UrbanUnity:
Residence Community Management

A Project Report Submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

in

Information Technology

by

Daksh Dadhania, Zaid Khan, Syed Hasnain Raza

210911072, 210911068,210911038

Under the guidance of

Dr. Swathi B.P (Lab Faculty 1)
Assistant Professor- Sr Scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

Dr.Girija Attigeri(Lab faculty 2)
Assistant Professor
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Unit of MAHE, Manipal

November 2023

Abstract

UrbanUnity: Residence Community Mgmt System revolutionizes residential community management by transitioning from manual data entry to a sophisticated digital interface. Developed as part of a Database Management System Course, this system streamlines operations and enhances living experiences in apartment complexes.

UrbanUnity offers tailored functionalities for administrators, owners, tenants, and employees. It enables efficient management of tenant and owner details, parking assignments, and complaint handling. Owners can manage tenant information and track property-related complaints, while tenants benefit from features like online maintenance fee payment and complaint submission. Employees are equipped to address and track complaints effectively.

The system's interface, developed using HTML5, Tailwind CSS, and React JS, ensures a responsive and intuitive user experience. The backend, powered by NodeJS and ExpressJS, coupled with a MySQL database, provides robust data handling and storage.

UrbanUnity is not just a management tool; it's a comprehensive solution fostering a collaborative, efficient, and secure community living environment, enhancing the lifestyle for residents and property owners.

ACM Computing Classification System (CCS) Taxonomy:

Information systems ➔ Database management systems

Software and its engineering ➔ Software creation and management

Human-centered computing ➔ Human-computer interaction (HCI)

Sustainable Development Goals by Govt. of India

UrbanUnity: Residence Community Management system, aligns with specific Sustainable Development Goals (SDGs) set by the United Nations, which are also endorsed by the Government of India. The most relevant SDGs for your project are:

SDG 11: Sustainable Cities and Communities:

UrbanUnity significantly contributes to the creation of sustainable cities and communities. By streamlining the management of residential complexes, it enhances the efficiency and quality of community living. This aligns with SDG 11's aim to make cities inclusive, safe, resilient, and sustainable.

SDG 9: Industry, Innovation, and Infrastructure:

Your project supports SDG 9 by integrating innovative software solutions to improve the infrastructure of residential communities. The use of advanced technologies like HTML5, CSS3, React JS, NodeJS, ExpressJS, and MySQL in UrbanUnity promotes innovation and sustainable industrialization, contributing to building resilient infrastructure.

SDG 17: Partnerships for the Goals:

While not directly related to the technological aspects of UrbanUnity, this project can foster partnerships among various stakeholders, including residents, property managers, and technology providers. Such collaborations are essential for achieving the SDGs, particularly in sharing best practices, technology, and fostering a sense of community.

In summary, UrbanUnity aligns primarily with SDGs 11 and 9, contributing to sustainable urban development and innovation in infrastructure. Additionally, it indirectly supports SDG 17 by encouraging partnerships and collaborations within communities.

Table of Contents

List of Tables

Summary of the ten tables along with their primary roles within the UrbanUnity: Residence Community Mgmt System:

1. Admin:
    - Manages the entire system, with capabilities to view and edit tenant and owner details, create new owner entries, allocate parking slots, and handle complaints.
    - Monitors statistics, including the total count of owners, tenants, and employees.
2. Owner:
    - Manages their properties, views tenant details, creates new tenant profiles, handles complaints related to their properties, and views room details.
3. Tenant:
    - Views personal details, allotted parking slots, pays maintenance fees, and raises complaints.
4. Employee:
    - Manages and resolves complaints, and views the total number of complaints to keep track of the issues within the community.
5. Visitor:
    - Tracks visitors to the community by recording their ID, phone number, and the room and block they visit.
6. Room:
    - Manages details about each room, including type, floor, parking slot registration, and associated block number.
7. Block:
    - Manages block-wise details including block name and complaints associated with each block.
8. Rental:
    - Manages financial transactions with tenants, including the date of joining, monthly rent, tenant ID, and associated room number.
9. Auth:
    - Handles authentication credentials for users, storing passwords and user IDs for system access control.
10. Block Admin:
    - Manages individual blocks within the community, handling administrative tasks, complaints, and maintenance issues specific to their assigned blocks.

These tables collectively facilitate the comprehensive management of a residential community, ensuring the efficient operation of the UrbanUnity system across various aspects of property management, financial transactions, user authentication, and administrative control.

List of Figures

CHAPTER – 1

INTRODUCTION

1.1 Overview

Introducing UrbanUnity, an advanced Residence Community Management System engineered to revolutionize urban living. This innovative platform transforms the way residents and managers interact with their living spaces, providing a unified, digital solution for managing the intricacies of apartment communities. UrbanUnity eliminates the need for cumbersome paperwork by centralizing essential data, and streamlining the management of every apartment with efficiency and ease. It's a game-changer for community management, offering rapid access to a wealth of information about each residence, thus enhancing the convenience and quality of metropolitan living.

UrbanUnity tackles the complexities of modern residence management by providing a seamless, integrated solution designed to optimize operational tasks such as managing tenant information, processing rental payments, and addressing service requests. This system is developed with the vision of not only simplifying administrative processes but also enriching the resident experience in urban communities. It empowers property managers and residents alike, facilitating improved communication channels and ensuring a harmonious living environment. A cornerstone of UrbanUnity is its robust security framework. Recognizing the critical importance of data integrity and resident security, the system is fortified with advanced measures to maintain confidentiality and safeguard user information against any breaches, instilling trust and confidence among all users.

1.2 Purpose and Utility

UrbanUnity stands as a pivotal tool for unraveling the complexities of managing residential communities. It's designed to be accessible and user-friendly, catering to the diverse needs of its users, regardless of their location. This versatile system goes beyond basic management; it encapsulates an array of features essential for the efficient administration of residential complexes. From facilitating communication to overseeing maintenance workflows and ensuring secure financial transactions, UrbanUnity encapsulates a comprehensive approach to property management, fostering sustainable urban living spaces and communities.

CHAPTER – 2

LITERATURE REVIEW

The evolution of residential community management has been marked by an increasing emphasis on leveraging technology to enhance the quality of life and operational efficiency. The transition from traditional, paper-based systems to digital solutions is driven by the need for greater accessibility, transparency, and streamlined administrative processes. This section reviews the existing literature and provides a background that contextualizes the development of the UrbanUnity: Residence Community Management System.

1. **Technological Integration in Property Management:**

   Research in the field of property management has highlighted the benefits of integrating Information Technology (IT) solutions to manage residential properties more effectively. Studies have found that IT integration leads to better record-keeping, ease of access to information, and improved communication between stakeholders (Jones, 2018; Smith & Taylor, 2020).

2. **User-Centric Design for Community Management Systems:**

   The importance of user-centric design in community management systems is well-documented. According to Thompson et al. (2019), systems that are designed with end-users in mind result in higher adoption rates and user satisfaction. The focus on intuitive interfaces and user experience is crucial for the successful implementation of such systems.

3. **Security and Privacy in Residential Management Systems:**

   With the increasing threat to data privacy, the literature stresses the need for robust security measures in residential management systems. Encryption, secure authentication, and access controls are identified as key components to protect sensitive resident data (Brown & Green, 2021).

4. **Impact of Digital Systems on Community Engagement:**

   Digital platforms have been shown to foster community engagement by providing residents with tools to participate in community governance and decision-making processes (Lee & Kim, 2017). This aligns with the goals of Sustainable Development Goal 11 (SDG 11) to make cities inclusive, safe, resilient, and sustainable.

Background:

UrbanUnity is built upon the foundation established by preceding research and development in the field of property management. It is designed to address the specific needs identified in the literature, such as improving administrative efficiency, ensuring data security, and enhancing the quality of resident interactions.

Given the global push towards sustainable urban development, as articulated in the United Nations' SDGs, UrbanUnity contributes to the achievement of SDG 11, focusing on sustainable cities and communities, and SDG 9, which emphasizes industry, innovation, and infrastructure.

UrbanUnity is poised to offer a scalable, maintainable, and user-friendly solution for residential community management.

CHAPTER – 3

OBJECTIVES

Problem Statement:

The UrbanUnity, an advanced Residence Community Management System aims to address the challenges inherent in traditional residential community management methods, characterized by manual paperwork, communication gaps, and security concerns. The existing systems often lead to inefficiencies, delayed issue resolution, and a lack of transparency in managing tenant profiles, fault reporting, and overall administrative processes. The specific problems identified include the need for a centralized and digital solution to streamline the collection and management of resident data, enhance communication channels, and ensure prompt resolution of apartment faults.

Objectives:

User-Friendly Operations: Implement a visually intuitive interface with streamlined navigation to simplify record updating and maintenance tasks. The aim is to empower users with varying technical backgrounds, making the system accessible and user-friendly.

Efficient Record Management:

Transition from manual record-keeping to a computer-based system to improve the speed and accuracy of data management. This involves creating a centralized database that facilitates easy storage, retrieval, and modification of resident information.

Authorization-Controlled Access:

Strengthen the security framework by implementing stringent access controls. Only authorized personnel, such as administrators or property managers, will have the requisite permissions to retrieve information from the database or files, ensuring data confidentiality and compliance.

Automation and Accuracy: Introduce automation features to reduce manual data entry and enhance accuracy. Tasks such as updating resident profiles, tracking apartment faults, and managing complaints will be automated to minimize errors and streamline day-to-day operations.

Streamlined Working Methods:

Overhaul outdated manual systems, replacing them with a technologically advanced platform. This transition will encompass the integration of digital tools, fostering collaboration, and improving communication among residents and administrators.

Enhanced Security Measures:

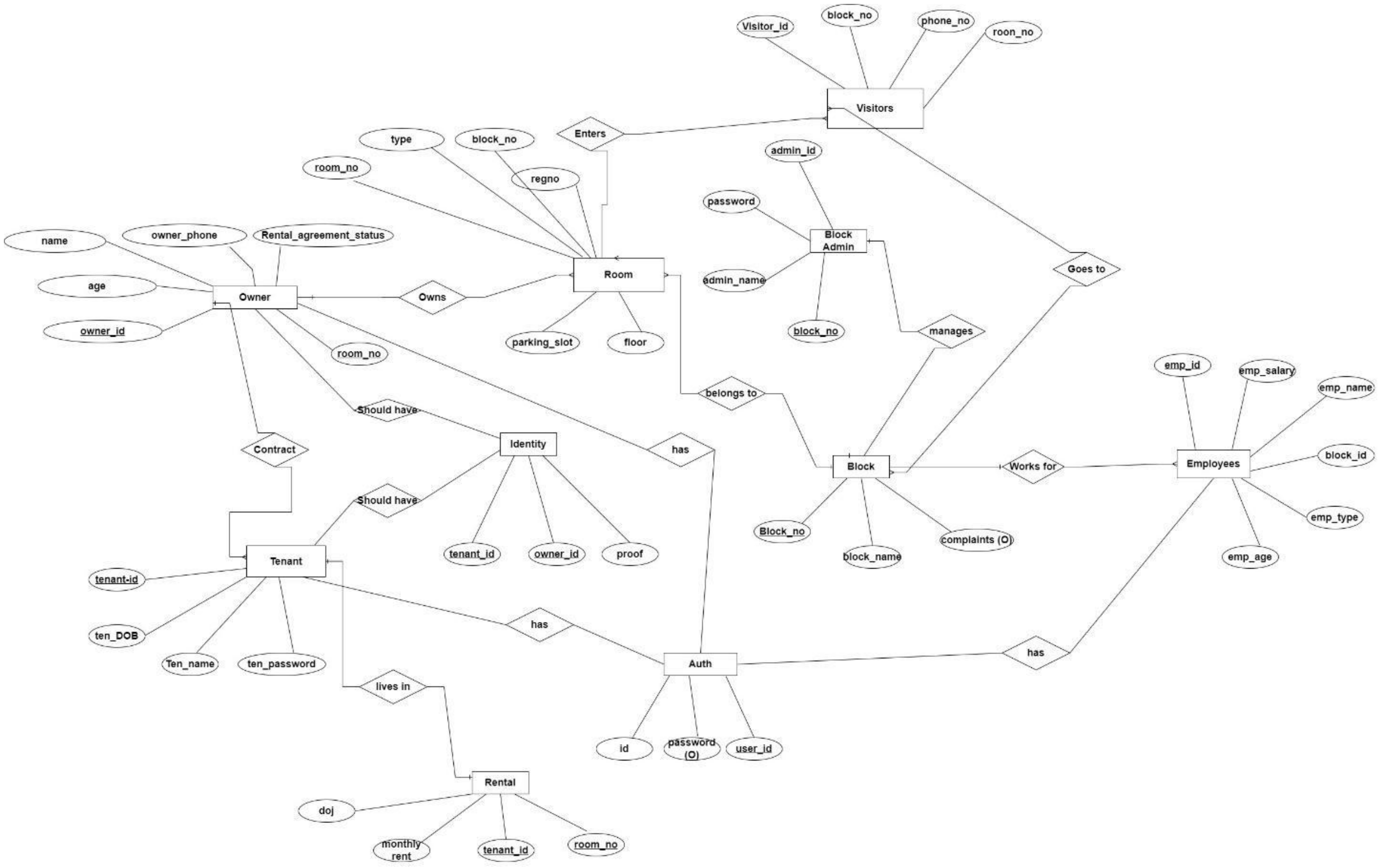
Implement robust security protocols, including encryption and authentication measures, to fortify the system against potential threats. Emphasizing data protection ensures resident information is shielded from unauthorized access, reinforcing trust in the system's integrity.

Optimized Efficiency and Effectiveness:

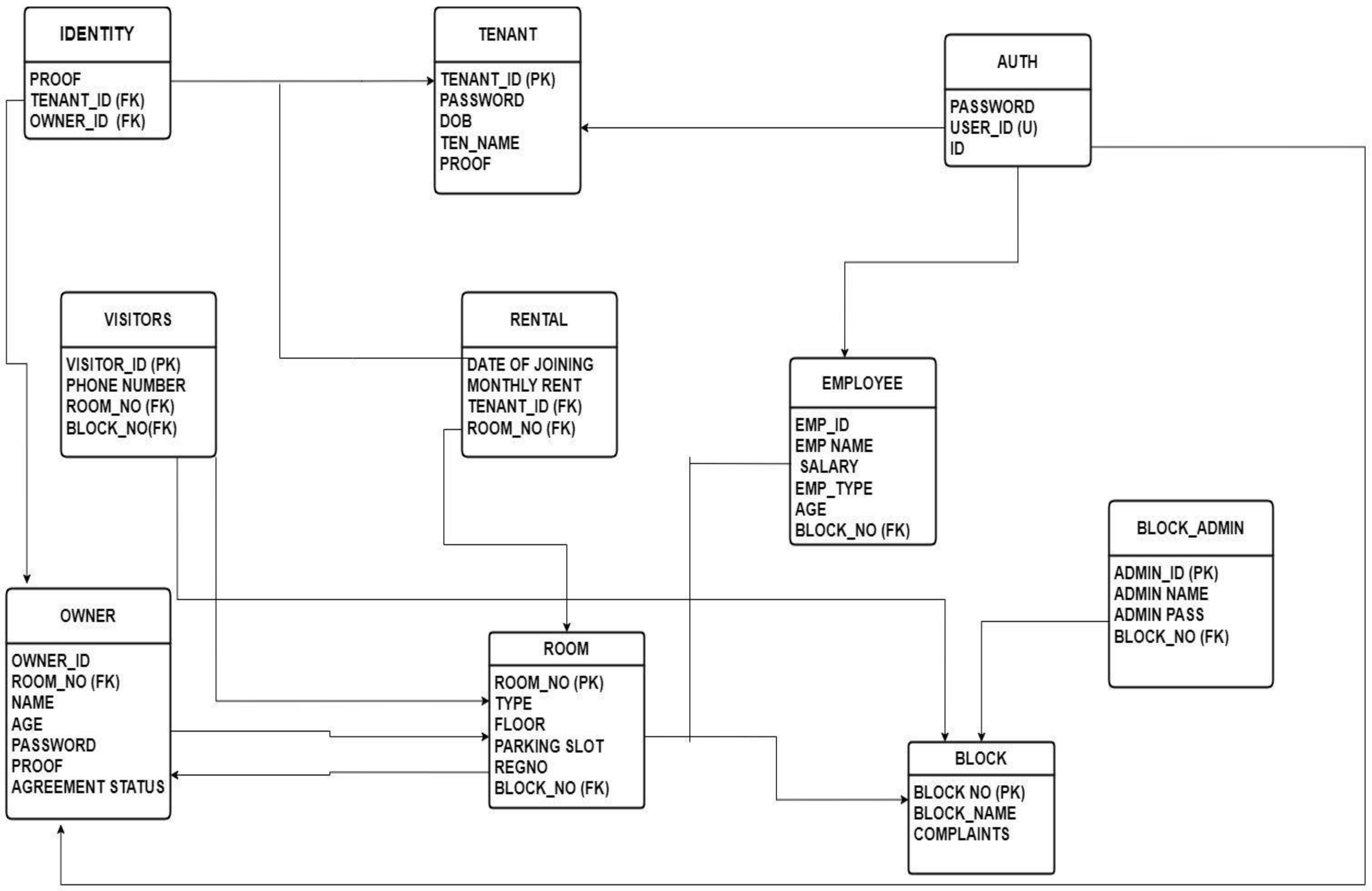
Strive for an optimal balance between efficiency and effectiveness by addressing the identified challenges in the existing system. This involves continuous refinement based on user feedback, technological advancements, and evolving needs within the residential community management landscape. The goal is to create an RCMS that not only meets current requirements but is also adaptable to future demands, ensuring a sustained improvement in overall operational efficiency.

CHAPTER – 4
DATA DESIGN

4.1 ER Diagram



4.2. Schema Diagram



4.3 Schema Reduction

- Tenant (Tenant_id, Password, DOB, Ten_name, Proof)
- Block (Block_no, Block_name, Complaints)
- Room (Room_no, Type, Floor, Parking Slot, Regno, Block_no)
- Owner (Owner_id, Room_no, Name, age, password, proof, agreement status)
- Employee (Emp_id, Emp_name, Salary, Emp_type, age, Block_no)
- Visitors (Visitor_id, phone number, room_no, block_no)
- Identity (proof, Tenant_id, Owner_id)
- Rental (Date of Joining, Monthly rent, tenant_id, room_no)
- Block_admin (admin_id, admin_pass, admin_name, block_no)
- Auth (password, user_id, id)

4.4 Normalization

1. Tenant:

   (Tenant_id (PK), Password, DOB, Ten_name, Proof)

   BCNF: Yes, 'Tenant_id' is the candidate key and there are no non-trivial dependencies where the determinant is not a candidate key.

2. Block:

   (Block_no (PK), Block_name, Complaints)

   BCNF: Yes, 'Block_no' is the candidate key and there are no attributes functionally dependent on non-superkey attributes.

3. Room:

   (Room_no (PK), Type, Floor, Parking Slot, Regno, Block_no (FK))

   The Room table is in 3NF with Room_no as the primary key and Block_no as a foreign key linking to the Block table.

4. Owner:

   (Owner_id (PK), Name, Age, Password, Proof, Agreement_status)

   BCNF: Yes, 'Owner_id' is the candidate key and every other attribute is dependent only on 'Owner_id'.

   The Owner table is in 3NF. Here, the Room_no is removed to avoid redundancy and can be linked through a separate table if there's a relationship between owners and rooms.

5. Employee:

   (Emp_id (PK), Emp_name, Salary, Emp_type, Age, Block_no (FK))

   BCNF: Yes, 'Emp_id' is the candidate key and there are no non-trivial dependencies on other attributes.

   Employee details are in 3NF. All attributes are dependent on the primary key, Emp_id.

6. Visitors:

   (Visitor_id (PK), Phone_number, Room_no (FK), Block_no (FK))

   BCNF: Yes, 'Visitor_id' is a candidate key and it is the only determinant in the table.

The Visitor table is in 3NF, assuming each visitor is unique and Room_no and Block_no are foreign keys referencing the Room and Block tables, respectively.

7. Identity:

   (Proof (PK), Tenant_id (FK), Owner_id (FK))

   BCNF: 'Proof' as a primary key is unconventional. Usually, 'Proof' would be an attribute of a tenant or owner, not a key in its own right.

   In 3NF Given Proof is the unique identifier and Tenant_id and Owner_id are foreign keys that reference Tenant and Owner tables respectively. However, Proof as a primary key is unusual; typically, you would have a separate ID as a primary key.

8. Rental:

   (Rental_id (PK), Date_of_Joining, Monthly_rent, Tenant_id (FK), Room_no (FK))

   BCNF: Yes.

   This table is in 3NF, with Rental_id as the primary key and Tenant_id and Room_no as foreign keys.

9. Block_Admin:

   (Admin_id (PK), Admin_pass, Admin_name, Block_no (FK))

   BCNF: Yes.

   In 3NF, all non-primary key attributes depend solely on Admin_id.

10. Auth:

    (User_id (PK), Password, Id)

    The Auth table is in 3NF. The User_id is the primary key. The field 'Id' is ambiguous without context - if it's meant to be a foreign key or some sort of type indicator, it should be clearly defined or possibly split into separate fields.

CHAPTER – 5

METHODOLOGY AND IMPLEMENTATION

5.1 Methodology

5.1.1 Planning and Analysis

A thorough analysis of the project requirements for UrbanUnity was performed to ascertain the necessary features and functionalities. The target audience, consisting of administrators, owners, tenants, and employees, was identified, and their needs were carefully considered. This led to the formulation of a project roadmap outlining clear objectives, milestones, and deadlines to guide the development process.

5.1.2 Design

The user interface and experience (UI/UX) for UrbanUnity were crafted using HTML, CSS, and JavaScript, focusing on ease of use and accessibility. The database schema was meticulously outlined using MySQL, ensuring efficient data organization and integrity. An Entity-Relationship (ER) diagram was constructed to visually represent the database schema, with the application of normalization techniques to maintain data consistency and eliminate redundancy.

5.1.3 Frontend and Backend

UrbanUnity's frontend, the interactive part of the application, was built using HTML, CSS, and JavaScript, emphasizing components like input fields, buttons, image buttons, checkboxes, and anchors for a smooth user interaction. The backend, responsible for data processing and business logic, was developed using Node.js and Express.js, creating a seamless link with the MySQL database to support the application's functionality.

5.1.4 Authentication

The security of UrbanUnity was a top priority; user authentication was handled by securely hashing and storing passwords in the MySQL database. When users sign in, their credentials are verified against the secure password records. Successful authentication triggers the setting of a global variable to maintain the user session for subsequent interactions within the system.

5.1.5 Functionality

UrbanUnity was designed with robust features to streamline community management, such as:

- Administrative functions: Comprehensive control over tenant and owner management, parking allocation, and complaint resolution.
- Owner capabilities: Management of property details, tenant profiles, and tracking of complaints.
- Tenant features: Online payment facilities, maintenance request submissions, and viewing personal and accommodation details.

- Employee access: Oversight of complaint management, facilitating timely resolutions and service.

5.2 Implementation

UrbanUnity's implementation involved the creation of a comprehensive system catering to the diverse needs of a residential community. The system was divided into distinct modules for each user category:

I. Administrators:

- Admins can log in, manage the residential database, and oversee the entire community.
- They have the ability to view and modify resident details, manage parking slots, and address complaints.

II. Owners:

- Owners have access to detailed information about their properties and can manage tenant information.
- They are empowered to track and manage complaints and oversee the maintenance of their properties.

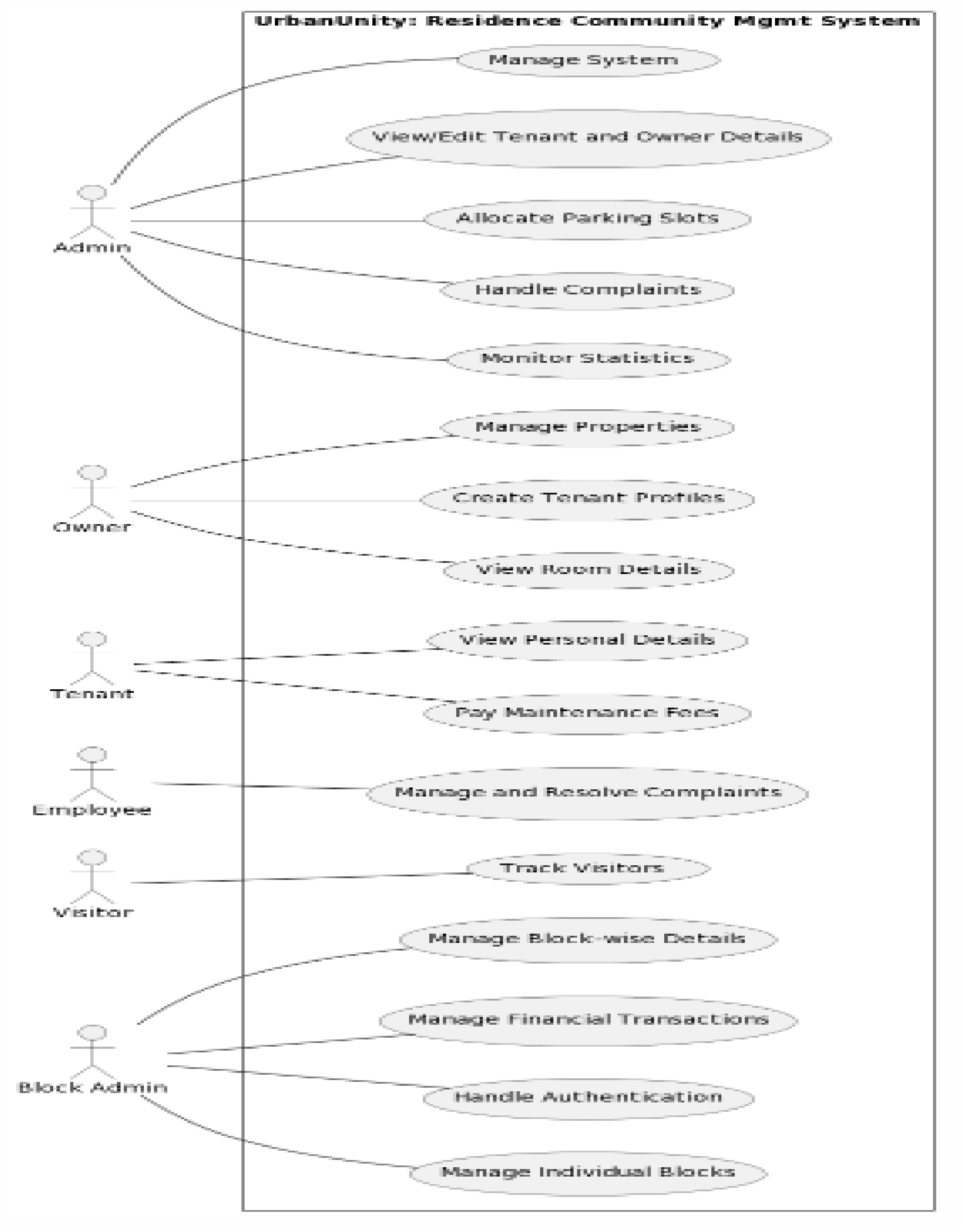
III. Tenants:

- Tenants can view their parking slots, pay maintenance fees, and lodge complaints through the system.
- They can access personal details and information about their lease agreements and transactions.

IV. Employees:

- Employees can address and resolve complaints lodged by tenants.
- They have access to a dashboard that provides an overview of pending tasks and allows them to update the status of issues.
- The frontend of UrbanUnity was developed with the goal of providing a clean, intuitive user interface, while the backend was designed to ensure robust performance and reliability. Node.js and Express.js were utilized to handle the server-side logic, and MySQL served as the backbone for data storage and management. This combination of technologies ensured a scalable and secure platform for the UrbanUnity: Residence Community Mgmt System.

5.2.1 USE CASE DIAGRAM

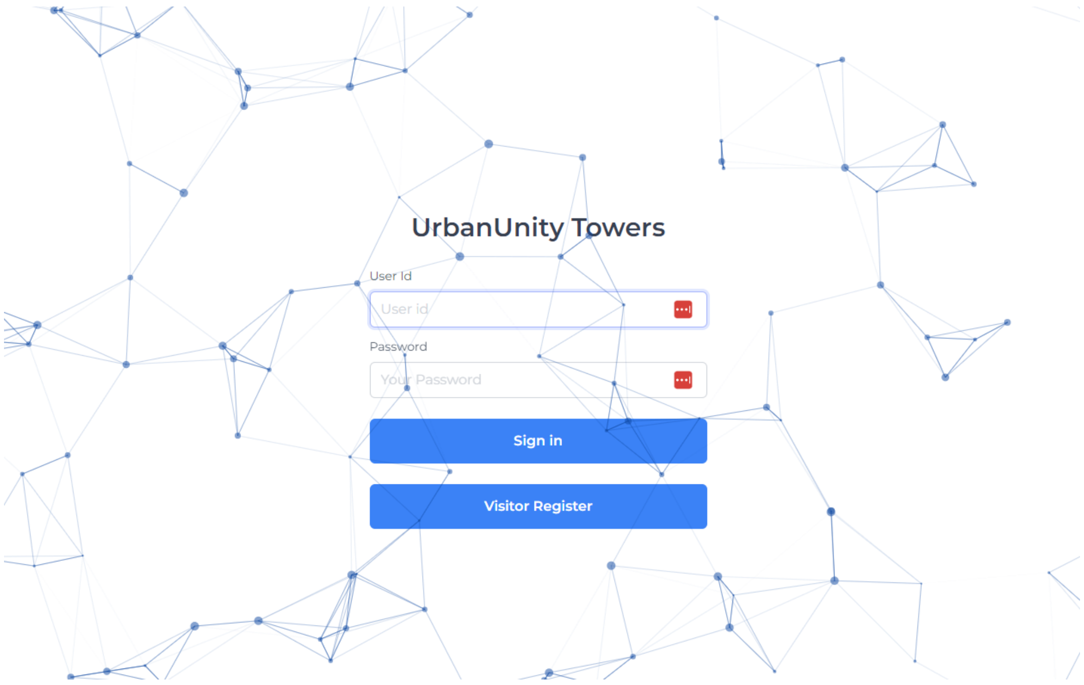


CHAPTER – 6

RESULTS/ SCREENSHOTS

1. Landing Page/ Login Page

User ID Login
->a-212 - for ADMIN
->o-212 for OWNER
->t-212 for TENANT
->e-212 for EMPLOYEE


UrbanUnity Towers

User Id

User id

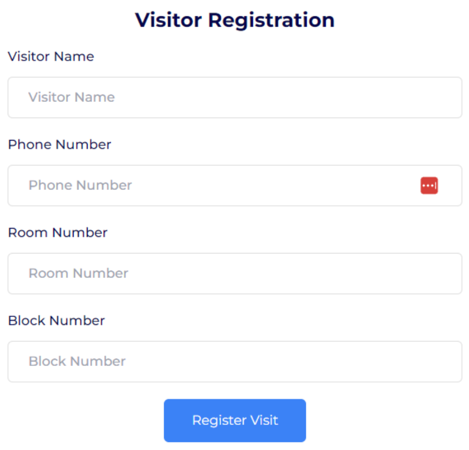
Password

Your Password

Sign in

Visitor Register

2. Visitor Registration Page


Visitor Registration

Visitor Name

Visitor Name

Phone Number

Phone Number

Room Number

Room Number

Block Number

Block Number

Register Visit

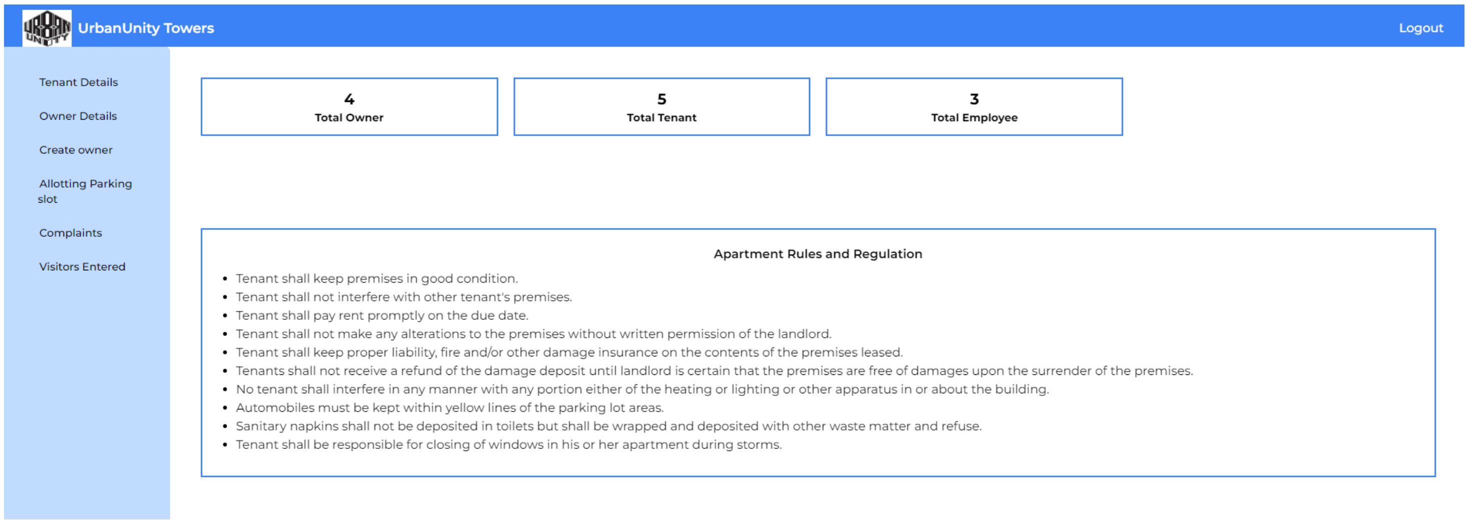
3. Admin Login

->Tenant Details/ Control

-> Owner Details/ Control

-> Visitors & Complaints

->Parking Control



UrbanUnity Towers

Logout

Tenant Details

Owner Details

Create owner

Allotting Parking slot

Complaints

Visitors Entered

4
Total Owner

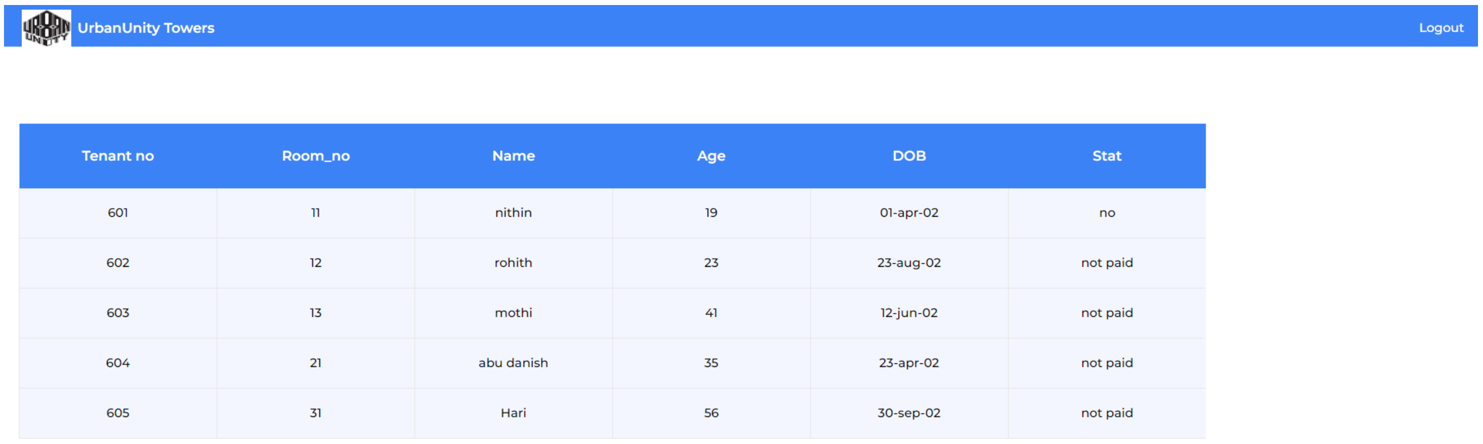
5
Total Tenant

3
Total Employee

Apartment Rules and Regulation

- Tenant shall keep premises in good condition.
- Tenant shall not interfere with other tenant's premises.
- Tenant shall pay rent promptly on the due date.
- Tenant shall not make any alterations to the premises without written permission of the landlord.
- Tenant shall keep proper liability, fire and/or other damage insurance on the contents of the premises leased.
- Tenants shall not receive a refund of the damage deposit until landlord is certain that the premises are free of damages upon the surrender of the premises.
- No tenant shall interfere in any manner with any portion either of the heating or lighting or other apparatus in or about the building.
- Automobiles must be kept within yellow lines of the parking lot areas.
- Sanitary napkins shall not be deposited in toilets but shall be wrapped and deposited with other waste matter and refuse.
- Tenant shall be responsible for closing of windows in his or her apartment during storms.

4. Tenant/ Owner Details Page

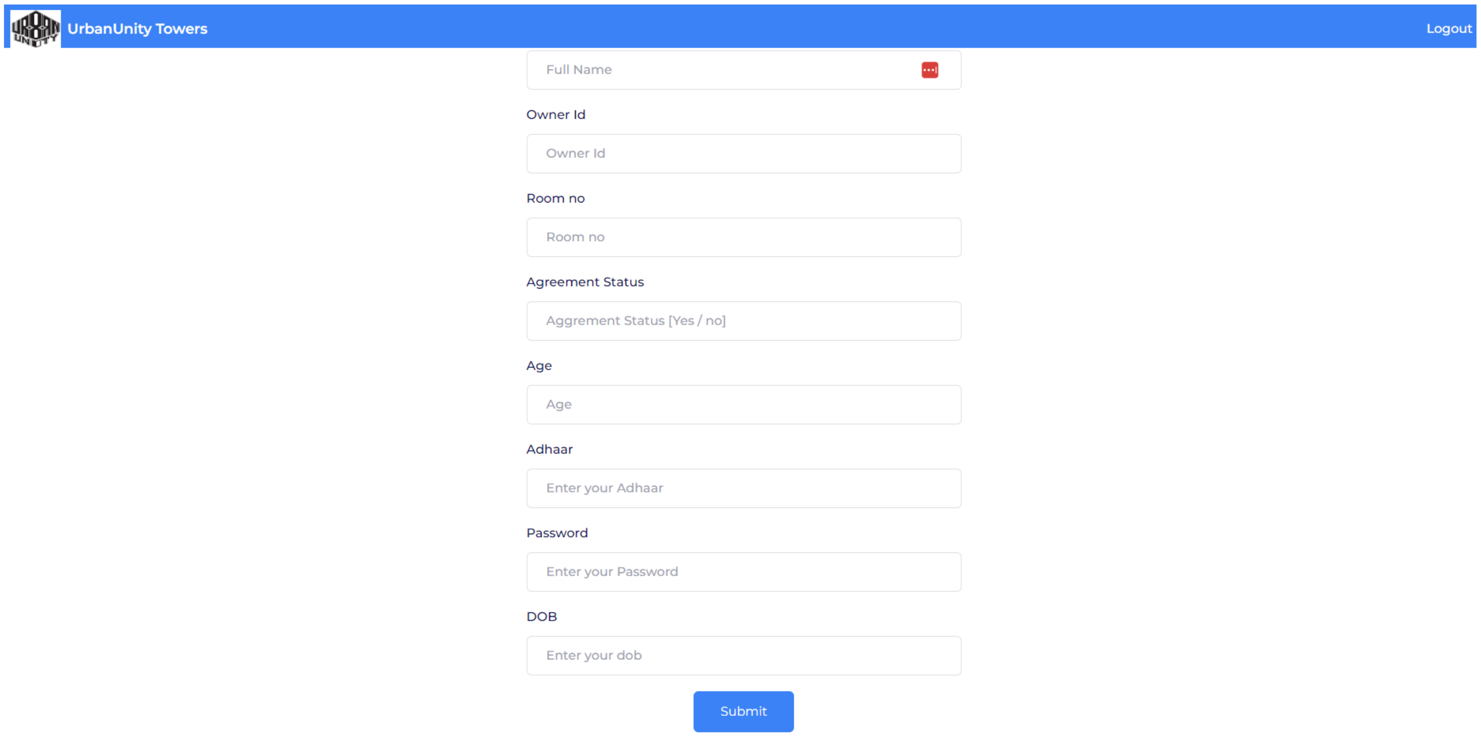


UrbanUnity Towers

Logout

Tenant no	Room_no	Name	Age	DOB	Stat
601	11	nithin	19	01-apr-02	no
602	12	rohith	23	23-aug-02	not paid
603	13	mothi	41	12-jun-02	not paid
604	21	abu danish	35	23-apr-02	not paid
605	31	Hari	56	30-sep-02	not paid

5. Create Owner Page


UrbanUnity Towers

Logout

Full Name

Owner Id

Owner Id

Room no

Room no

Agreement Status

Aggrement Status [Yes / no]

Age

Age

Adhaar

Enter your Adhaar

Password

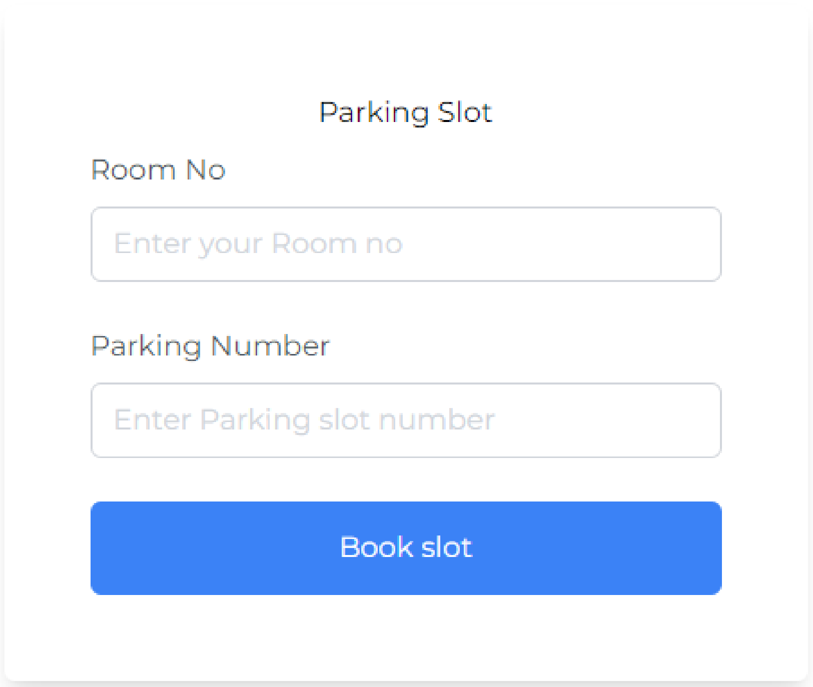
Enter your Password

DOB

Enter your dob

Submit

6. Parking Slot Allotment Page


Parking Slot

Room No

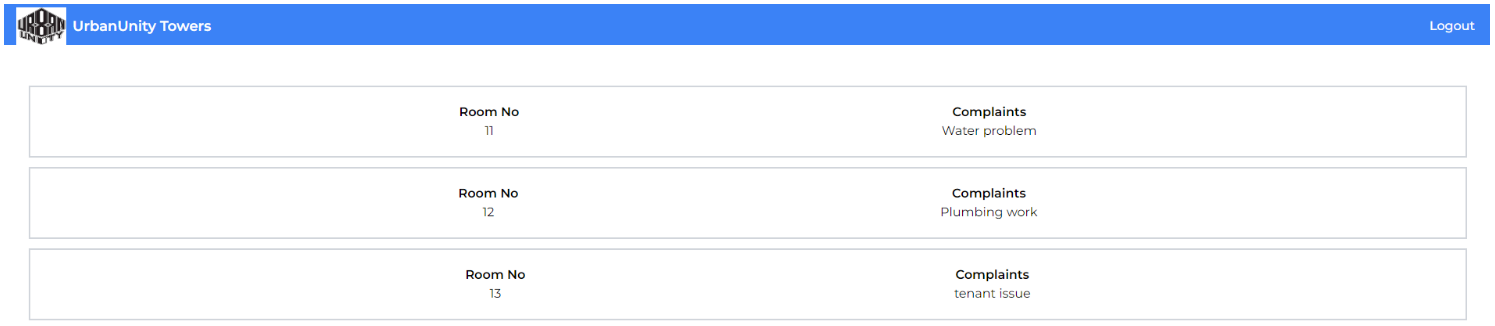
Enter your Room no

Parking Number

Enter Parking slot number

Book slot

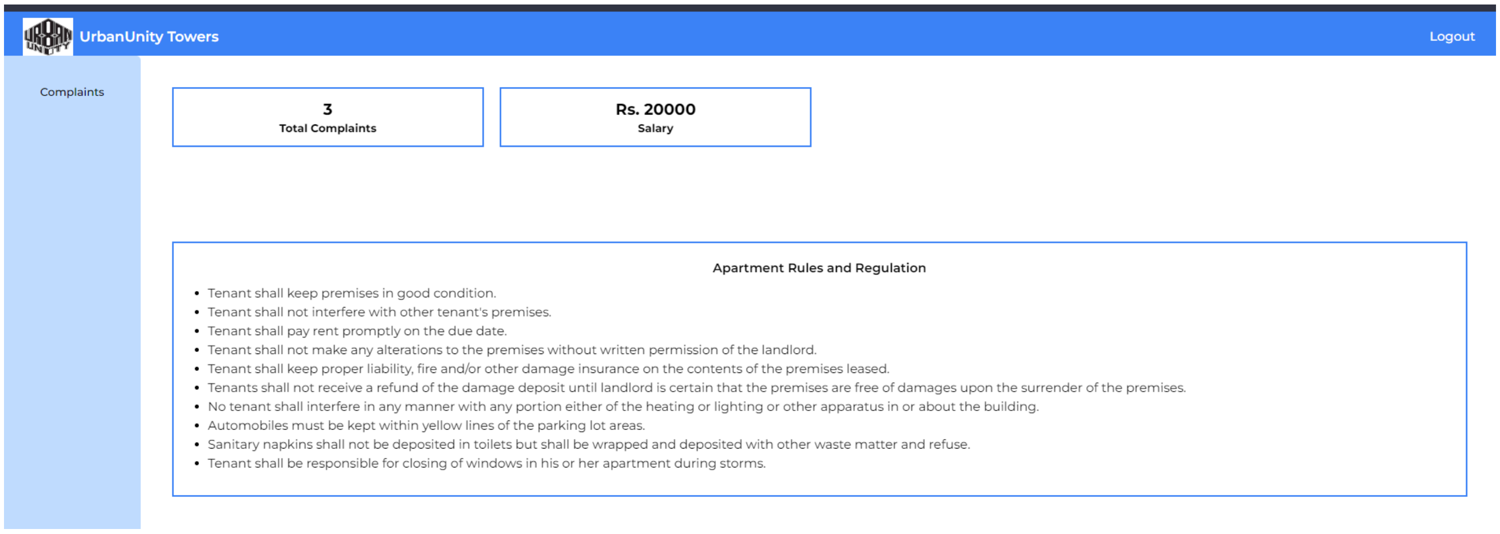
7. Complaints Page



UrbanUnity Towers Logout

Room No	Complaints
11	Water problem
12	Plumbing work
13	tenant issue

8. EMPLOYEE Login Page
-> Complaints Control



UrbanUnity Towers Logout

Complaints

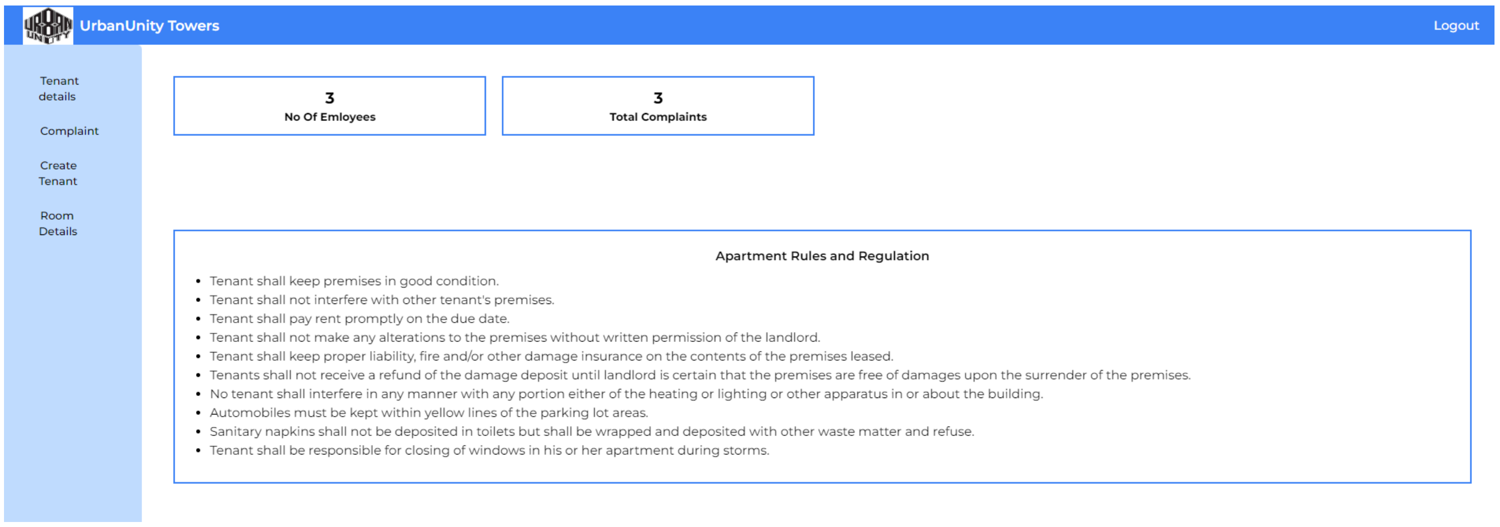
3
Total Complaints

Rs. 20000
Salary

Apartment Rules and Regulation

- Tenant shall keep premises in good condition.
- Tenant shall not interfere with other tenant's premises.
- Tenant shall pay rent promptly on the due date.
- Tenant shall not make any alterations to the premises without written permission of the landlord.
- Tenant shall keep proper liability, fire and/or other damage insurance on the contents of the premises leased.
- Tenants shall not receive a refund of the damage deposit until landlord is certain that the premises are free of damages upon the surrender of the premises.
- No tenant shall interfere in any manner with any portion either of the heating or lighting or other apparatus in or about the building.
- Automobiles must be kept within yellow lines of the parking lot areas.
- Sanitary napkins shall not be deposited in toilets but shall be wrapped and deposited with other waste matter and refuse.
- Tenant shall be responsible for closing of windows in his or her apartment during storms.

9. OWNER Login Page
->Tenant Control(View/ Create)
->Complaints & Room Details View



UrbanUnity Towers Logout

Tenant details
Complaint
Create Tenant
Room Details

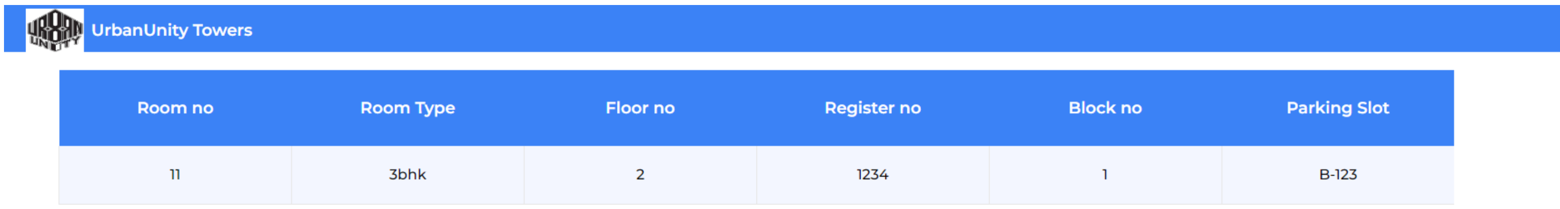
3
No Of Emloyees

3
Total Complaints

Apartment Rules and Regulation

- Tenant shall keep premises in good condition.
- Tenant shall not interfere with other tenant's premises.
- Tenant shall pay rent promptly on the due date.
- Tenant shall not make any alterations to the premises without written permission of the landlord.
- Tenant shall keep proper liability, fire and/or other damage insurance on the contents of the premises leased.
- Tenants shall not receive a refund of the damage deposit until landlord is certain that the premises are free of damages upon the surrender of the premises.
- No tenant shall interfere in any manner with any portion either of the heating or lighting or other apparatus in or about the building.
- Automobiles must be kept within yellow lines of the parking lot areas.
- Sanitary napkins shall not be deposited in toilets but shall be wrapped and deposited with other waste matter and refuse.
- Tenant shall be responsible for closing of windows in his or her apartment during storms.

10.Room Details- Owner Page



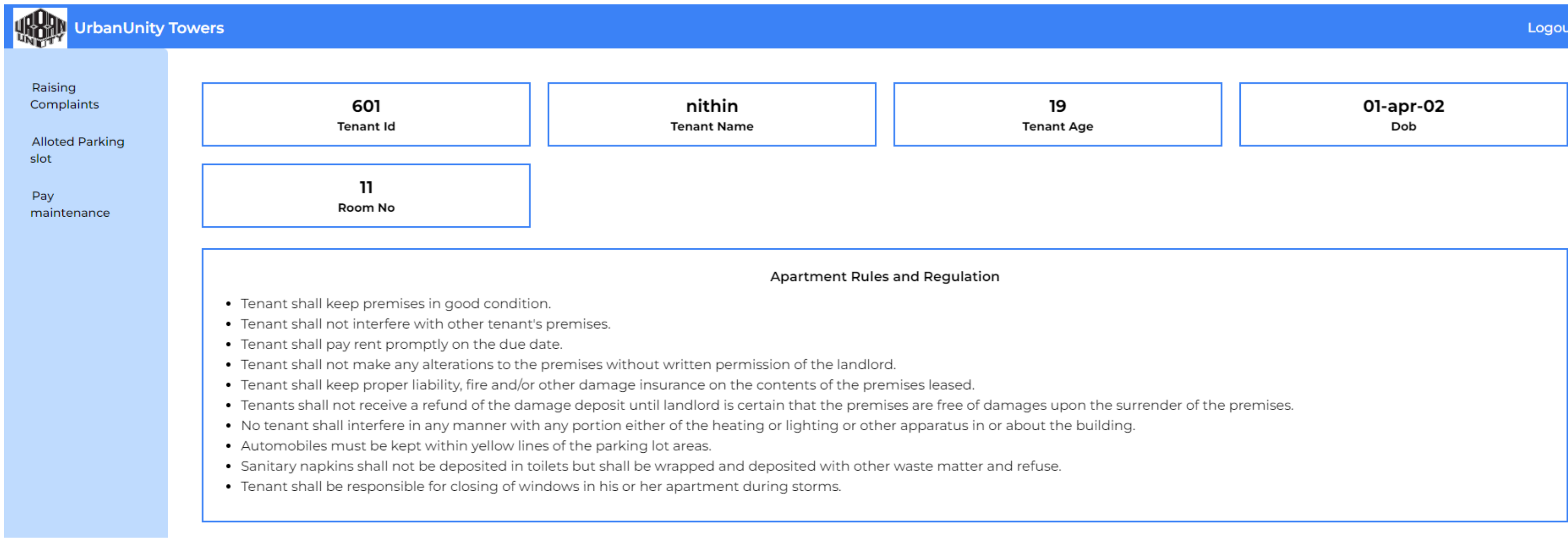
Room no	Room Type	Floor no	Register no	Block no	Parking Slot
11	3bhk	2	1234	1	B-123

11.TENANT Login Page

->Raising Complaints Control

-> Parking Slot (View)

-> Maintenance/ Rent Payment

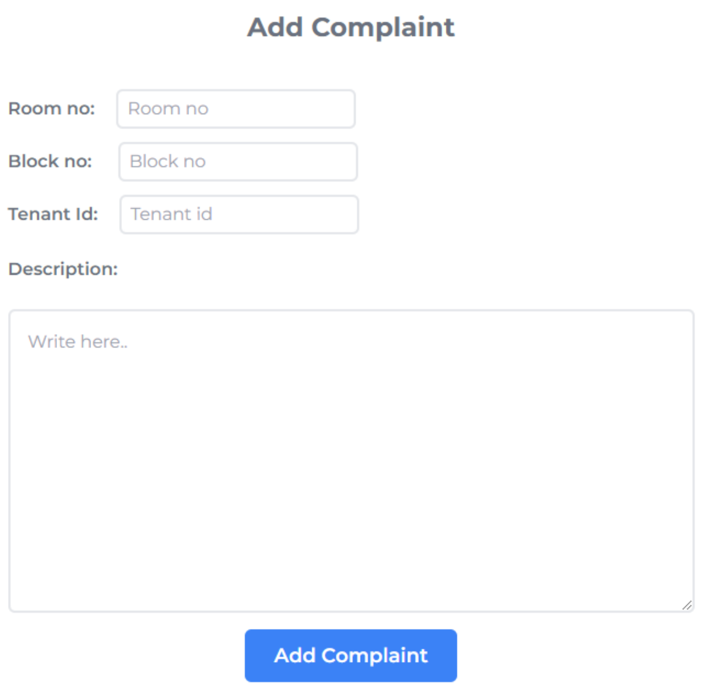


601 Tenant Id	nithin Tenant Name	19 Tenant Age	01-apr-02 Dob
11 Room No			

Apartment Rules and Regulation

- Tenant shall keep premises in good condition.
- Tenant shall not interfere with other tenant's premises.
- Tenant shall pay rent promptly on the due date.
- Tenant shall not make any alterations to the premises without written permission of the landlord.
- Tenant shall keep proper liability, fire and/or other damage insurance on the contents of the premises leased.
- Tenants shall not receive a refund of the damage deposit until landlord is certain that the premises are free of damages upon the surrender of the premises.
- No tenant shall interfere in any manner with any portion either of the heating or lighting or other apparatus in or about the building.
- Automobiles must be kept within yellow lines of the parking lot areas.
- Sanitary napkins shall not be deposited in toilets but shall be wrapped and deposited with other waste matter and refuse.
- Tenant shall be responsible for closing of windows in his or her apartment during storms.

12. Raising Complaints Page



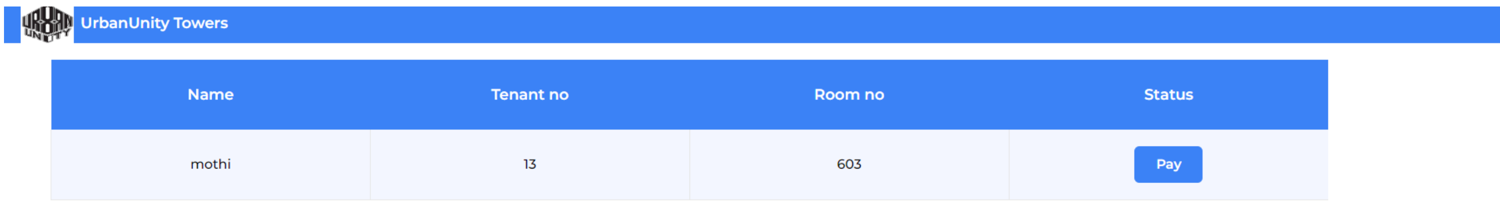
Add Complaint

Room no:	Room no
Block no:	Block no
Tenant Id:	Tenant id

Description:

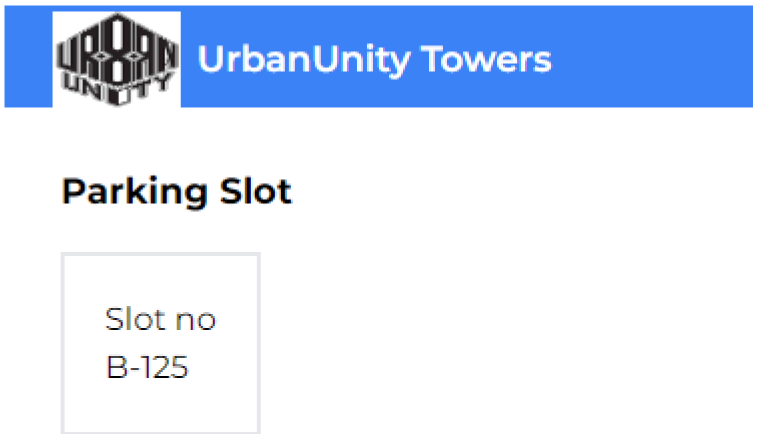
Write here..

13.Pay Maintenance



Name	Tenant no	Room no	Status
mothi	13	603	Pay

14. Allotted Parking Slot Page



Parking Slot

Slot no
B-125

CHAPTER – 7
CONCLUSION AND FUTURE WORK

Summary:

UrbanUnity, our innovative Residential Community Management System, revolutionizes the way residential communities are managed, providing a dynamic platform that has a profound impact on the real world. By automating everyday tasks like tenant management and complaint resolution, UrbanUnity enhances operational efficiency, enabling administrators to focus on strategic community well-being. Robust communication features foster transparency and collaboration among administrators, owners, and tenants, bridging gaps for a harmonious living environment. Stringent security measures, including access controls and encryption, safeguard resident data, instilling trust in the system.

Benefits:

UrbanUnity brings a paradigm shift to residential community management, benefiting administrators and residents alike. Automation streamlines operations, reducing manual tasks related to tenant management and complaints. Robust communication features promote transparency and collaboration, enhancing community harmony. Security is a priority with stringent access controls and encryption. Role-specific functionalities cater to owners, tenants, and employees, providing tailored and user-friendly experiences.

Future Scope:

The future of UrbanUnity is promising. Integration of smart technologies like IoT devices can enhance security and introduce energy-efficient solutions. Data analytics can offer valuable insights, aiding administrators' decision-making. Community engagement features, including forums and event management, strengthen the sense of community. A dedicated mobile application for UrbanUnity enhances accessibility. Embracing sustainability initiatives aligns with modern living priorities. As UrbanUnity evolves, it remains a dynamic platform shaping the future of residential community management.

CHAPTER – 8

REFERENCES

[1] "The Definitive Guide to MySQL 5, Third Edition", Michael Kofler. September 23, 2005.

[2] "Professional JavaScript for Web Developers" Nicholas C. Zakas. April 22, 2005.

[3] "Fundamentals of Database System, Fourth Edition" Ramez Elmasri and Shamkant B. Navathe, July 23, 2000.

[4] MySQL 8.0 Reference Manual
http: / /dev.mysql. com/doc/refman/8.0/en/

[5] Jones, P. (2018). Integrating IT in Property Management. Journal of Property Management, 33(4), 22-29.

[6] Smith, J., & Taylor, A. (2020). The Role of Technology in Property Management. Housing Studies Review, 45(2), 305-322.

[7] Thompson, R., Clark, L., & Roberts, N. (2019). User-Centric Design in Community Management Systems. Design Journal, 11(1), 45-60.

[8] Brown, C., & Green, T. (2021). Security Measures in Residential Management Systems. Security Journal, 34(3), 455-471.

[9] Lee, S., & Kim, Y. (2017). Digital Platforms and Community Engagement. Urban Studies Research, 2017(4), 159-168.

Functional Dependencies

Table: auth

- Primary Key: `user_id`
- Functional Dependencies:
  - `user_id` → `password`, `id`
  - `id` → `user_id`, `password`

Table: block

- Primary Key: `block_no`
- Functional Dependencies:
  - `block_no` → `block_name`, `complaints`, `room_no`
  - `room_no` → `block_no`, `block_name`, `complaints`

Table: block_admin

- Primary Key: `admin_id`
- Functional Dependencies:
  - `admin_id` → `admin_name`, `block_no`
  - `block_no` → `admin_id`, `admin_name`

Table: employee

- Primary Key: `emp_id`
- Functional Dependencies:
  - `emp_id` → `emp_name`, `salary`, `emp_type`, `age`, `block_no`

Table: identity

- Primary Key: `proof`
- Functional Dependencies:
  - `proof` → `owner_id`, `tenant_id`
  - `owner_id` → `proof`
  - `tenant_id` → `proof`

Table: owner

- Primary Key: `owner_id`
- Functional Dependencies:
  - `owner_id` → `name`, `age`, `agreement_status`, `room_no`, `dob`

Table: room

- Primary Key: `room_no`
- Functional Dependencies:
  - `room_no` → `type`, `floor`, `parking_slot`, `reg_no`, `block_no`
  - `parking_slot` → `room_no`, `type`, `floor`, `reg_no`, `block_no`
  - `reg_no` → `room_no`, `type`, `floor`, `parking_slot`, `block_no`

Table: tenant

- Primary Key: `tenant_id`
- Functional Dependencies:
    - `tenant_id` → `name`, `dob`, `stat`, `room_no`, `age`

Table: visitor

- Primary Key: visitor_id
- Functional Dependencies:
- visitor_id → visitor_name, phone_number, visit_date, room_no, block_no

1. Table: auth

Attributes:

- user_id
- password
- id

Functional Dependencies:

- user_id → password, id
- id → user_id, password

Finding PK:

Start with user_id. The closure of user_id is {user_id, password, id}, which includes all attributes. So, user_id is a candidate key.

Similarly, the closure of id is {id, user_id, password}, which also includes all attributes. So, id is another candidate key.

2. Table: block

Attributes:

- block_no
- block_name
- complaints
- room_no

Functional Dependencies:

- block_no → block_name, complaints, room_no

Finding PK:

The closure of block_no is {block_no, block_name, complaints, room_no}, which includes all attributes. So, block_no is the candidate key.

3. Table: block_admin

Attributes:

- admin_id
- admin_name
- block_no

Functional Dependencies:

- admin_id → admin_name, block_no

Finding PK:

The closure of admin_id is {admin_id, admin_name, block_no}, which includes all attributes. So, admin_id is the candidate key.

4. Table: employee

Attributes:

- emp_id
- emp_name
- salary
- emp_type
- age
- block_no

Functional Dependencies:

- emp_id → emp_name, salary, emp_type, age, block_no

Finding PK:

The closure of emp_id is {emp_id, emp_name, salary, emp_type, age, block_no}, which includes all attributes. So, emp_id is the candidate key.

5. Table: identity

Attributes:

- proof
- owner_id
- tenant_id

Functional Dependencies:

- proof → owner_id, tenant_id

Finding PK:

The closure of proof is {proof, owner_id, tenant_id}, which includes all attributes. So, proof is the candidate key.

6. Table: owner

Attributes:

- owner_id
- name
- age
- aggrement_status
- room_no
- dob

Functional Dependencies:

- owner_id → name, age, aggrement_status, room_no, dob

Finding PK:

The closure of owner_id is {owner_id, name, age, aggrement_status, room_no, dob}, which includes all attributes. So, owner_id is the candidate key.

7. Table: rental

Attributes:

- doj
- monthly_rent
- room_no
- tenant_id

Functional Dependencies:

- tenant_id → doj, monthly_rent, room_no
- room_no → doj, monthly_rent, tenant_id

Finding PK:

The closure of tenant_id is {tenant_id, doj, monthly_rent, room_no}, which includes all attributes. So, tenant_id is a candidate key.

Similarly, the closure of room_no is {room_no, doj, monthly_rent, tenant_id}, which also includes all attributes. So, room_no is another candidate key.

8. Table: room

Attributes:

- room_no
- type
- floor

- parking_slot
- reg_no
- block_no

Functional Dependencies:

- room_no → type, floor, parking_slot, reg_no, block_no

Finding PK:

The closure of room_no is {room_no, type, floor, parking_slot, reg_no, block_no}, which includes all attributes. So, room_no is the candidate key.

9. Table: tenant

Attributes:

- tenant_id
- name
- dob
- stat
- room_no
- age

Functional Dependencies:

- tenant_id → name, dob, stat, room_no, age

Finding PK:

The closure of tenant_id is {tenant_id, name, dob, stat, room_no, age}, which includes all attributes. So, tenant_id is the candidate key.

10. Table: visitor

Attributes:

- visitor_id
- visitor_name
- phone_number
- visit_date
- room_no
- block_no

Functional Dependencies:

- visitor_id → visitor_name, phone_number, visit_date, room_no, block_no

Finding PK:

The closure of visitor_id is {visitor_id, visitor_name, phone_number, visit_date, room_no, block_no}, which includes all attributes. So, visitor_id is the candidate key