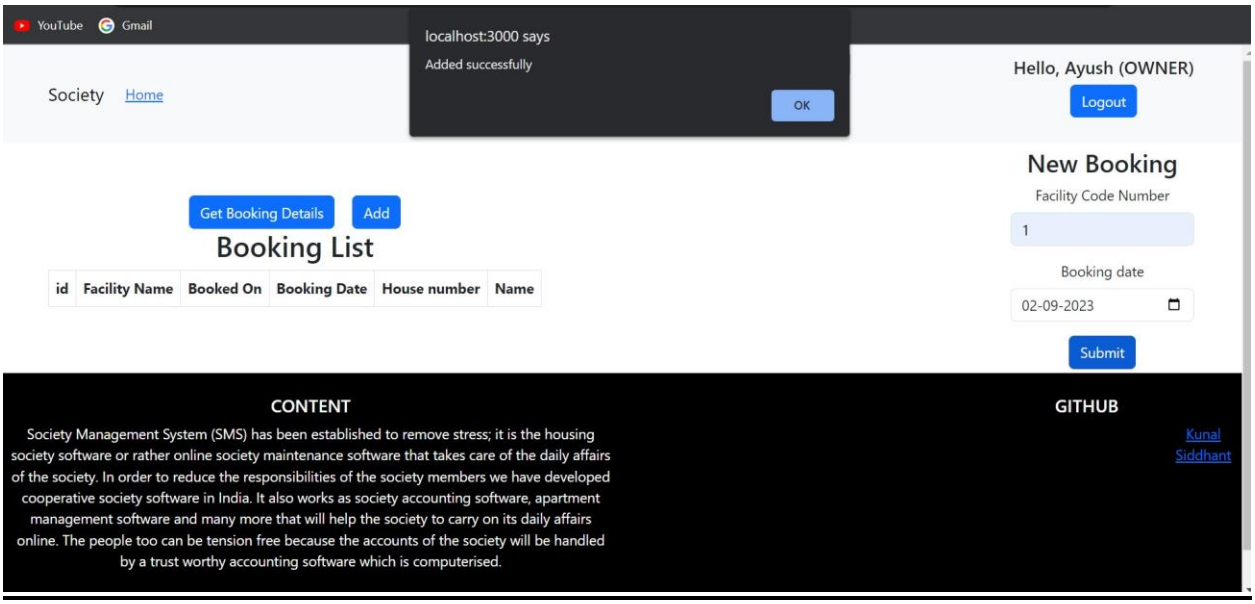
19. Facility page of Owner



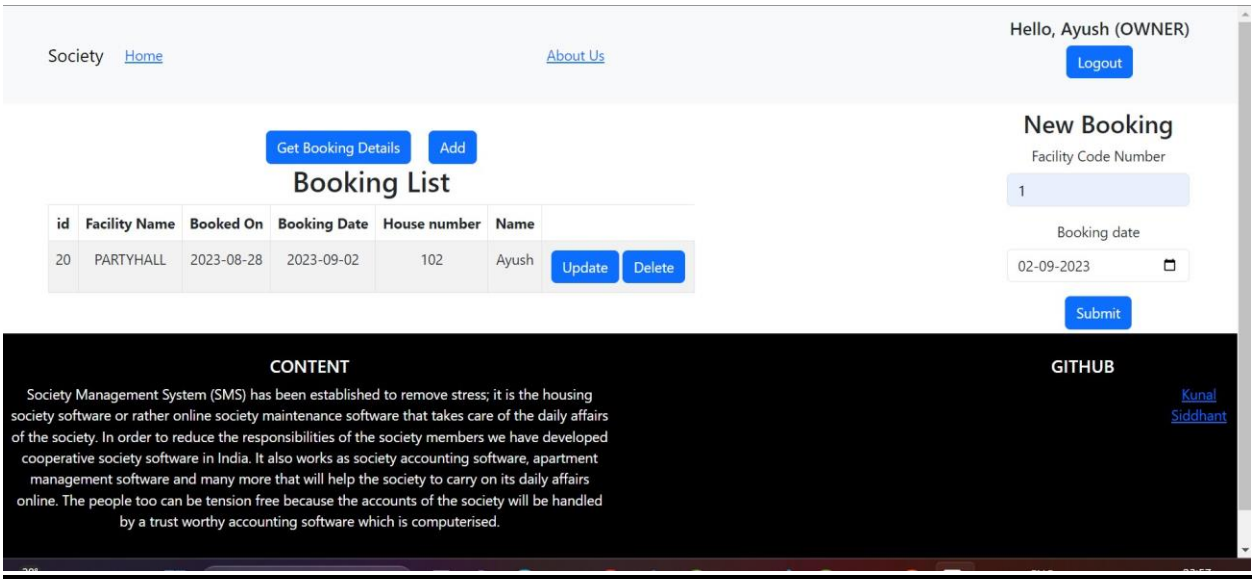
20. Booking page of Owner



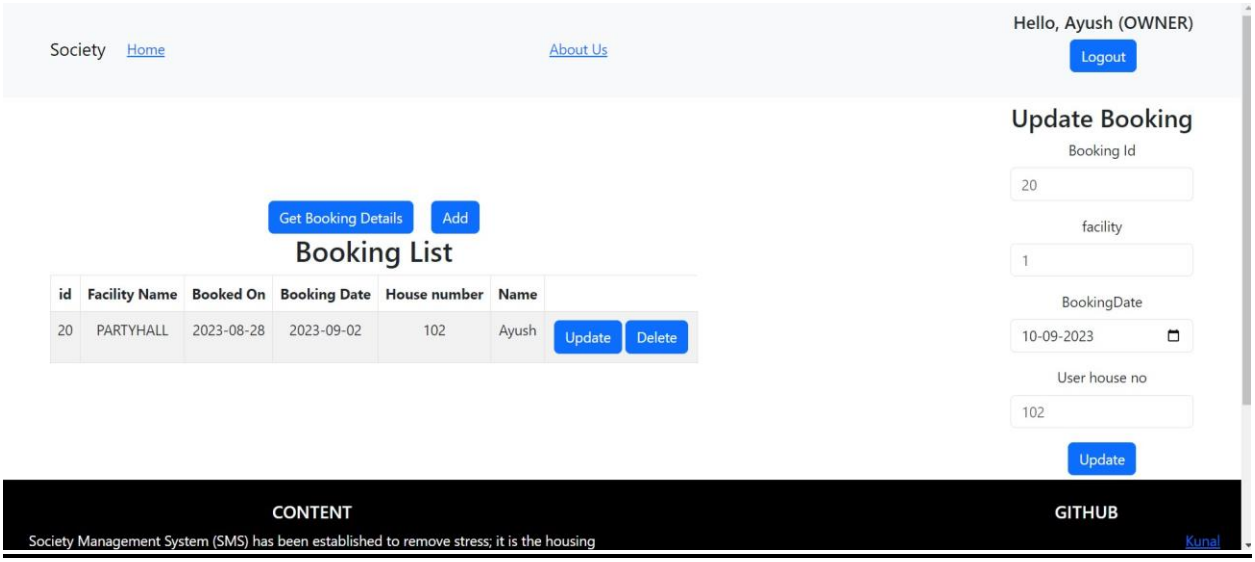
21. Booking done successfully of Owner



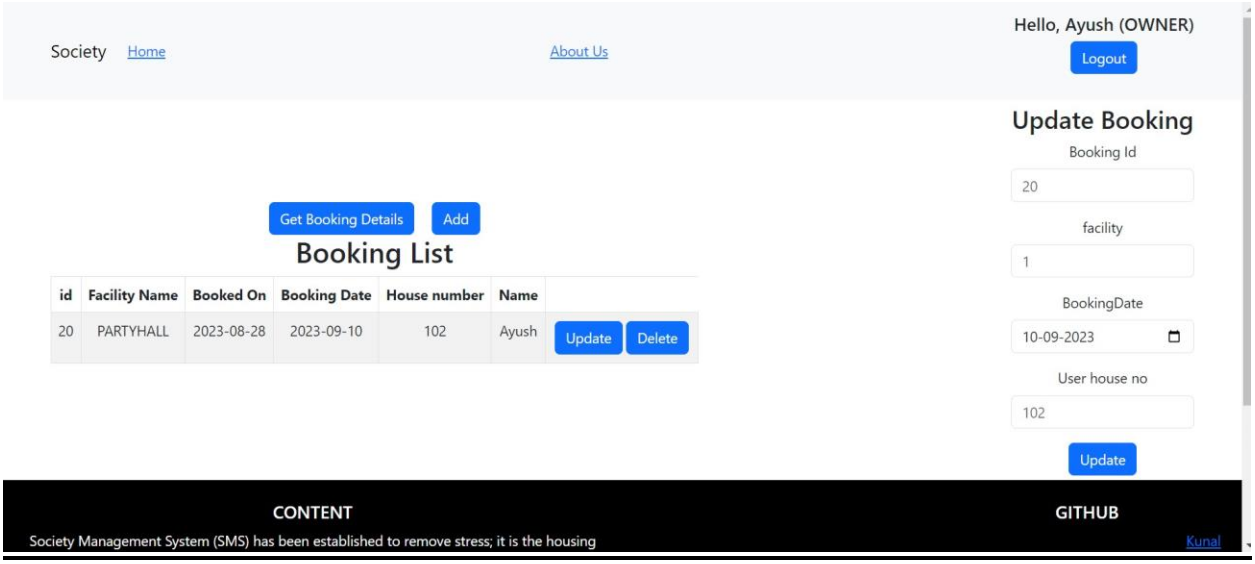
22. Booking history of Owner



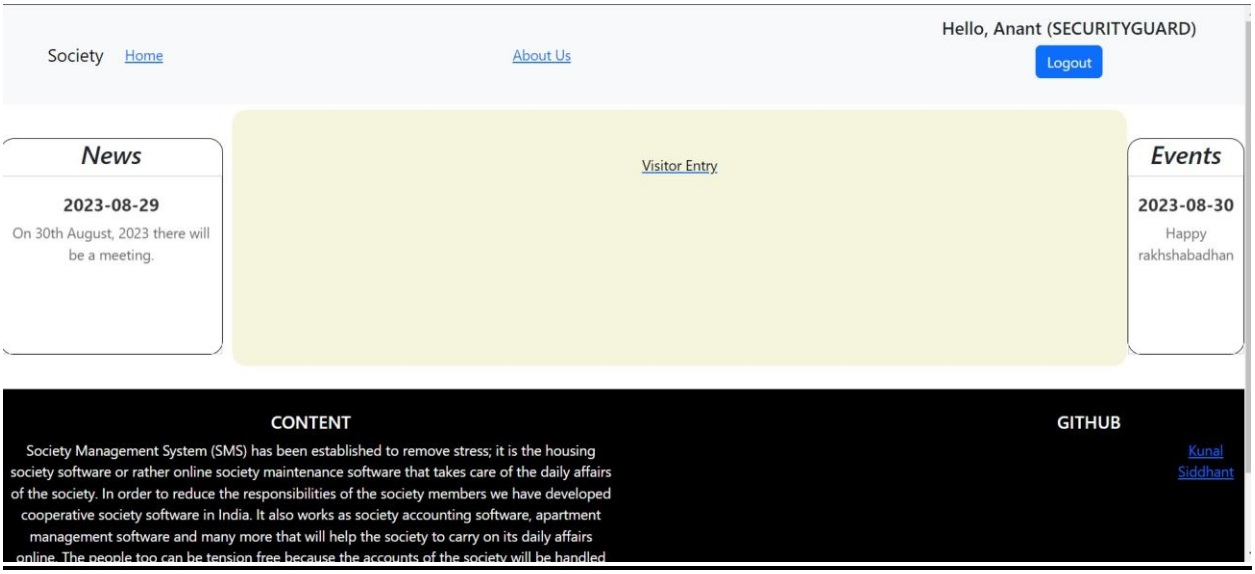
23. Update Booking details by Owner



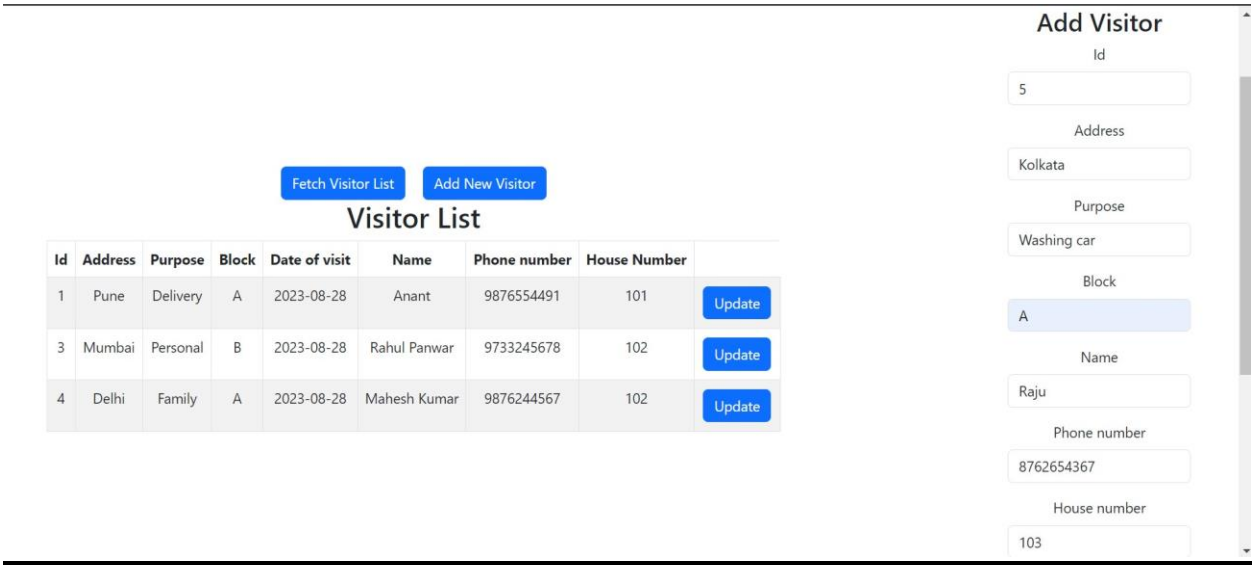
24. Booking details updated by Owner



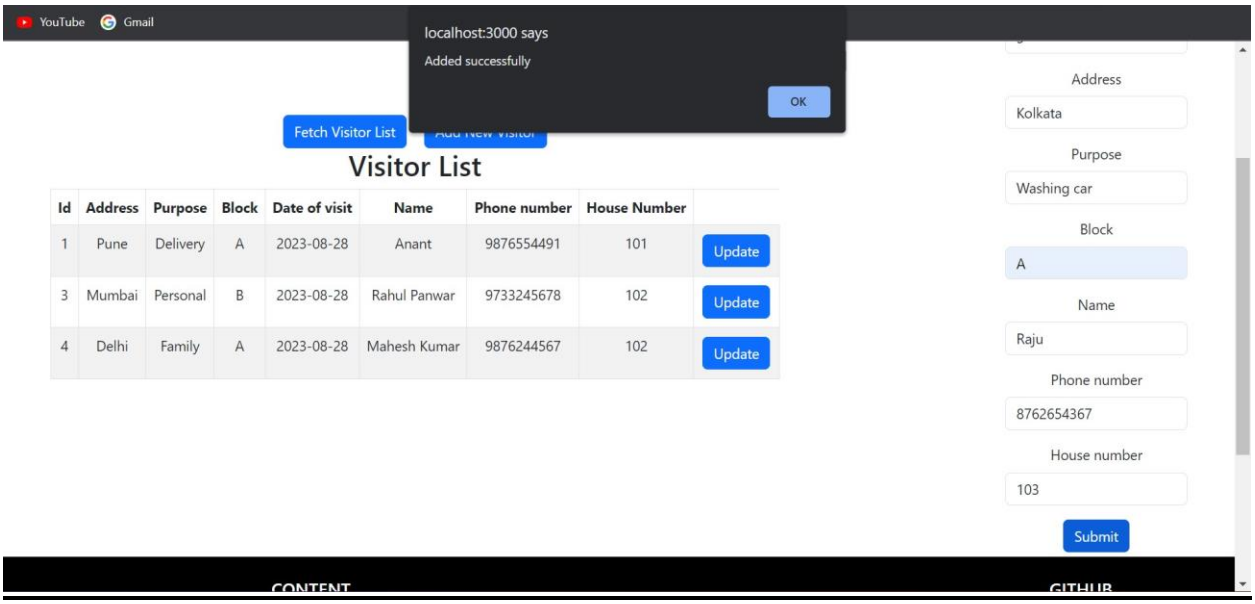
25. Security guard Home page



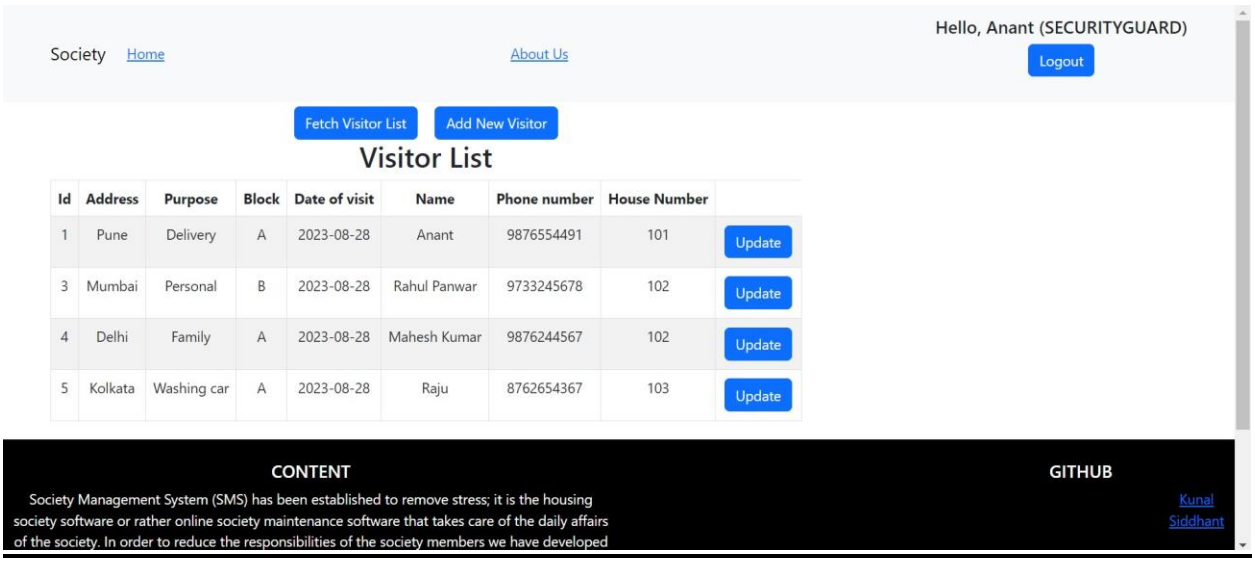
26. Visitor entry page of Security guard



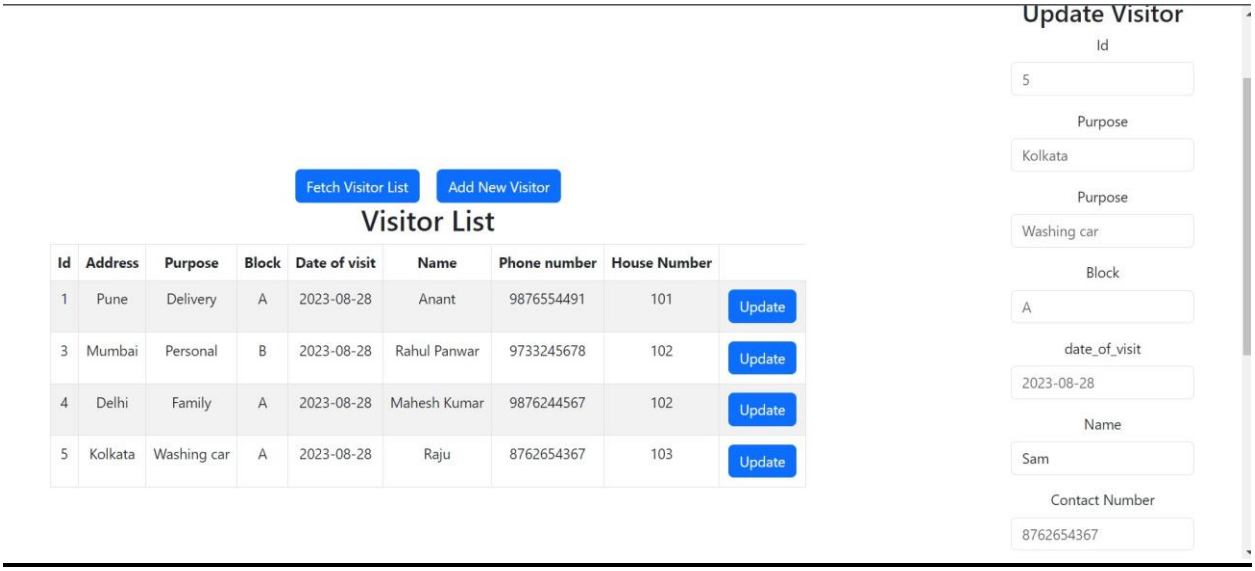
27. Visitor added by Security guard



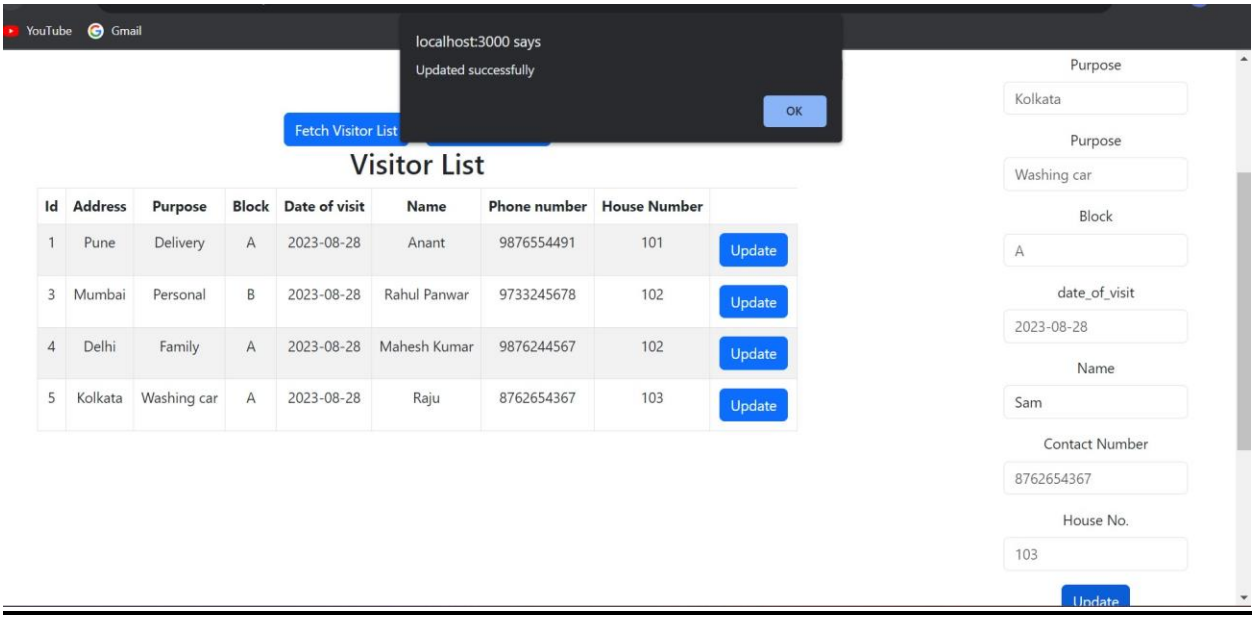
28. Visitor list after addition



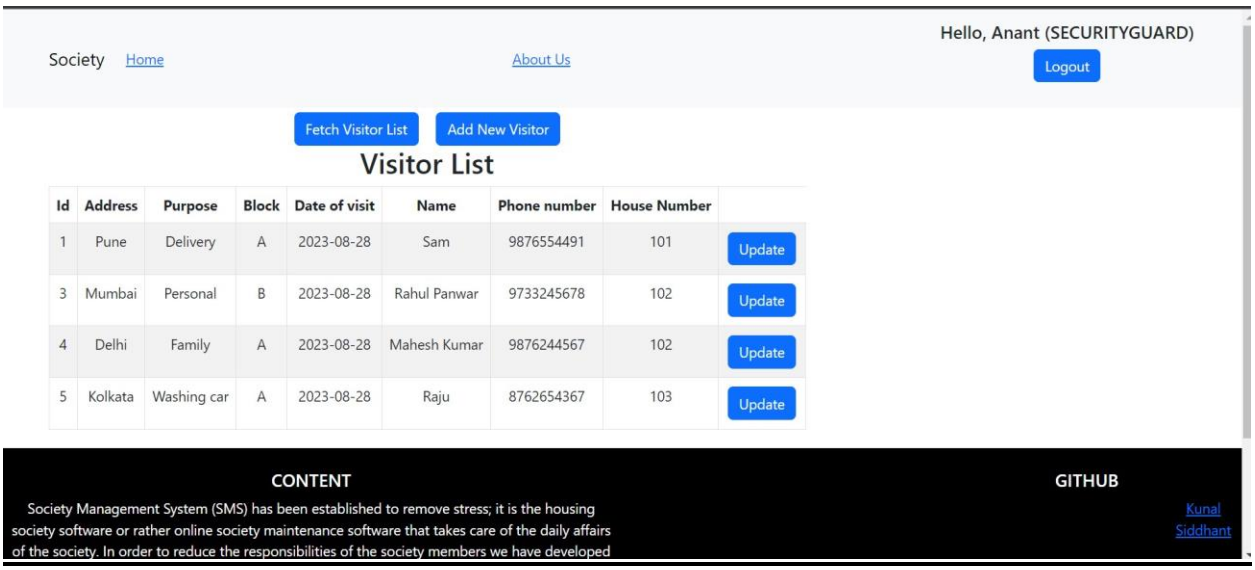
29. Update Visitor details by Security guard



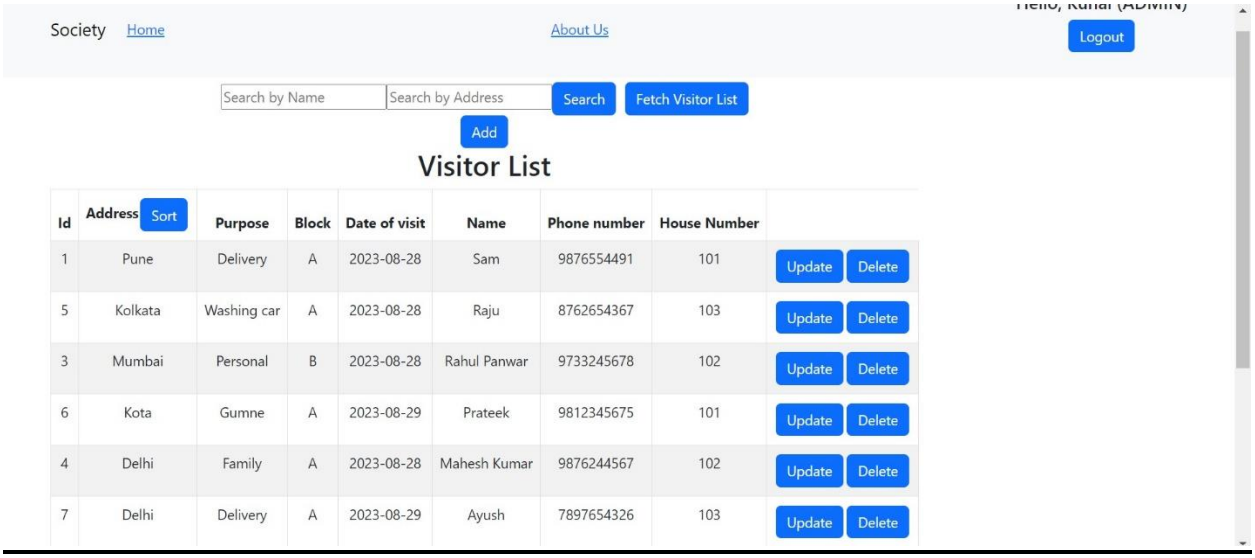
30. Visitor updated



31. Visitor List after updation



32. Sort visitor List by address



7. DESIGN

7.1 Database Design

The following table structures depict the database design.

All Tables

```
mysql> use society;
Database changed
mysql> show tables;
+-------------------+
| Tables_in_society |
+-------------------+
| booking           |
| event             |
| facility          |
| news              |
| user              |
| visitor           |
+-------------------+
6 rows in set (0.00 sec)
```

Table 1: Users

```
mysql> desc user;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| house_no | bigint      | NO   | PRI | NULL    |       |
| block    | varchar(1)  | YES  |     | NULL    |       |
| email    | varchar(40) | YES  | UNI | NULL    |       |
| name     | varchar(20) | YES  |     | NULL    |       |
| password | varchar(12) | NO   |     | NULL    |       |
| role     | int         | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Table 2: Visitors

```
mysql> desc visitor;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id           | bigint      | NO   | PRI | NULL    | auto_increment |
| purpose      | varchar(50) | YES  |     | NULL    |                |
| address      | varchar(50) | YES  |     | NULL    |                |
| block        | varchar(1)  | YES  |     | NULL    |                |
| date_of_visit | date       | YES  |     | NULL    |                |
| name         | varchar(20) | YES  |     | NULL    |                |
| phn_no       | varchar(10) | YES  |     | NULL    |                |
| house_no     | bigint      | YES  | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
8 rows in set (0.01 sec)
```

Table 3: Facilities

```
mysql> desc facility;
+------------------+--------+------+-----+---------+----------------+
| Field            | Type   | Null | Key | Default | Extra          |
+------------------+--------+------+-----+---------+----------------+
| id               | bigint | NO   | PRI | NULL    | auto_increment |
| status           | int    | YES  |     | NULL    |                |
| facility_code_no | int    | YES  |     | NULL    |                |
+------------------+--------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Table 4: News and Events

```
mysql> desc news;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | bigint      | NO   | PRI | NULL    | auto_increment |
| date  | date        | YES  |     | NULL    |                |
| news  | varchar(100) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> desc event;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| id     | bigint      | NO   | PRI | NULL    | auto_increment |
| events | varchar(100) | YES  |     | NULL    |                |
| date   | date        | YES  |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Table 5: Booking

```
mysql> desc booking;
+------------------+--------+------+-----+---------+----------------+
| Field            | Type   | Null | Key | Default | Extra          |
+------------------+--------+------+-----+---------+----------------+
| id               | bigint | NO   | PRI | NULL    | auto_increment |
| booked_on        | date   | YES  |     | NULL    |                |
| booking_date     | date   | YES  | UNI | NULL    |                |
| facility_code_no | bigint | YES  | MUL | NULL    |                |
| user_id          | bigint | YES  | MUL | NULL    |                |
+------------------+--------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

7.2 System Design of Society Management System:

In this phase, a logical system is built which fulfills the given requirements. Design phase of software development deals with transforming the client's requirements into a logically working system. Normally, design is performed in the following in the following two steps:

1. **Primary Design Phase:**

   In this phase, the system is designed at block level. The blocks are created on the basis of analysis done in the problem identification phase. Different blocks are created for different functions; emphasis is put on minimizing the information flow between blocks. Thus, all activities which require more interaction are kept in one block.

2. **Secondary Design Phase:**

   In the secondary phase the detailed design of every block is performed.

The general tasks involved in the design process are the following:

1. Design various blocks for overall system processes.

2. Design smaller, compact and workable modules in each block.

3. Design various database structures.

4. Specify details of programs to achieve desired functionality.

5. Design the form of inputs, and outputs of the system.

6. Perform documentation of the design.

7. System reviews

7.3 User Interface Design:

User Interface Design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventual presentation of desired inputs and outputs. The overall flow of screens and messages is called a dialogue.

The following steps are various guidelines for User Interface Design:

1. The system user should always be aware of what to do next.

2. The screen should be formatted so that various types of information, instructions and messages always appear in the same general display area.

3. Messages, instructions or information should be displayed long enough to allow the system user to read them.

4. Use display attributes sparingly.

5. Default values for fields and answers to be entered by the user should be specified.

6. A user should not be allowed to proceed without correcting an error.

7. The system user should never get an operating system message or fatal error.

8. CODING STANDARDS IMPLEMENTED

Naming and Capitalization

Below summarizes the naming recommendations for identifiers in Pascal casing is used mainly (i.e. capitalize first letter of each word) with camel casing (capitalize each word except for the first one) being used in certain circumstances**.**

Identifier	**Case**	**Examples**	**Additional Notes**
Class	Pascal	Person, BankVault, SMSMessage, Dept	Class names should be based on "objects" or "real things" should generally be nouns. No '_' signs allowed. Do not use type prefixes like 'C' for class.
Method	Camel	getDetails, updateStore	Methods should use verbs or verb phrases

Parameter	Camel	personName, bankCode	Use descriptive parameter names. Parameter names should be descriptive enough that the name of the parameter and its type can be used to determine its meaning in most scenarios.
Interface	Pascal with "I" prefix	Disposable	Do not use the ‘_’ sign
Property	Pascal	ForeColor, BackColor	Use a noun or noun phrase to name properties
Associated private member variable	_camelCase	_foreColor, _backColo	Use underscore camel casing for the private member variables
Exception Class	Pascal with "Exception" suffix	WebException	

Comments

- Comment each type, each non-public type member, and each region declaration.
- Use end-line comments only on variable declaration lines. End-line comments are comments that follow code on a single line.
- Separate comments from comment delimiters (apostrophe) or // with one space.
- Begin the comment text with an uppercase letter.
- End the comment with a period.
- Explain the code; do not repeat it.

9. Conclusion of the project Society Management System

In conclusion, the implementation of a comprehensive Society Management System software serves as a pivotal solution to the diverse needs of today's residential communities. By seamlessly integrating modules for adding visitor details, maintaining house owner information, booking facilities like party halls and meeting halls, and providing a platform to view news and events, the software transforms the way societies are managed and experienced.

Through efficient visitor management, the software prioritizes security and accountability while simplifying administrative tasks. Accurate house owner details ensure effective communication and streamlined record-keeping. The facility booking feature optimizes the usage of community spaces, enhancing convenience for residents. The dedicated section for news and events fosters engagement and a sense of community by keeping residents well-informed and connected.

Collectively, the Society Management System software transcends traditional management approaches, offering a centralized and user-friendly platform that enhances security, communication, convenience, and engagement within the community. By embracing technology to meet these fundamental needs, the software redefines the standard of living and management efficiency for modern residential societies.

10. Future Scope of the Project

The future scope of a Society Management System software for adding visitor details, maintaining house owner details, booking facilities, and viewing news and events holds tremendous potential for further enhancements and advancements. As technology continues to evolve, here are some potential directions that the software could explore:

1. Mobile Application Integration: Develop dedicated mobile apps for residents, security personnel, and administrators, allowing them to access and manage the system on the go. This would enhance convenience and real-time interaction.

2. Integration with Smart Devices: Integrate the system with IoT devices such as smart locks and cameras to automate visitor entry, enhance security, and streamline the visitor registration process.

3. Biometric Integration: Implement biometric authentication for visitors, making the check-in process more secure and efficient.

4. Predictive Analytics: Utilize data collected from visitor records, facility bookings, and resident interactions to generate insights and predictions for future resource allocation and community engagement strategies.

5. AI-powered Chatbots: Implement AI chatbots to assist residents with queries, provide event suggestions, and facilitate bookings, improving user experience.

6. Virtual Facility Tours: Integrate virtual reality to offer residents virtual tours of community facilities before booking, enhancing their understanding and choice of available spaces.

7. Automated Reminders: Implement automated reminders for upcoming events, facility bookings, and important community announcements, enhancing resident engagement.

8. Community Marketplace: Create a platform within the software where residents can buy, sell, or exchange goods and services, fostering a sense of shared economy and community support.

9. Localized Notifications: Provide targeted notifications to specific areas or blocks within the society to personalize news and events based on residents' preferences.

10. Social Integration: Enable residents to connect with each other through the software, facilitating the formation of interest groups, forums, and community discussions.

11. Energy and Resource Management: Expand the system's capabilities to monitor and optimize energy consumption, water usage, and other resources within the community.

12. Sustainability Initiatives: Incorporate features that promote eco-friendly practices within the society, such as recycling programs and events promoting sustainability.

13. Integration with Local Services: Collaborate with local businesses and service providers to offer residents discounts, services, and promotions through the software.

14. Voice Command Integration: Enable residents to interact with the system using voice commands for hands-free access and convenience.

15. Augmented Reality (AR) Navigation: Implement AR navigation within the community to guide residents to facilities, events, and specific locations.

As technology advances and user needs evolve, the Society Management System software has the potential to become an integral part of daily life within residential communities, offering an all-encompassing platform that enhances convenience, engagement, security, and overall quality of life.

11. Limitations of the project

The future scope of a Society Management System software for adding visitor details, maintaining house owner details, booking facilities, and viewing news and events holds tremendous potential for further enhancements and advancements. As technology continues to evolve, here are some potential directions that the software could explore:

1. Mobile Application Integration: Develop dedicated mobile apps for residents, security personnel, and administrators, allowing them to access and manage the system on the go. This would enhance convenience and real-time interaction.

2. Integration with Smart Devices: Integrate the system with IoT devices such as smart locks and cameras to automate visitor entry, enhance security, and streamline the visitor registration process.

3. Biometric Integration: Implement biometric authentication for visitors, making the check-in process more secure and efficient.

4. Predictive Analytics: Utilize data collected from visitor records, facility bookings, and resident interactions to generate insights and predictions for future resource allocation and community engagement strategies.

5. AI-powered Chatbots: Implement AI chatbots to assist residents with queries, provide event suggestions, and facilitate bookings, improving user experience.

6. Virtual Facility Tours: Integrate virtual reality to offer residents virtual tours of community facilities before booking, enhancing their understanding and choice of available spaces.

7. Automated Reminders: Implement automated reminders for upcoming events, facility bookings, and important community announcements, enhancing resident engagement.

8. Community Marketplace: Create a platform within the software where residents can buy, sell, or exchange goods and services, fostering a sense of shared economy and community support.

9. Localized Notifications: Provide targeted notifications to specific areas or blocks within the society to personalize news and events based on residents' preferences.

10. Social Integration: Enable residents to connect with each other through the software, facilitating the formation of interest groups, forums, and community discussions.

11. Energy and Resource Management: Expand the system's capabilities to monitor and optimize energy consumption, water usage, and other resources within the community.

12. Sustainability Initiatives: Incorporate features that promote eco-friendly practices within the society, such as recycling programs and events promoting sustainability.

13. Integration with Local Services: Collaborate with local businesses and service providers to offer residents discounts, services, and promotions through the software.

14. Voice Command Integration: Enable residents to interact with the system using voice commands for hands-free access and convenience.

15. Augmented Reality (AR) Navigation: Implement AR navigation within the community to guide residents to facilities, events, and specific locations.

As technology advances and user needs evolve, the Society Management System software has the potential to become an integral part of daily life within residential communities, offering an all-encompassing platform that enhances convenience, engagement, security, and overall quality of life.

12. References

- http://www.javaworld.com/javaworld/jw-01-1998/jw-01-Credentialreview.html
- We utilized an AI language model known as ChatGPT to assist in generating responses and providing information. ChatGPT is a state-of-the-art AI developed by OpenAI, designed to understand and generate human-like text based on the input provided.
- Database Programming with JDBC and Java by O'Reilly
- Head First Java 2nd Edition
- http://www.jdbc-tutorial.com/
- Java and Software Design Concepts by Après
- https://www.tutorialspoint.com/java/
- http://www.javatpoint.com/java-tutorial
- https://docs.oracle.com/javase/tutorial/
- http://www.wampserver.com/en/
- http://www.JSP.net/
- http://www.tutorialspoint.com/mysql/
- httpd.apache.org/docs/2.0/misc/tutorials.html