**Housing Management System**

*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*

**Computer and Communication Engineering**

*by*

**Shivam Mishra, Nishka Kulshrestha, Yatharth Srivastava**

**210953042, 210953068 2109530208**

*Under the guidance of*

Dr. Sumith N Dr. Raghavendra Achar

Associate Professor Associate Professor

Department of I&CT Department of I&CT

Manipal Institute of Technology Manipal Institute of Technology

****Manipal, Karnataka, India Manipal, Karnataka, India

**March 2023**

**ABSTRACT**

**In today's technology-driven world, efficient data management plays a crucial role in various domains, including businesses, organizations, and research institutions. This project aims to develop an Housing Management System with a C# frontend to streamline data storage, retrieval, and manipulation processes.**

**The system utilizes a robust database management system, such as Microsoft SQL Server, to store and organize vast amounts of structured data. The C# frontend serves as an intuitive user interface, enabling users to interact with the database system seamlessly.**

**The project focuses on the following key components:**

**1- Database Design and Management: A well-structured and optimized database schema is designed to meet the specific requirements of the application. It includes tables, relationships, and appropriate constraints to ensure data integrity.**

**2- Data Manipulation: The system provides functionalities for efficient data manipulation operations, such as inserting, updating, deleting, and querying records. Users can perform complex queries and generate custom reports based on their specific needs.**

**3- Data Security and Privacy: Adequate security measures are implemented to protect sensitive data, including authentication, authorization, and encryption techniques. Access control mechanisms ensure that only authorized users can interact with the database.**

**4- User-Friendly Frontend: The C# frontend provides an intuitive and user-friendly interface for users to interact with the database. It includes features such as forms, menus, and controls that enable easy data entry, retrieval, and visualization.**

**This project aligns with the following Sustainable Development Goal (SDG):**

**[SDG]: Sustainable Cities and Communities**

**Keywords: Database Management, C# Frontend, Data Manipulation, Data Security, Data Privacy, User Interface, Integration, Sustainable Development Goal (SDG), Sustainable Cities and Communities.**

**ACM Taxonomy Keywords: Database Management Systems, User Interfaces, Data Security and Privacy, Data Manipulation, Database Design, Information Systems.**

**Table of Contents**

1. **Introduction**
2. **Literature Survey / Background**
3. **Objectives/Problem Statement**
4. **Data design (ER Diagram, Reduction, Schema Diagram, Normalization with suitable justifications)**
5. **Methodology (Implementation Details with block diagram to explain the project in detail. Do not put code here.)**
6. **Results**
7. **Conclusion and Future Work**
8. **References**

**INTRODUCTION**

**Objective:**

 The objective of the **“Housing Management System”** is to handle the entire activity of a Housing System. The software keeps track of all the information about the apartments, owners, tenants, and their complete details along with total number of rooms available. The user will find it easy in this automated system rather than using the manual writing system. The system contains a database where all the information will be stored safely. The system will be user-friendly, saves time and reduce the likelihood of errors or oversights.

**Scope:**

This project is helpful to track all the details of Owner, Tenant, Employee and the booking status of apartments. This will eradicate the problem of manual entry of the track records by providing an interface for managing it. The software will be able to handle all the necessary information.

**Project Description:**

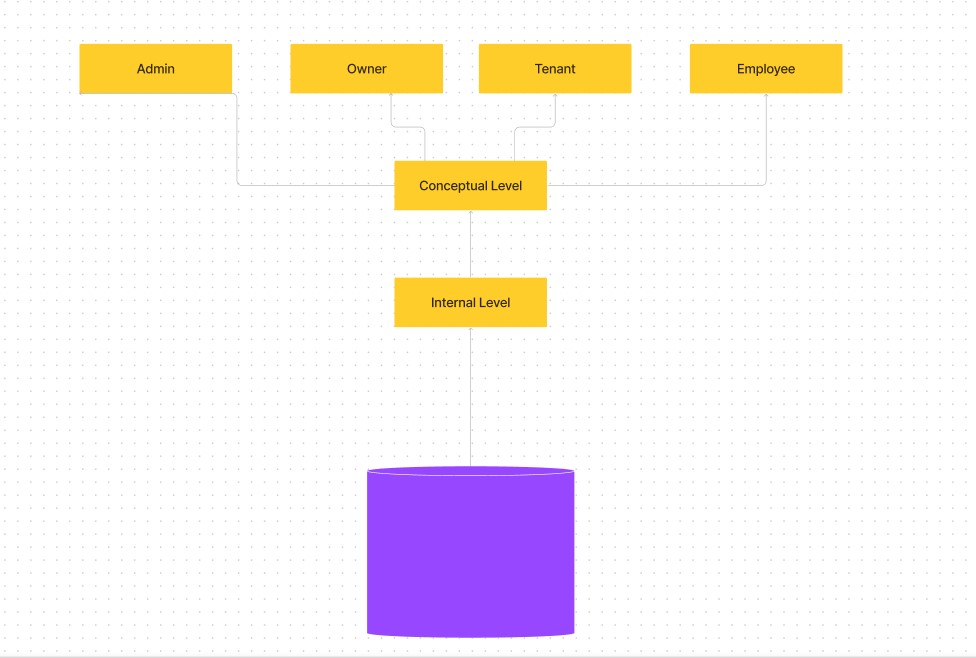
The Housing Management System software we are developing is designed to streamline the management of apartment complexes and residential buildings. This system features four distinct user modes: Administrator, Owner, Tenant, and Employee. Each user mode has its own set of functionalities tailored to their specific needs. With proper recording of all processes, everything from renting apartments to managing complaints will be smooth and efficient.

In the Administrator mode, users have complete access to all data and can manage and create owners, view complaints logged by tenants, and even manage parking slot allotments in the building. The Owner mode allows users to manage their owned buildings by accessing relevant data such as the number of apartments, available apartments, all tenant details, and complaints raised by tenants regarding their building. Only an owner can create a tenant, just as only an admin can add an owner. The Tenant mode is designed for tenants to view their personal details, including their id, room number, allotted parking space, and to pay maintenance fees or raise complaints or concerns regarding their apartments. These concerns are visible to the owner, admin, and employee. Finally, in the Employee mode, users can manage all complaints and work towards resolving them.

Overall, housing management system is an invaluable tool for landlords or apartment managers. By automating many of the complex and time-consuming tasks associated with managing an apartment complex or residential building, apartment managers or landlords can save time and reduce the likelihood of errors or oversights. With the right system in place, the management of apartment complexes and residential buildings can be made much more efficient, ensuring that tenants are satisfied, and owners are able to easily manage their properties.

The Perspective of the **“Housing Management System”** is to handle the entire activity of a Housing System. The software keeps track of all the information about the apartments, owners, tenants, and their complete details along with the total number of rooms available. The user will find it easy in this automated system rather than using the manual writing system. The system contains a database where all the information will be stored safely. The system will be user-friendly, save time and reduce the likelihood of errors or oversights.

**1.2 Database Architecture**



The above image shows the architecture of the database one such is the Database tier or backend where Data are stored and managed. SQL Server Management Studio has been used for back-end development. In the top-level tier, the User Interface (UI) in the application level is responsible for collecting and displaying information. Our desired use of choice for GUI is VC# 2022.

**LIST OF TABLES**

1. Signin table (username, password)
2. Tenant (username, tenant\_name, aadhar\_id, phone\_no)
3. Building (building\_name)
4. Apartment (occupancy\_status, apartment\_no, building\_name, design, floor\_no, parking\_space)
5. Owner\_table (username, owner\_name, apartment\_no, aadhar\_id, building\_name)
6. Employee (username, emp\_name, salary, building\_name)
7. Agreement (owner\_id, tenant\_id)
8. Rental (date\_of\_renting, monthly\_rent, tenant\_id, apartment\_no, building\_name)
9. Complaint (complaint\_no, building\_name, complaints, complaint\_status)

**LIST OF FIGURES**

1. Three-tier architecture
2. ER diagram
3. Relational Schema
4. Block Diagram

**Abbreviations**

**SRS:** Software Requirements Specification

**DBS:** Database System

**DBMS:** Database Management System

**SQL:** Structured Query Language

**GUI:** Graphical User Interface

**Px:** Pixels

**VC#:** Visual C#

**GHz:** Gigahertz

**GB:** Gigabytes

**RAM:** Random Access Memory

**XML:** Extensible Markup Language

**ID:** Identity document

**Chapter 2:**

**2.1 Capabilities and features of the Database:**

Here there are 4 users. They are the Admin, Employee, Owner and the Tenant.



ADMIN- Owner Details, Tenant Details, Employee, Create Owner, Create Employee

The admin can see all the information for everyone and can recruit new employees, and oversees the handling of the data safely.



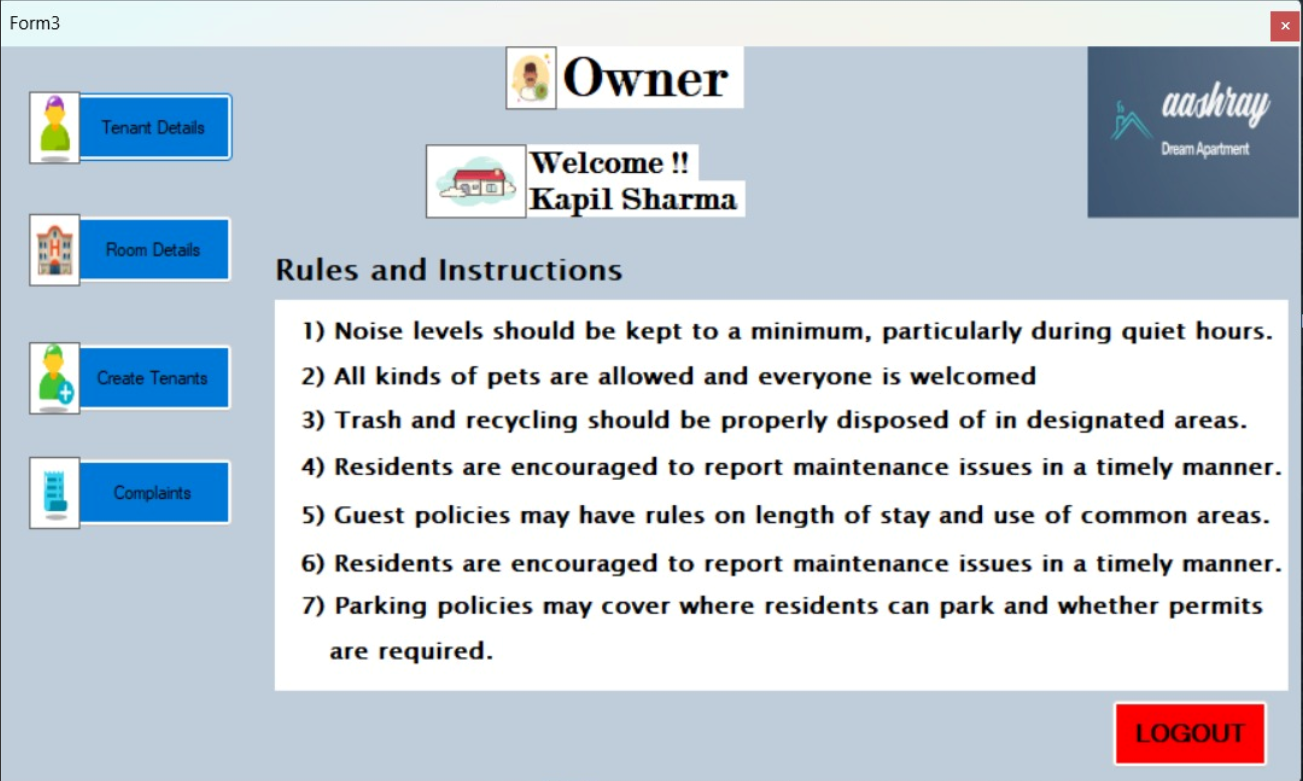
TENANT- My Details, My Apartment Details, My Owner Details, Create Complaint

The tenant can view his/her details, their apartment details log complaints and when they wish they can also contact the owner by viewing their details.

Procedure "GetTenantDetails" that takes one parameter "@tenant\_id" of type "varchar(30)".

The stored procedure performs an inner join on three tables - Tenant, rental, and agreement - to retrieve the details of a specific tenant identified by the input parameter "@tenant\_id". The retrieved tenant details include Tenant\_Name, aadhar\_ID, age, Phone\_no, Apartment\_no, and Monthly\_rent.

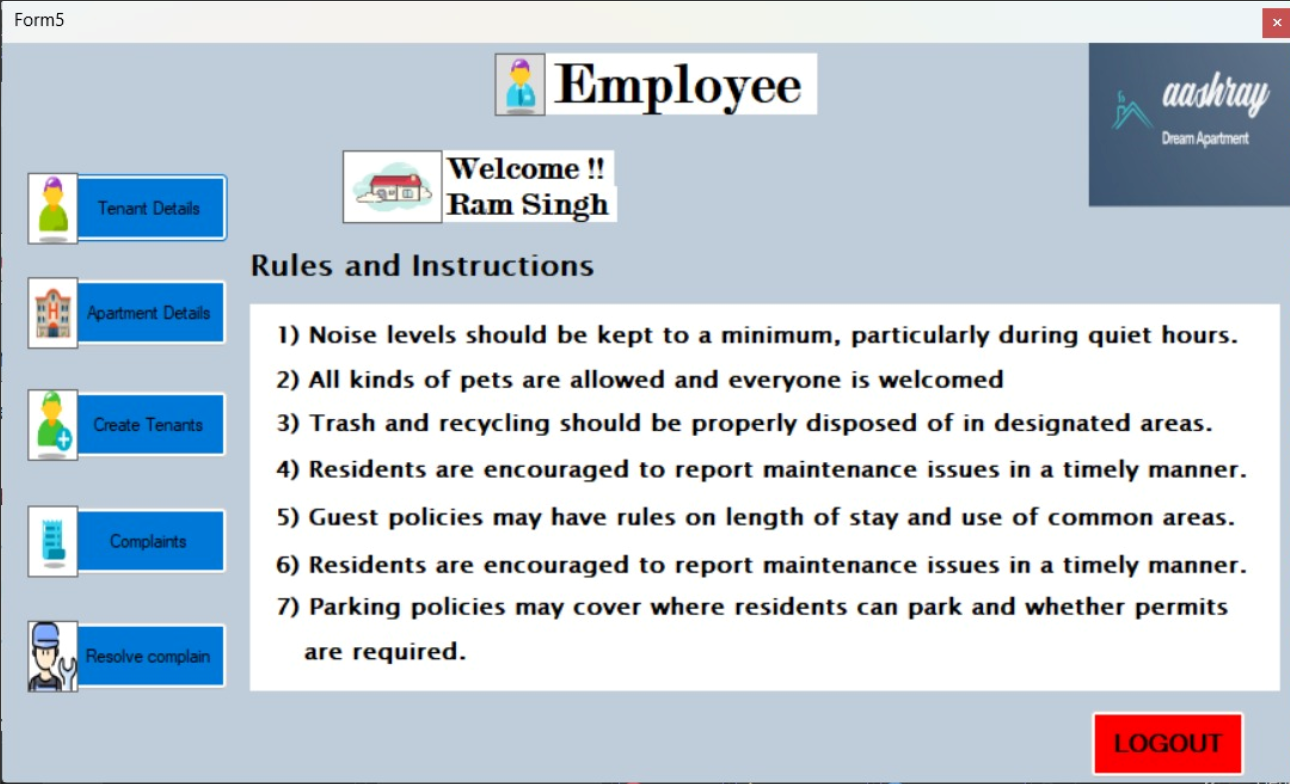
The "SET NOCOUNT ON" statement is used to turn off the message indicating the number of rows affected by the SELECT statement.



OWNER- My Tenant Details, My Apartment detail, Create Tenant(Allocated), All Complaints

The Owner can create a trigger such that when a new tenantID is created it asks the user for all the tenant details like tenant name, aadhar ID, phone no and updates those in the tenant table, simultaneously it also fetches the owner ID for that tenant and updates it in the aggrement table, updates the rental table with the tenant\_ID and the apartment allotted to him/her and also updates the occupancy status from ‘Available’ to ‘Occupied’ in the Apartment table.

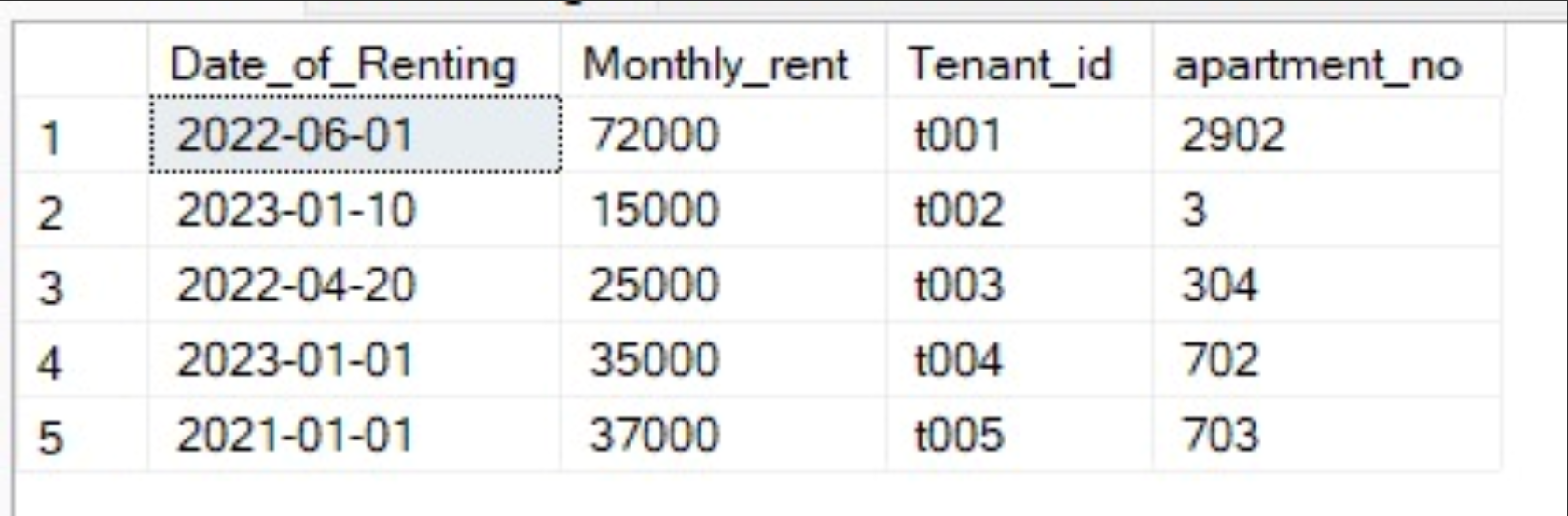
From these attributes the username and password are fixed predefined attributes and cannot be changed whereas the other attributes such as phone\_no, etc. can be.



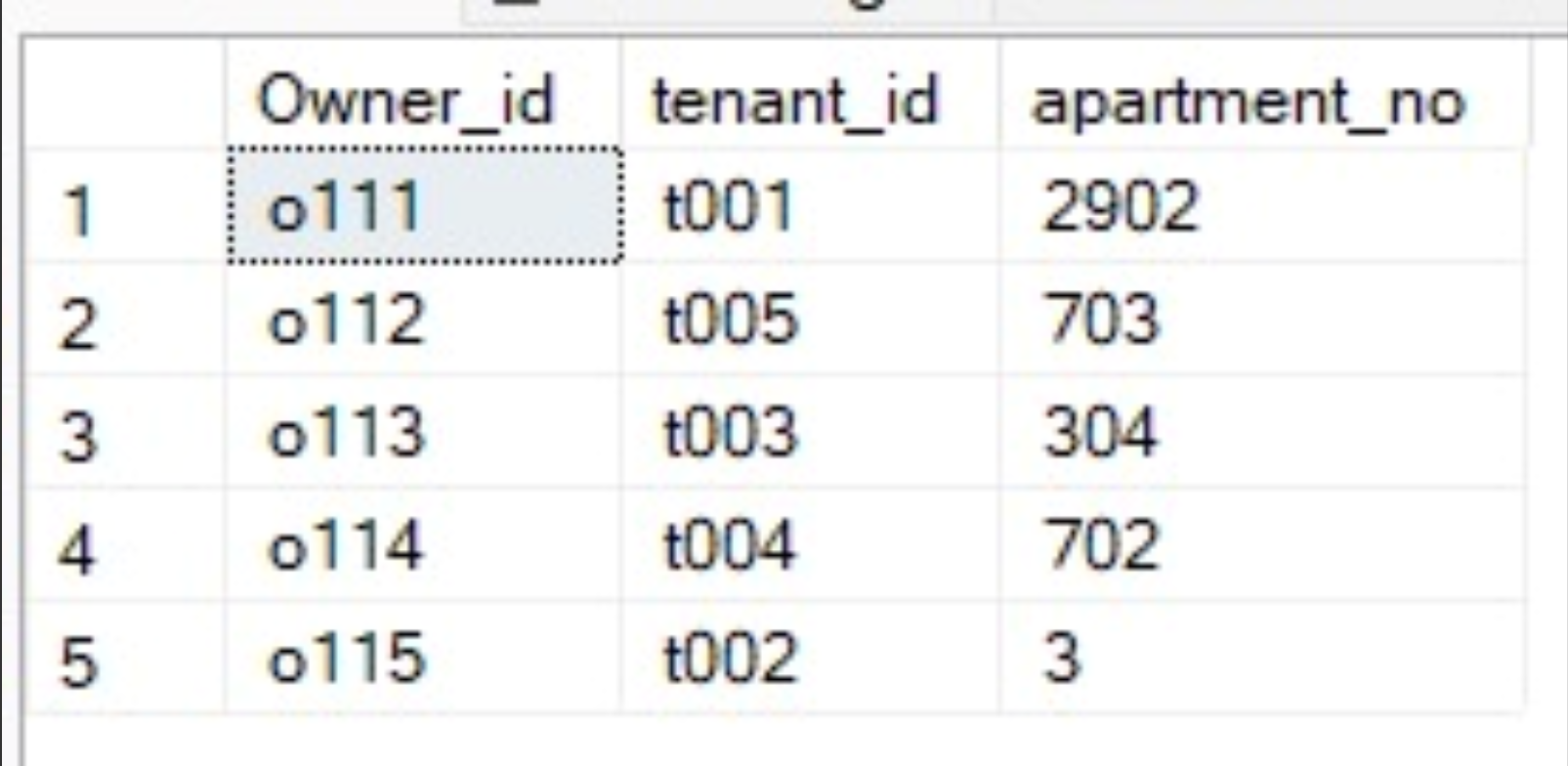
EMPLOYEE- All Tenant Details , All Apartment Details, Create Tenant, All Complaint Detail , Resolve complaints

The procedure UpdateComplaintStatus updates the Complaint\_status column in the complaint table with the value passed in @pStatus for the row where the complaint\_no equals the value passed in @pcomplaint\_no.

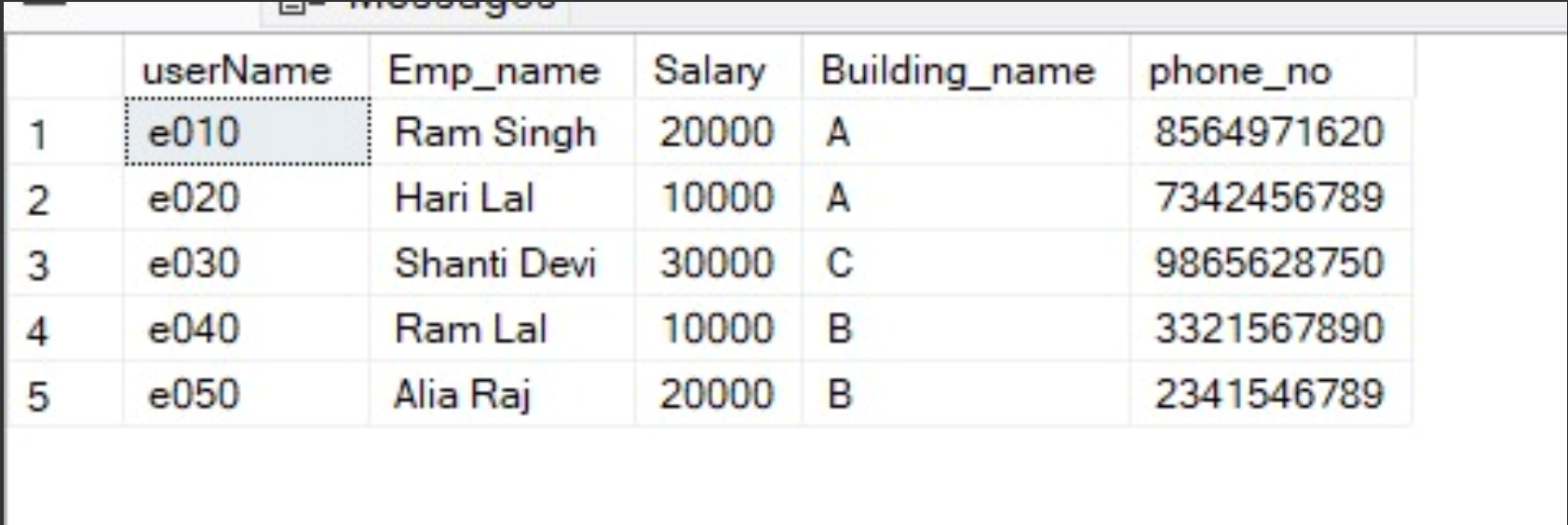
The employee can also act like a broker and create tenants in a waiting list so that whenever a vacancy is opened up the waiting tenants can be informed and can book the apartments easily.



Rental table

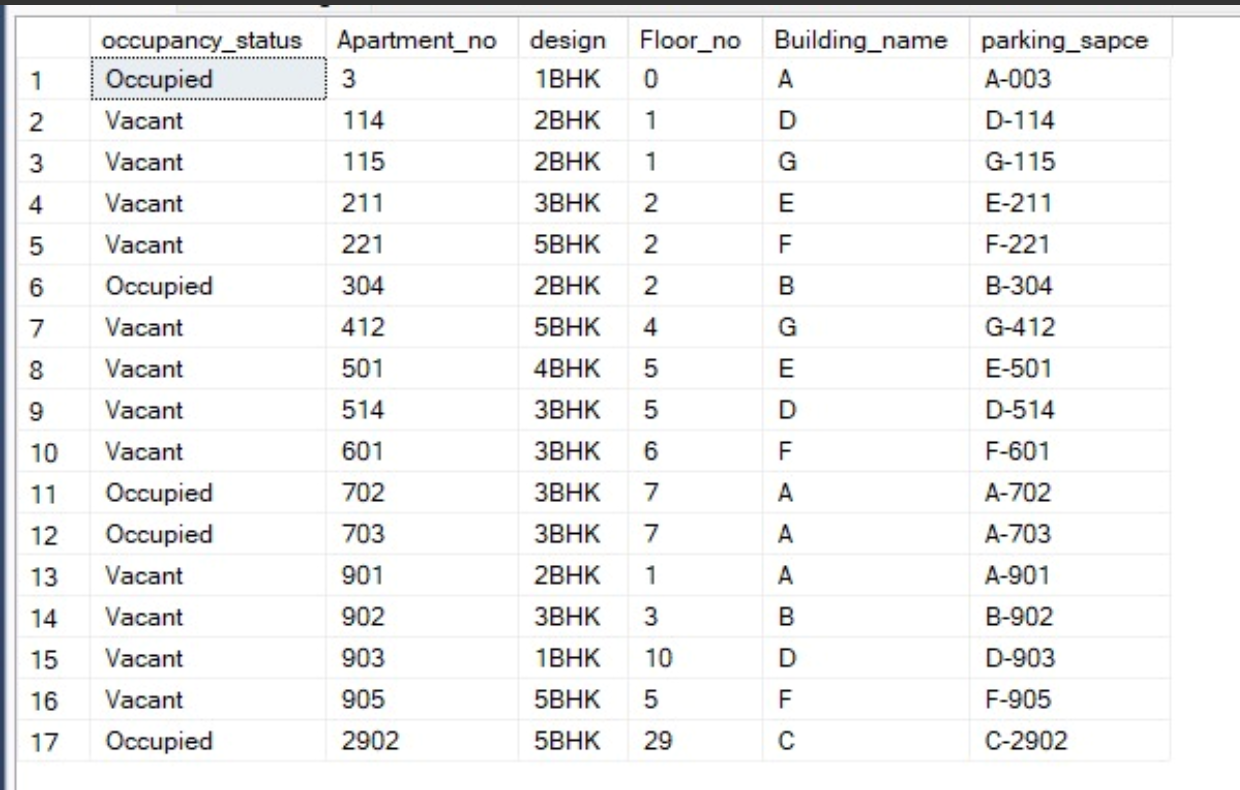


Agreement table

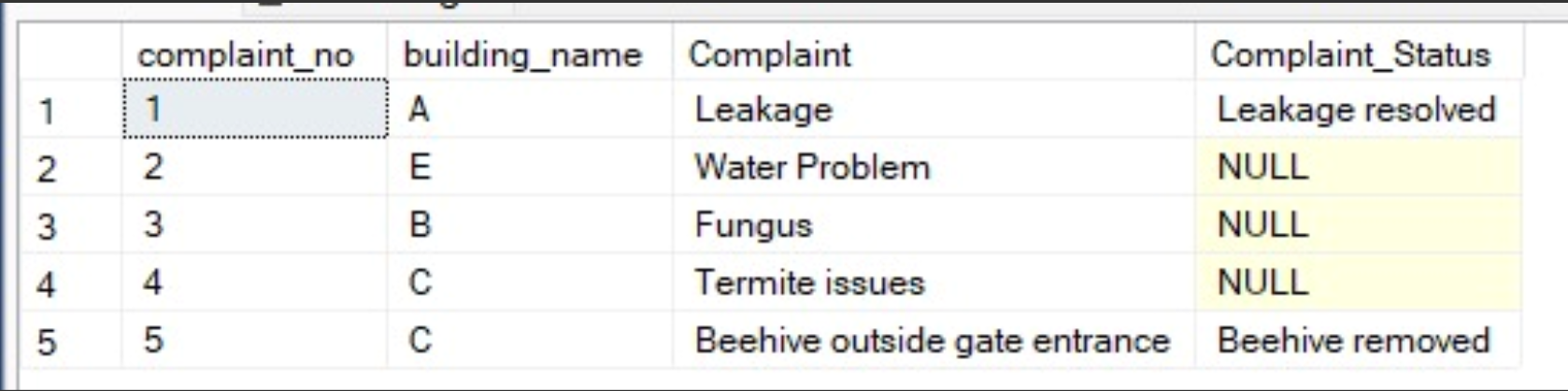


Employee table

Owner table



Apartment table



Complaint table

**Chapter 3:**

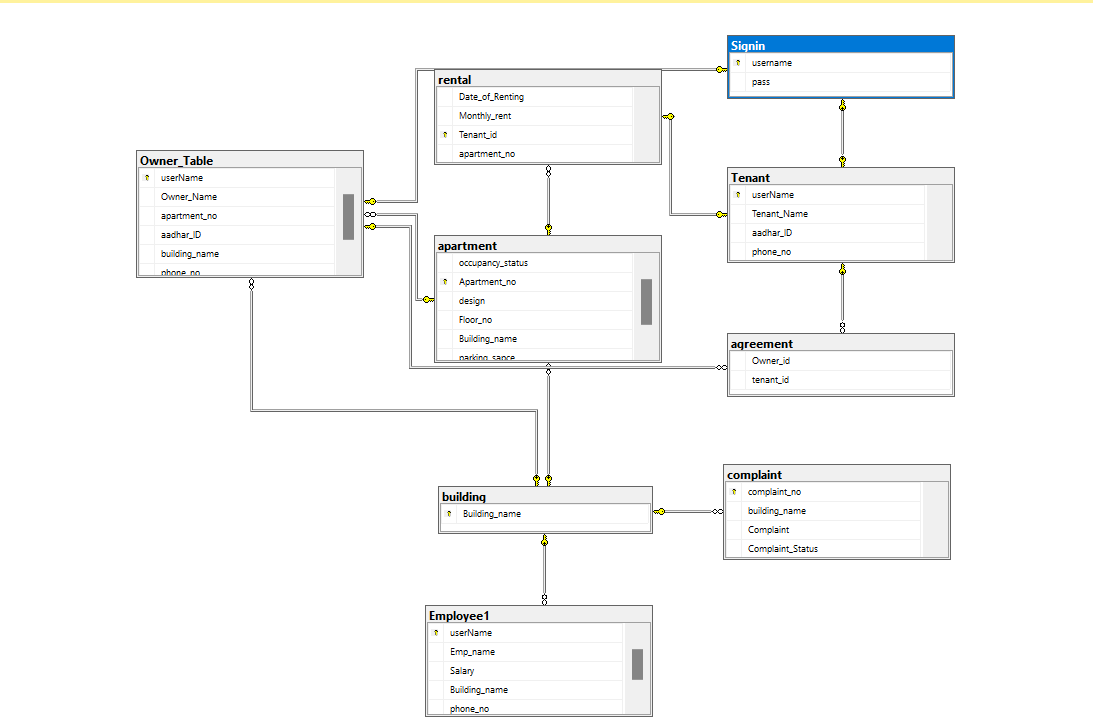
3.1 Problem statement:

Develop a software system for a housing management system that caters to the tenant and employee. The tenant can see their details as well as the apartment details. The owner can create tenants and see their details. The system should be easy to navigate, handle large amounts of data efficiently, and have access control mechanisms to prevent unauthorized access. The software should be effective and efficient, employees, tenants and owners with a seamless experience.

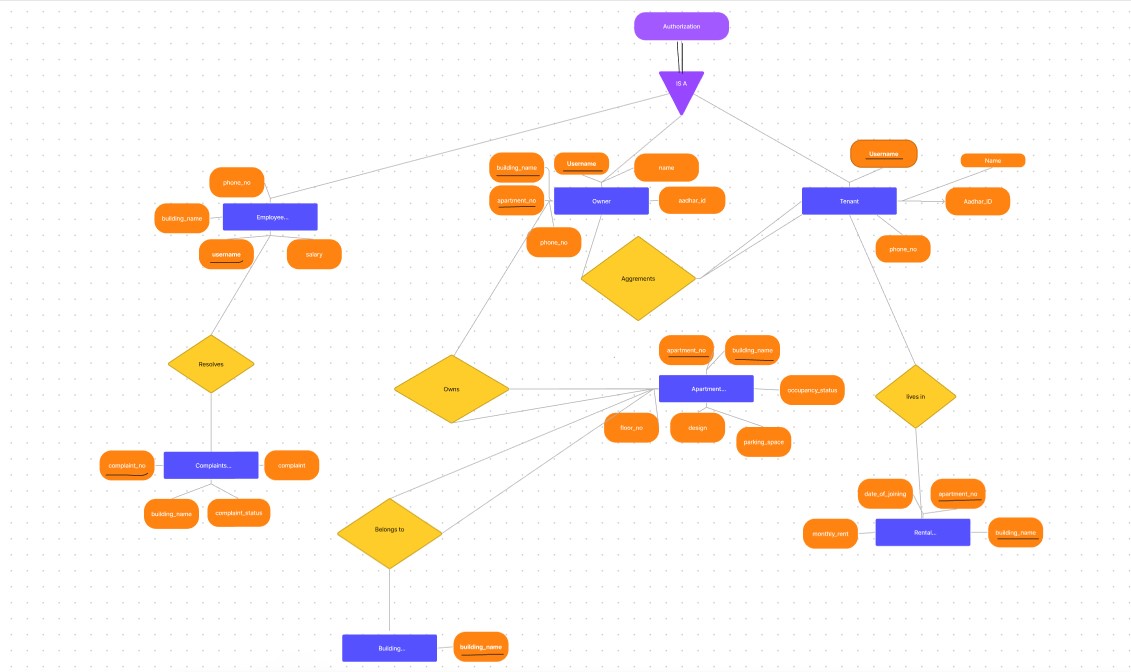
3.2 Objective of the Apartment Database:

When a housing management system will be implemented, it would be helpful to track all the details of Owner, Tenant, Employee, and the booking status of apartments. This will eradicate the problem of manual entry of the track records by providing an interface for managing it. The software will be able to handle all the necessary information.

**Chapter 4: Data Design**



Schema Diagram



ER diagram

Normalization is the process of organizing data in a database to reduce redundancy and dependency, minimize data modification errors, and simplify the query process. The tables are in 1NF, 2NF, and 3NF as each table has a primary key, all non-key attributes are dependent on the primary key, and there are no transitive dependencies. This ensures that each piece of data is stored only once and that any changes made to that data are consistent across the entire database.

In the first table 'Signin', a primary key has been defined to ensure that each record is unique, fulfilling the requirement of 1NF.

In the 'Tenant' table, the primary key has been defined as 'userName', and a foreign key relationship has been established with the 'Signin' table to ensure that each record is unique and to avoid repeating groups, fulfilling the requirement of 2NF.

In the 'Building' table, a primary key has been defined as 'Buildingname' to ensure that each record is unique, fulfilling the requirement of 1NF.

In the 'Apartment' table, the primary key has been defined as 'Apartmentno', and a foreign key relationship has been established with the 'Building' table to avoid redundancy, fulfilling the requirement of 2NF.

In the 'OwnerTable' table, a primary key has been defined as 'userName', and foreign key relationships have been established with the 'Signin', 'Apartment', and 'Building' tables to ensure that each record is unique and to avoid redundancy, fulfilling the requirement of 2NF.

In the 'Employee1' table, a primary key has been defined as 'userName', and a foreign key relationship has been established with the 'Building' table to ensure that each record is unique and to avoid redundancy, fulfilling the requirement of 2NF.

In the 'Agreement' table, foreign key relationships have been established with the 'OwnerTable' and 'Tenant' tables to ensure data consistency, fulfilling the requirement of 3NF.

In the 'Rental' table, a primary key has been defined as 'Tenantid', and foreign key relationships have been established with the 'Tenant' and 'Apartment' tables to ensure data consistency and avoid redundancy, fulfilling the requirement of 3NF.

In the 'Complaint' table, a primary key has been defined as 'complaintno', and a foreign key relationship has been established with the 'Building' table to ensure data consistency and avoid redundancy, fulfilling the requirement of 3NF.

**Chapter 5: Methodology**

Owner view

The userName column is the primary key of the table, which means that it uniquely identifies each owner in the table. The apartment\_no column is a foreign key that references the apartment table, which means that it ensures that the apartment number entered in the Owner\_Table exists in the apartment table. Similarly, the userName column is a foreign key that references the Signin table, which means that it ensures that the username entered in the Owner\_Table exists in the Signin table. Finally, the Building\_name column is a foreign key that references the building table, which means that it ensures that the building name entered in the Owner\_Table exists in the building table.

The Owner\_Table can be used to retrieve information about the owners of apartments in a building. It can also be used to update or delete information about owners, and they can also create new tenants.

Tenant view

The userName column is the primary key of the table and is used to uniquely identify each tenant. The Tenant\_Name column stores the name of the tenant, the aadhar\_ID column stores the Aadhar ID of the tenant, and the phone\_no column stores the phone number of the tenant.

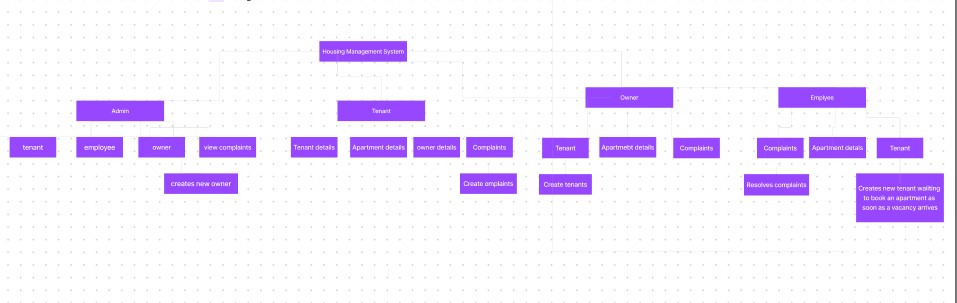
This table can be used to retrieve, update, or delete information about tenants in an apartment. For example, if the owner wants to know the name of a particular tenant, they can retrieve it from the Tenant\_Name column using the userName as a reference. Similarly, if the owner wants to update the phone number of a tenant, they can do so by updating the phone\_no column for that particular userName. The aadhar\_ID column can be used to verify the identity of the tenant.

Employee View

The Employee table is table to store information about employees, including their username, name, salary, building name, and phone number. The Building\_name column is a foreign key that references the Building table.

On the other hand, the Complaint table seems to be a table that stores information about complaints, including the complaint number, building name, complaint description, and complaint status. The Building\_name column is also a foreign key that references the Building table with cascading delete and update.

With regards to complaints, the Employee table can be used to retrieve information about employees who work in a particular building. This information can be used to investigate and resolve complaints related to that building. Additionally, the Employee1 table can be used to update the status of a complaint if an employee is responsible for handling complaints.



Block Diagram

**Chapter 6: Result**

The result of the project was to fulfil the objectives of the project. Moreover, it helps to save time and efficiently books the flats available more actively according to their requirements based on the availability and the real-time updates. All this is done while adding new features and improving the security of the system.

**Chapter 7:conclusion and future work**

The housing management system has been normalized up to the third level of normalization, ensuring data consistency and avoiding redundancy. This means that the data has been organized in a structured way to eliminate data duplication and inconsistencies, leading to more efficient data analysis and better decision-making. This also helps in improving the user experience by providing standardized data flow and easy access to information. Additionally, future work on the system can focus on features like predictive maintenance and sustainability tracking, which can further optimize the system, making it more efficient and cost-effective for both the building owners and tenants.

**References:**

[1] DBMS Lab Manual

[2] Database System Concepts, 6th Edition, A. Silberchatz, Henry F.Korth, and S. Sudarshan

<https://www.tutorialspoint.com/csharp/index.htm>

<https://www.w3schools.com/sql/>

<https://financesonline.com/10-must-features-property-management-system/>

**Specifications to be followed:**

1. **Font size: 12 for regular text, font size: 14 for side headings and font size: 26 for chapter numbers and title**
2. **Font Type: Times New Roman**
3. **Text alignment: Justified (Both Sides)**

**For further clarifications contact the faculty.**

**Following list of chapters will be expected:**

1. **Introduction**
2. **Literature Survey / Background**
3. **Objectives/Problem Statement**
4. **Data design (ER Diagram, Reduction, Schema Diagram, Normalization with suitable justifications)**
5. **Methodology (Implementation Details with block diagram to explain the project in detail. Do not put code here.)**
6. **Results**
7. **Conclusion and Future Work**
8. **References**