APARTMENT MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

RISHI KHARE[RA2111047010027]
MAYUR PAL[RA2111047010033]

Under the Guidance of

Dr. Dinesh G
(Assistant Professor, Department of Computational Intelligence)

In partial satisfaction of the requirements for the degree of

BACHELORS IN TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

April 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this Course Project Report titled **"APARTMENT MANAGEMENT SYSTEM"** is the bonafide work done by **RISHI KHARE (RA2111047010027)** and **MAYUR PAL (RA2111047010033)** of II Year/ IV Sem B.Tech (AI) who carried out under my supervision for the course 18AIC207J - Database Management Systems for Artificial Intelligence. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE
Faculty In-Charge
Dr.Dinesh G
Assistant Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

HEAD OF THE DEPARTMENT
Dr. R Annie Uthra
Professor and Head ,
Department of Computational Intelligence,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

ABSTRACT

The main aim of Apartment Management Mini DBMS project is to rent apartments and get payments from respective tenants. We aim to demonstrate the use of create, read, update and delete MySQL operations through this project. The project starts by creating a Owner and by adding details of tenant buying apartment. The owner provides building & apartment details to their tenant with the start date of rent. The tenants should pay their rents to the owner after every month completion. The admin has full access to the system and can manage every action within it. The system requires users to log in as either an admin or an employee to access it. The design of the project is simple and easy to use. The system enables the user to manage the activities and the database seamlessly.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Apartment Management System

Apartment management system is a computer-based system which is used to monitor the various activities of a regular residential metropolitan society. The concept of apartment management system has arisen from the fact that various large societies need monitoring and maintenance for their various day to day activities. The goal of this research work is to provide a solution to the problem of apartment management, by designing a computerized system which is user-friendly and GUI-oriented that will be compatible with the existing manual systems. Apartment Management System is used to help people in the apartment to pay bills such as maintenance bills, water bills, make a complaint and update tenants' information. This will make the system more effective, efficient, and interactive, and it will also solve the problems that come up when using the current system.

1.2. Objectives

Following are the objectives of this project:

- Ensure process time and increase throughput.
- Simplifies the operation.
- Avoid some manual work in the existing system.
- Reduce data redundancy and inconsistency.
- User friendly input screens to enter data.
- Provide uniformity among all screen formats.
- The system works in high speed and accuracy.
- It handles bulk amount of data.
- Immunization of the system from unauthorized user accesses.

CHAPTER 2

MYSQL

2.1. About MySQL

MySQL is a popular open-source relational database management system (RDBMS) used in web development to store and manage data. MySQL is used to create and manage databases, tables, and relationships between data, and can be used in conjunction with programming languages like PHP, Python, and Java to build dynamic web applications.

2.1.1. Key Features

Here are some key features and modules of MySQL:
1. Database creation and management: MySQL allows you to create and manage databases, which can contain one or more tables of related data.

2. Data types: MySQL supports several data types, such as integer, text, and date/time, which can be used to store different types of data.

3. Queries: MySQL supports a powerful SQL syntax that allows you to retrieve, update, and manipulate data in tables.

4. Indexes: MySQL allows you to create indexes on columns in tables, which can speed up queries and improve performance.

5. Relationships: MySQL supports several types of relationships between tables, such as one-to-one, one-to-many, and many-to-many, which can be used to model complex data relationships.

6. Security: MySQL provides several security features, such as user authentication and access control, to ensure that only authorized users can access and modify data.

2.2. Connectivity Used

Connectivity used for database access:
PHP (Hypertext Preprocessor) is a server-side programming language that is commonly used in web development to build dynamic web applications. PHP can be used to connect to a MySQL database and perform CRUD (Create, Read, Update, Delete) operations on data.
Here are some key features and modules of PHP for database connectivity:
1. MySQL extension: PHP provides the MySQL extension, which allows you to connect to a MySQL database and perform CRUD operations on data. The MySQL extension provides both an object-oriented and a procedural interface for working with databases.
2. PDO extension: PHP also provides the PDO (PHP Data Objects) extension, which is a database abstraction layer that supports multiple databases, including MySQL. PDO provides a consistent interface for working with databases, regardless of the underlying database system.
3. Connection parameters: To connect to a MySQL database using PHP, you need to provide several connection parameters, such as the host name, username, password, and database name. These parameters are used to establish a connection to the database.
4. Query execution: Once you have established a connection to the MySQL database, you can execute SQL queries using PHP. PHP provides several functions for executing queries, such as MySQL query () and PDO::query().
5. Prepared statements: To improve security and prevent SQL injection attacks, PHP supports prepared statements, which allow you to parameterize queries and separate the query logic from the data.

CHAPTER 3

FEASIBILITY STUDY

3.1. Study

All projects are feasible when given unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. A feasibility study is not warranted for systems in which economic justification is obvious, technical risk is low, few legal problems are expected and no reasonable alternative exists. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study will decide if the proposed system will be cost effective from the business point of view and if it can be developed in the given existing budgetary constraints. The feasibility study should be relatively cheap and quick. The result should inform the decision of whether to go ahead with a more detailed analysis. Feasibility study may be documented as a separated report to higher officials of the top-level management and can be included as an appendix to the system specification. Feasibility and risk analysis is related in many ways. If there is more project risk then the feasibility of producing the quality software is reduced. The study is done in these phases:

3.2. Phases for Quality Software

Technical Feasibility

This is related to the technicality of the project feasibility if check the cost to conduct a full system investigation, cost of hardware and software. The apartment Management system supports the economic feasibility to a great extent. Development of the system and the cost of hardware and software are not high. This reduces effort and time of us. This makes software economically feasible.

Economic Feasibility

A system that can be developed technically and that will be used, if installed, must be sill good. Always the financial benefits must be equal or exceed the cost. Economic analysis is the most frequently used method for evaluating the effectiveness of a candidate system or more commonly known as cost or benefits analysis.

Relational Feasibility

Proposed systems are beneficial only if they can be turned into information systems. That is, it will meet the organizations operating requirements and also checks that whether the system will work when it is developed and installed. Therefore, it is understandable that the introduction of a candidate system requires special efforts to educate, sell and train others. The Apartment Management system supports the operational feasibility to a great extends. The performance of this software is more accurate, more user friendly, effective, error free.

Behavioral Feasibility

Computers have been known to facilitate changes where as people are intently resistant to change. Therefore, it is necessary that an evaluation should be made about the user's attitude towards the new system. This is called behavioral feasibility.

CHAPTER 4

FUNCTIONALITIES

Following are the functions the users are allowed to access to:

4.1. Admin

Admin can login.
Admin can view the tenant and owner details.
Admin can create owner.
Admin can allot parking slot.
Admin can view the complaints.
Admin can see total Owners.
Admin can see total Tenants.
Admin can see total Employee.

4.2. Owner

Owner can see the Tenant details of his/her owned room.
Owner can create Tenant.
Owner can see the complaints from his/her owned room.
Owner can see the Room Details.
Owner can see Total Complaint.
Owner can see Number of Employees.

4.3. Tenant

Tenant can see the allotted parking slot.
Tenant can pay maintenance fee.
Tenant can raise complaints.
Tenant can see his/her Tenant id.
Tenant can see his/her Name.
Tenant can see his/her Age.
Tenant can see his/her DOB.
Tenant can see his/her Room no.

4.4. Employee

Employee can see all the complaints.
Employee can see total number of complaints.

All the admins, owners, tenant, employees can login and logout.

CHAPTER 5

ER DIAGRAMS

5.1. Database Schema and ER

This is from the authorized and general point of view:

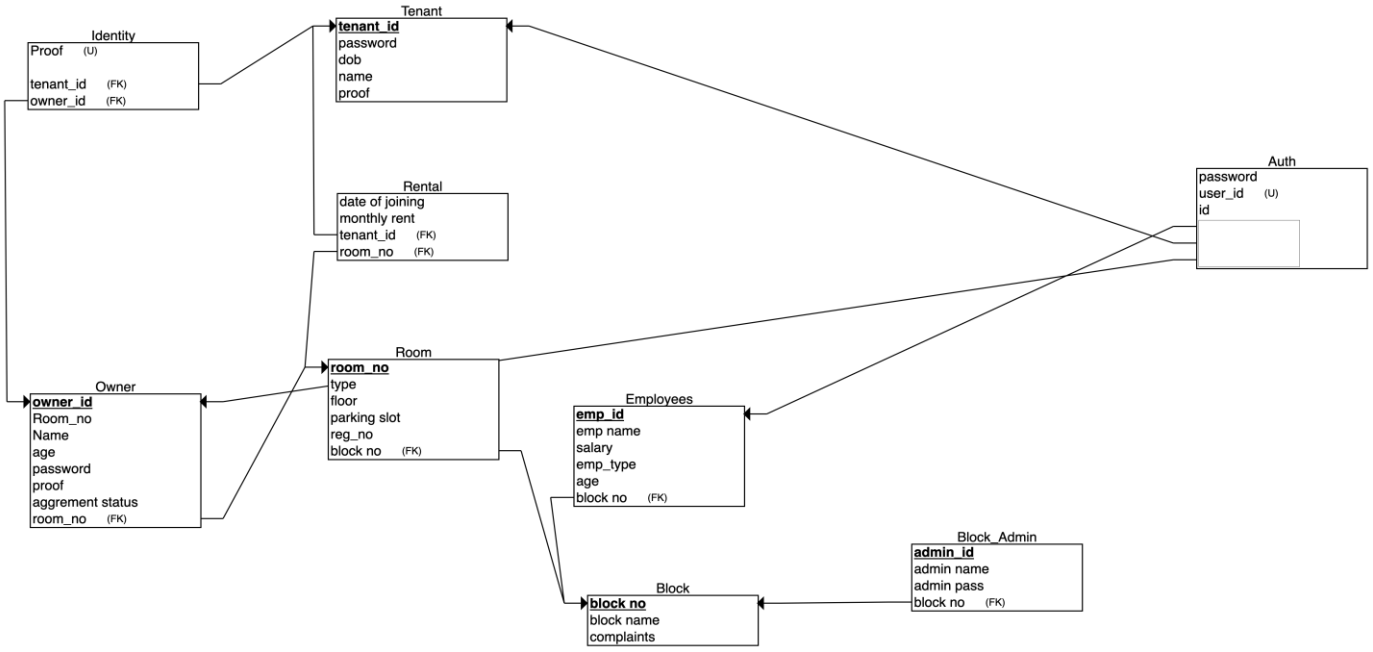

Fig 5.1 – Database Schema

The following entity relationship diagram show how each character (owner, tenant, admin, employee) is related to each other in this scenario.
It also depicts what roles each characters share and have.
The database will have the details of all the tenants and their respective owners as well as the details of the rooms and their types.

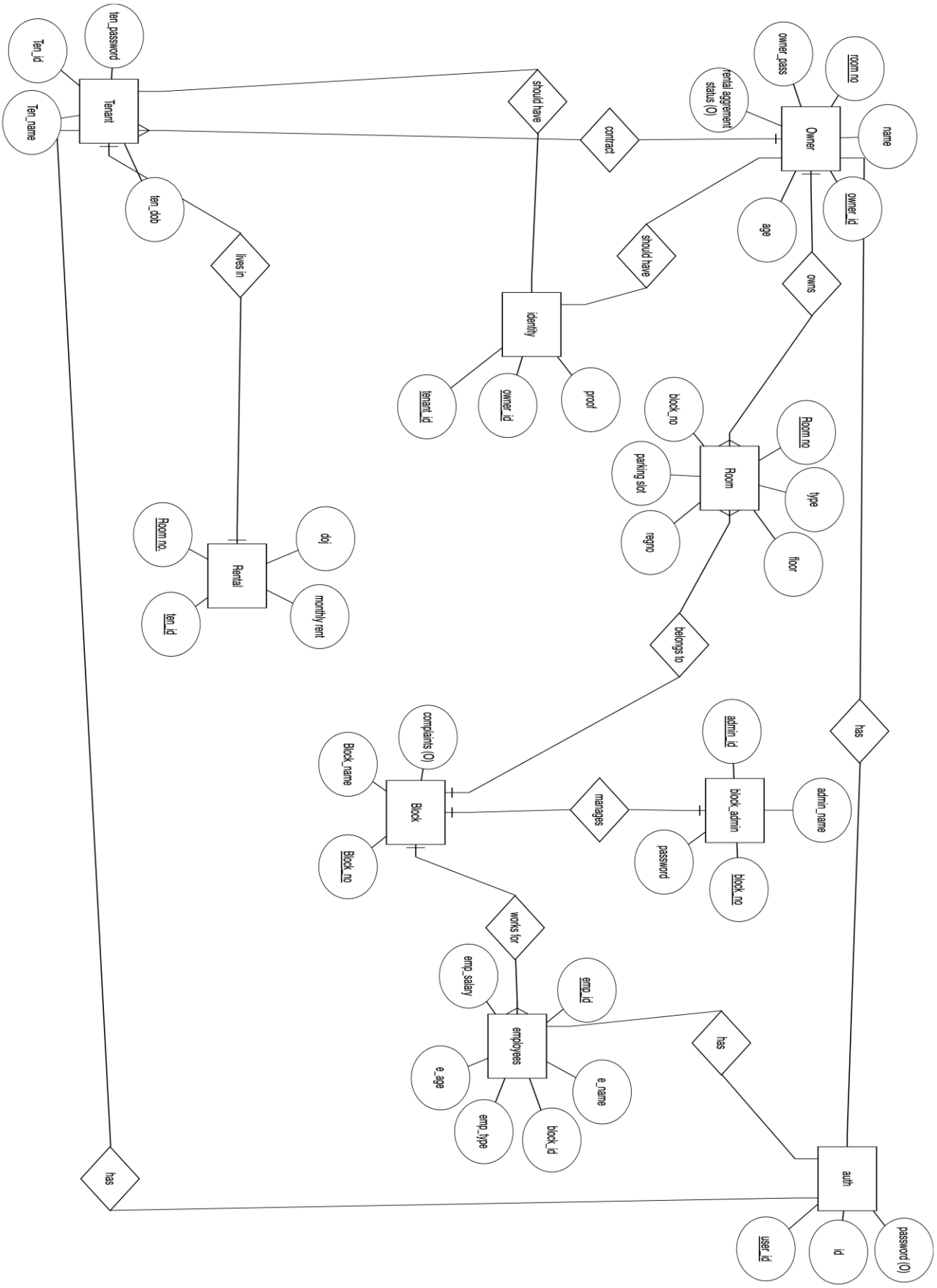

Fig 5.2 – Entity Relationship

CHAPTER 6

CODE

```
const mysql = require('mysql');
const config = require('./config_sql');
const con = mysql.createConnection({
    host: config.host,
    user: config.uname,
    password: config.upass,
    database: config.database
});
connect();
function connect()
{
    con.connect(function(err)
    {
        if (err) throw err;
        console.log("database Connected!");
    });
}
function registercomplaint(values,callback)
{
  sql = ' update block set complaints= ? where block_no = ? and room_no= ?';
  console.log();
  con.query(sql,values,(err,results)=>
  {
      if (err)
      {
          console.log(err);
      }
    callback(err,results);
  })
}
function totalowner(callback)
{
    sql = 'SELECT COUNT(owner_id) FROM owner';
    con.query(sql,(err,results)=>
    {
        callback(err,results);
    })
}
function getdata(tablename,callback)
{
    sql = 'select * from '+tablename+';';
    con.query(sql,(err,results)=>
    {
        callback(err,results);
    })
}
function createowner(values,callback)
{
    sql = 'insert into owner values(?,?,?,?,?,?)';
    con.query(sql,values,(err,results)=>
    {
```

```
        callback(err,results);
    })}
function createownerproof(values,callback)
{
    sql = 'insert into identity values(?,?,null);';
    con.query(sql,values,(err,results)=>
    {
        callback(err,results);
    })}
function bookslot(values,callback)
{
    sql = 'update room set parking_slot =  ? where room_no = ?';
    con.query(sql,values,(err,results)=>
    {
        callback(err,results);
    })}
function viewcomplaints(callback)
{
    sql = 'select * from oo;';
    con.query(sql,(err,results)=>
    {
        callback(err,results);
    })}
function ownercomplaints(ownerid,callback)
{
    sql = 'select complaints,room_no from block where room_no in (select room_no from owner
where owner_id in(select id from auth where user_id=?))';
    con.query(sql,ownerid,(err,results)=>
    {
        callback(err,results);
    })}
function totaltenant(callback)
{
    sql = 'SELECT COUNT(tenant_id) FROM tenant';
    con.query(sql,(err,results)=>
    {
        callback(err,results);
    })}
function totalemployee(callback)
{
    sql = 'SELECT COUNT(emp_id) FROM employee';
    con.query(sql,(err,results)=>
    {
        callback(err,results);
    })}
function totalcomplaint(callback)
{
    sql = 'SELECT COUNT(complaints) FROM block';
    con.query(sql,(err,results)=>
    {
        callback(err,results);
    })
}
function gettenantdata(tid,callback)
{
    sql = 'select * from tenant where tenant_id in (select id from auth where user_id=?)';
```

```
    con.query(sql,tid,(err,results)=>
    {
        callback(err,results);
    })
}
function createtenant(values,callback)
{
    sql = 'insert into tenant values(?,?,?,null,?,?)';
    con.query(sql,values,(err,results)=>
    {
        callback(err,results);
    })
}
function createtenantproof(values,callback)
{
    sql = 'insert into identity values(?,null,?)';
    con.query(sql,values,(err,results)=>
    {
        callback(err,results);
    })
}
function createuserid(values,callback)
{
    sql = 'insert into auth values(?,?,?)';
    con.query(sql,values,(err,results)=>
    {
        callback(err,results);
    })
}
function ownertenantdetails(values,callback)
{
    sql = 'select * from tenant where room_no in (select room_no from owner where owner_id
in(select id from auth where user_id=?))';
    con.query(sql,values,(err,results)=>
    {
        callback(err,results);
    })
}
function paymaintanence(id,callback)
{
    sql = 'update tenant set stat="paid" where tenant_id in (select id from auth where user_id=?)';
    con.query(sql,id,(err,results)=>
    {
        callback(err,results);
    })
}
function ownerroomdetails(values,callback)
{
    sql = 'select * from room where room_no in (select room_no from owner where owner_id
in(select id from auth where user_id=?))';
    con.query(sql,values,(err,results)=>
    {
        callback(err,results);
    })}
function viewparking(id,callback)
{
```

```
    sql = 'select parking_slot from room where room_no in (select room_no from tenant where
tenant_id in (select id from auth where user_id=?))';
    con.query(sql,id,(err,results)=>
    {
        callback(err,results);
    })}
function empsalary(id,callback)
{
    sql = 'select salary from employee where emp_id in (select id from auth where user_id=?)';
    con.query(sql,id,(err,results)=>
    {
        callback(err,results);
    })}
function authoriseuser(username,password,callback)
{
    let results;
    sql = 'SELECT password from auth where user_id = ?';
    const value = [username];
    console.log(value);
    con.query(sql,value,(err,result)=>
    {
        if(result.length===0)
        {
            results = "denied";
            callback(err,results);
            return;
        }
        else
        {
        const  resultArray = Object.values(JSON.parse(JSON.stringify(result))[0])[0];
        if(password === resultArray)
        {
            results = "granted";
        }
        else
        {
            results = "denied";
        }
        callback(err,results);
    }
    })}
module.exports = {
    connect, registercomplaint,
    createowner, bookslot,
    getdata, totalowner,
    totaltenant, totalemployee,
    totalcomplaint, createownerproof,
    viewcomplaints, authoriseuser,
    gettenantdata, createtenant,
    createtenantproof, ownerroomdetails,
    ownercomplaints, viewparking,
    createuserid, paymaintanence,
    empsalary, ownerroomdetails,
    ownertenantdetails
}
```

CHAPTER 7

CONCLUSION

The conclusion arrived at, is that the maintenance of this project which has been developed is totally beneficial in all aspects, considering the data accuracy or data security or data access speed or even in case of usage. The project has met its objectives. The system has been thoroughly tested with various test and found to be fit for implementation. The system reliability is high and enough security has been provided. The objectives of the system have been achieved.

7.1. Objectives Achieved

The objectives that have been achieved are:

- Ensure process time and increase throughput.
- Simplifies the operation.
- Avoid some manual work in the existing system.
- Reduce data redundancy and inconsistency.
- User friendly input screens to enter data.
- Provide uniformity among all screen formats.
- The system works in high speed and accuracy.
- It handles bulk amount of data.
- Immunization of the system from unauthorized user accesses.

7.2. Advantages

Advantages:

- Involves fast processing of data
- Flexible
- Very less paperwork
- Easy availability of data
- Cost effective
- Minimize number of staff
- Group booking

REFERENCES

1. https://github.com/imtharun/Apartment-management-system-dbms

2. https://www.slideshare.net/Mayankgautam19/apartment-manageemnt-system?next_slideshow=154011623

3. https://www.scribd.com/document/339096229/Apartment-management-project-report#