VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belagavi - 590018

DBMS MINI PROJECT REPORT ON "APARTMENT MANAGEMENT SYSTEM"

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

In

INFORMATION SCIENCE & ENGINEERING

Submitted by

PRANJAL AGRAWAL (1JS19IS062)

&

REETHU R (1JS19IS077)

Under the guidance of

Dr. DAYANAND P

Professor

Dept. of ISE, JSSATE

&

Dr. SOWMYA K N

Assistant Professor

Dept. of ISE, JSSATE

DEPARTMENT OF INFORMATION SCIENCE OF ENGINEERING

JSS ACADEMY OF TECHNICAL EDUCATION

JSSATEB Campus, Dr Vishnuvardhan Road, Uttrahalli-Kengeri Main Road

Bangalore-560060

JSS ACADEMY OF TECHNICAL EDUCATION

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that DBMS MINI PROJECT Report entitled "APARTMENT MANAGEMENT SYSTEM" is a bonafide work carried out PRANJAL AGRAWAL (1JS19IS062) and REETHU R (1JS19IS077) in partial fulfillment for the award of degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during the year 2021-22.

Signature of the Guide	**Signature of the Guide**	**Signature of the HOD**
Dr. Dayanand P	Dr. Sowmya KN	Dr. Rekha PM
Professor,	Assistant Professor,	Associate Professor & HOD,
Dept of ISE,	Dept of ISE,	Dept. of ISE,
JSSATE, Bengaluru	JSSATE, Bengaluru	JSSATE, Bengaluru

Signature of the EXAMINERS

1.

2.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible. So with gratitude, we acknowledge all those whose guidance and encouragement crowned my effort with success.

First and foremost, we would like to thank his **Holiness Jagadguru Sri Shivarathri Deshikendra Mahaswamiji** and **Dr. Mrityunjaya V Latte**, Principal, JSSATE, Bangalore for providing an opportunity to carry out the Project Work as a part of our curriculum in the partial fulfilment of the degree course.

We express our sincere gratitude for our beloved Head of the department, **Dr. Rekha P.M.** for her co-operation and encouragement at all the moments of our approach.

It is our pleasant duty to place on record our deepest sense of gratitude to our respected guide **Dr. Dayanand P** and **Dr. Sowmya KN** for the constant encouragement, valuable help and assistance in every possible way.

We are thankful to the Project Coordinators **Dr. Dayanand P,** Associate Professor, and **Dr. Sowmya KN,** Assistant Professor, for her continuous co- operation and support.

We would like to thank all ISE Department Teachers, non-teaching staff and Library staff for providing us with their valuable guidance and for being there at all stages of our work.

PRANJAL AGRAWAL(1JS19IS062)
REETHU R(1JS19IS077)

ABSTRACT

Nowadays, the apartments are increasing day by day. People coming from abroad in order to settle down in the city, they will not be knowing the information about the apartments. As the name suggests, the apartment management system helps in maintaining the information of the people in the particular apartment, number of apartments available and so on. The user interface must be simple and easy to understand even by the common man. This application will help in reducing the pen paper work through which we store the details of the people who stay in the apartments. Their details can be obtained just in one click and with great ease. This will be one of the interesting projects that one can work on and implement in real time since it will be very useful to the owners of the apartments to maintain the details of the people with great ease.

As our software is coded in html/php/javascript, so it is platform independent i.e. It can work on any operating system whether it can be any version of Microsoft window, Linux, Mac OS or any other.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 INTRODUCTION TO PROJECT

1.1.1 OVERVIEW

Apartment management system, in all its uniqueness, is a comprehensive solution for efficient, quick and elegant management of a apartment system which includes a wide array of functional activities ranging from buildings, apartments and owner details editor to manage apartment. It also provides an option for the user to filer and search for the required data.

1.1.2 PROBLEM STATEMENT

To build a management system that can efficiently handle all the functional activities of an apartment - number of buildings, number of apartments, apartments on rent, apartments on lease and booking of the apartments.

1.1.3 SCOPE AND OBJECTIVES

The goal of this mini project is to develop a website for admin to carry out the apartment management process conveniently. The main objectives of this website development can be defined as follows:

1. To develop a system that provides functions to support admin to sign in to their accounts or to sign up to continue using the system.
2. To maintain records of buildings, apartments and the owner's information in a centralized database system.
3. To develop a system for admin to be able to assign an apartment to a tenant.
4. To provide the users the functionalities of filtering out the required data.

1.1.4 LIMITATIONS

Every user must go through the process of authentication and authorization provided by the system to use the application. One of the limitations of this system is that the admin has to do all the work of adding, removing, updating the details of the buildings, apartments and then managing the number of apartments by assigning an apartment to a tenant. The admin has to also filter the required data.

1.2 INTRODUCTION TO DBMS

1.2.1 INTRODUCTION TO DATABASE AND DBMS

Database and database technology has a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, education, and library science. The word database is so commonly used that we must begin by defining what the database is.

Our initial definition is quite general. A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book, or you may have stored it on a hard drive, using personal computers and software such as Microsoft Excel. This collection of related data with an implicit meaning is a database.

The preceding definition of a database is quite general, for example, we may consider the collection of words that make up this page of text to be related data and hence to constitute a database. However, the common use of the term database is usually more restricted. A database has the following properties:

A database represents some aspect of the real world, sometimes called the mini world or the universe of discourse. The changes to the mini world are reflected in the database.

A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.

A database is designed, built and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

In other words, a database has some source from which data is derived, some degree of interaction with events in the real world, and an audience that is actively interested in its contents. The end-users of the database may perform business transactions (for example, a customer buys a camera) or events may happen that may cause the information in the database to change.

In order for a database to be accurate and reliable at all times, it must be a true reflection of the mini world that it represents; therefore, changes must be reflected in the database as soon as possible.

A database can be of any size and complexity. A database may be generated and maintained manually or computerized. For example, a library card catalog is a database that may be created and maintained manually. A database is a collection of data, typically describing the activities of one or more related organizations. For example, a university database might contain information about the following:

1. Entities such as students, faculty, courses, and classrooms.

2. Relationships between entities, such as student's enrollment in courses, faculty teaching courses, and the use of rooms for courses.

A database management system, or DBMS, is software designed to assist in maintaining and utilizing a large collection of data. The need for such systems as well as their use is growing rapidly.

DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more. In the case of multiple users, it also maintains data consistency.

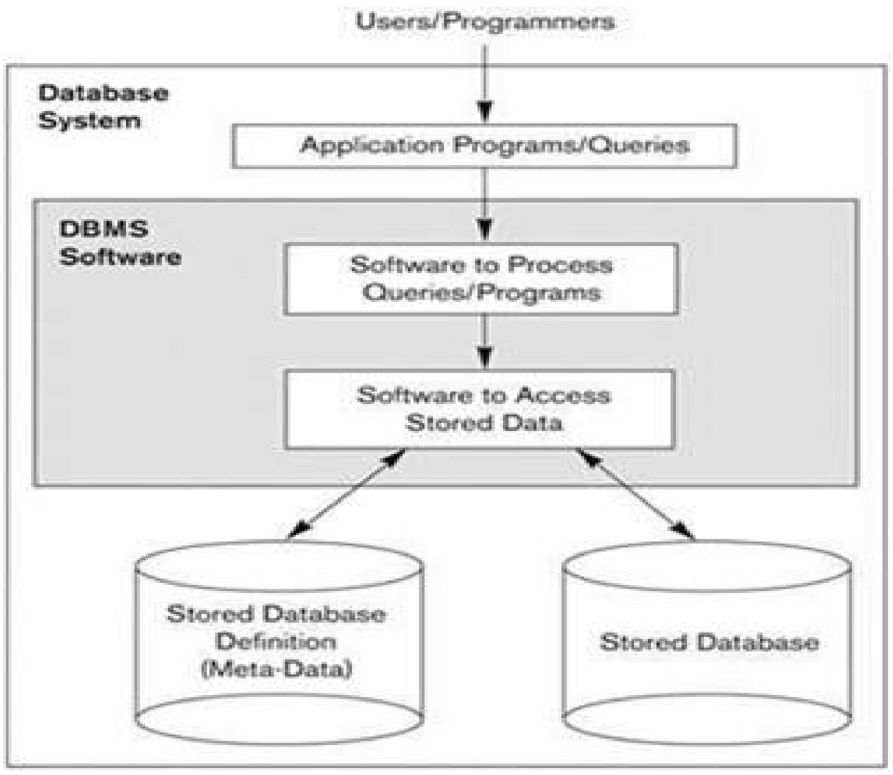

Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Fig: Simplified view of DBMS

1.2.2 ARCHITECTURE OF DBMS

Three schema architecture: The goal of the three-schema architecture illustrated in the figure is to separate the user application from the physical database. In this architecture, schemas can be defined at the following three levels:

1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
2. The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.
3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of a database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on external schema design in a high-level data model.

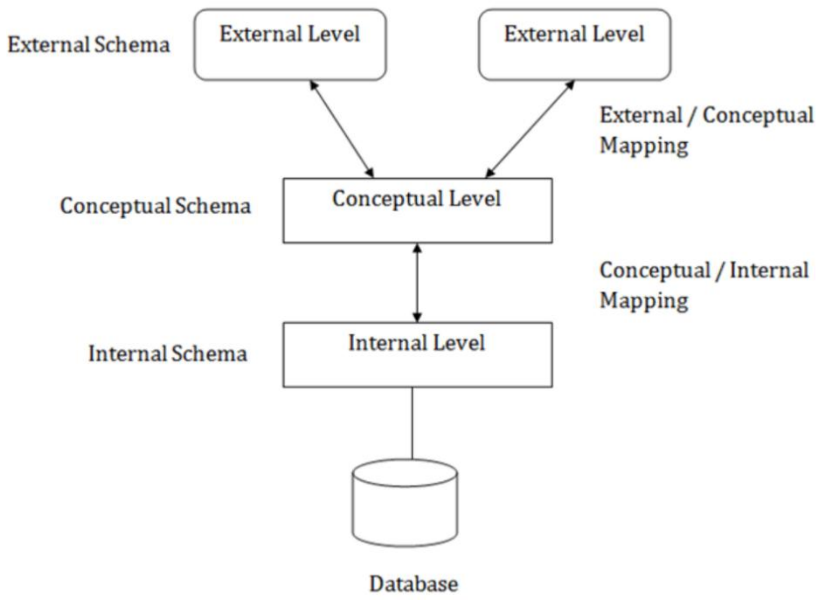


Fig: Three schema architecture of DBMS

1.2.3 ADVANTAGES OF USING DBMS

Using a DBMS to manage data has many advantages:

1. Data Independence: The application program should not, ideally, be expected to details of data representation and storage, the DBMS provides an abstract view of the data that hides such details.
2. Efficient Data Access: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is to be stored on an external device.
3. Data Integrity and Security: If data is always accessed through DBMS, the DBMS can enforce integrity constraints. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, it can enforce access controls that govern what data is visible to different classes of users.
4. Data Administration: When several users share data, centralizing the administration of data can offer significant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, it can be responsible for organizing the data representation to minimize redundancy and for fine-tuning the storage of the data to make retrieval efficient.
5. Concurrent Access and Crash Memory: A DBMS schedules concurrent access to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.
6. Backup and Recovery: Database Management System automatically takes care of backup and recovery. The users don't need to back up data periodically because this is taken care of by the DBMS. Moreover, it also restores the database after a crash or system failure to its previous condition.
7. Sharing of Data: In a database, the users of the database can share the data among themselves. There are various levels of authorization to access the data, and consequently the data can only be shared based on the correct authorization protocols being followed. Many remote users can also access the database simultaneously and share the data between themselves.

1.3 Introduction to HTML (HyperText Markup Language)

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs,lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <p>, <input>, etc. Browsers do not display the HTML tags, but use them to interpret the content of the page.

ADVANTGES OF HTML:

- HTML is easy to use and understand
- All browsers support HTML
- HTML and XML syntax is very similar
- Most development tools support HTML
- HTML is most search engine friendly

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

1.4 Introduction to CSS (Cascading Style Sheets)

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering

methods, such as on-screen on a computer monitor but also for alternate formatting if the content is accessed on a mobile device.
The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.
The CSS specifications are maintained by the World Wide Web Consortium (W3C). The W3C operates a free CSS validation service for CSS documents

1.5 Introduction to JavaScript (JS)

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.
Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.
As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). However, the language itself does not include any input/output (I/O), such as networking, storage, or graphics facilities, as the host environment (usually a web browser) provides those APIs.
JavaScript engines were originally used only in web browsers, but they are now embedded in some servers, usually via Node.js. They are also embedded in a variety of applications created with frameworks such as Electron and Cordova.

1.6 Introduction to PHP (Hypertext PreProcessor)

PHP is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control. Arbitrary PHP code can also be interpreted and executed via command-line interface (CLI).
The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and

platform, free of charge.

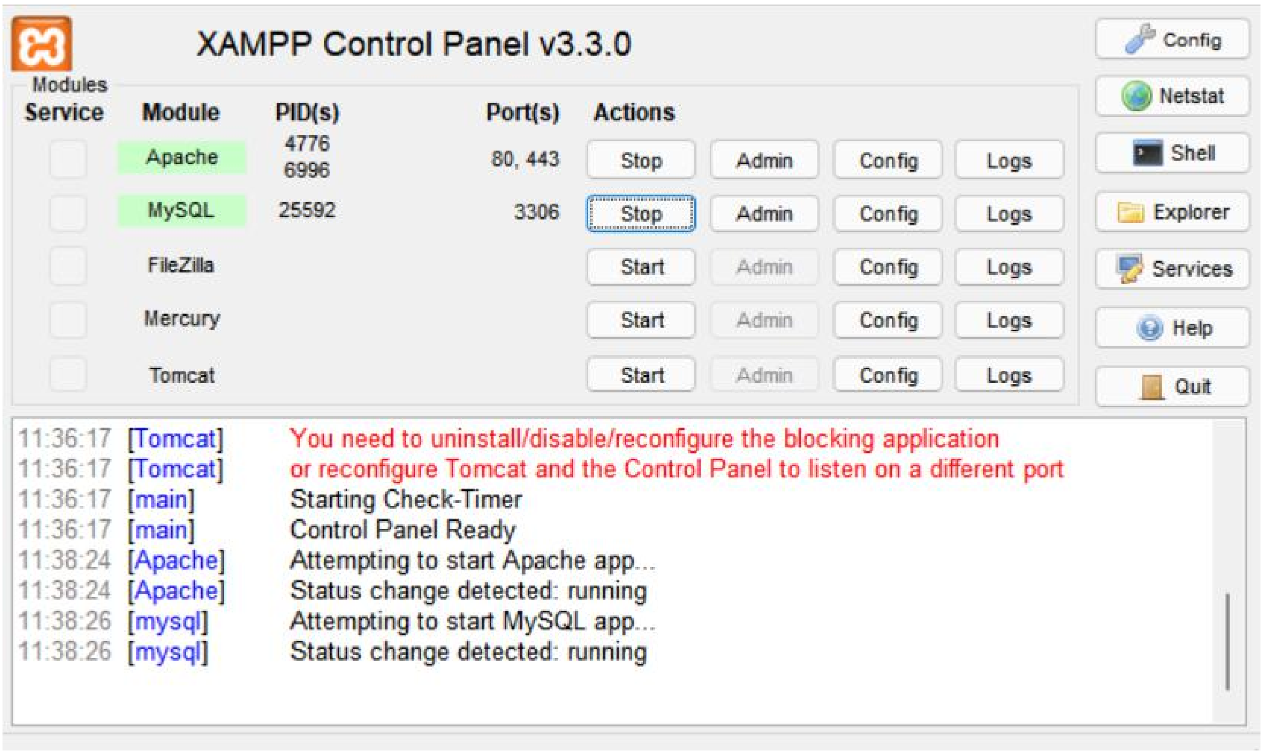
The PHP language evolved without a written formal specification or standard until 2014, with the original implementation acting as the de facto standard which other implementations aimed to follow. Since 2014, work has gone on to create a formal PHP specification

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what is going on behind scene.

1.7 Introduction to XAMPP

XAMPP is an abbreviation where X stands for Cross-Platform, A stands for Apache, M stands for MYSQL, and the P(s) stand for PHP and Perl, respectively. It is an open-source package of web solutions that includes Apache distribution for many servers and command-line executables along with modules such as Apache server, MariaDB, PHP, and Perl.

XAMPP helps a local host or server to test its website and clients via computers and laptops before releasing it to the main server. It is a platform that furnishes a suitable environment to test and verify the working of projects based on Apache, Perl, MySQL database, and PHP through the system of the host itself. Among these technologies, Perl is a programming language used for web development, PHP is a backend scripting language, and MariaDB is the most vividly used database developed by MySQL. The detailed description of these components is given below.



Service	Module	PID(s)	Port(s)	Actions			
	Apache	4776 6996	80, 443	Stop	Admin	Config	Logs
	MySQL	25592	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
11:36:17  [Tomcat]    You need to uninstall/disable/reconfigure the blocking application
11:36:17  [Tomcat]    or reconfigure Tomcat and the Control Panel to listen on a different port
11:36:17  [main]      Starting Check-Timer
11:36:17  [main]      Control Panel Ready
11:38:24  [Apache]    Attempting to start Apache app...
11:38:24  [Apache]    Status change detected: running
11:38:26  [mysql]     Attempting to start MySQL app...
11:38:26  [mysql]     Status change detected: running
```

Fig: XAMPP Control Panel

1.8: Introduction to MYSQL:

MYSQL is an open source relational database management system (RDBMS) based on Structured Query Language (SQL). MYSQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MYSQL is most often associated with web-based applications and online publishing and is an important component of an open source enterprise stack called LAMP. LAMP is a Web development platform that uses Linux as the operating system, Apache as the Web server, MYSQL as the relational database management System and PHP the object-oriented scripting language.(Sometimes Perl or Python is used instead of PHP).MYSQL is a full-featured relational database management system (RDBMS) that competes with the likes of Oracle DB and Microsoft's SQL Server. MYSQL is sponsored by the Swedish company MYSQL AB, which is owned by Oracle Corp.

Following are the basic syntaxes of MySQL with examples:

CREATE:-

CREATE TABLE table_name (column_name column_type);

INSERT:-

INSERT INTO table_name (field1, field2,…. fieldN) VALUES (value1, value2,….valueN) ; To insert string data types, it is required to keep all the values into double or single quotes.

SELECT:-

SELECT field1, field2…fieldN from table_name1, table_name2 [WHERE Clause];

DELETE:-

DELETE FROM table_name [WHERE Clause];

UPDATE:-

UPDATE table_name

SET attribute = 'value' [WHERE Clause]

2. REQUIREMENT SPECIFICATION

2.1 FUNCTIONAL REQUIREMENTS

The software is used for apartment management. The system should satisfy the following requirements:

1. Admin Sign in - The system must allow existing users to sign in to their accounts and continue using their account.

2. Add, remove, update, and search for buildings and apartments - The system must allow the user to add, remove, update and search the apartments for rent and lease.

3. Admin Sign out - The system must allow existing users to sign out of their accounts and quit the application.

2.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are not mandatory, which means that they are not compulsory to be fulfilled. Some non-functional requirements of the system are:

1. The system must be portable and should work on most of the platforms.

2. Each page must load within 2 seconds.

3. The system must be display accurate data without any errors.

4. Database security must meet the standard requirements.

2.3 HARDWARE REQUIREMENTS

The selection of hardware configuration is an important task related to the software development insufficient random-access memory may affect adversely on the speed and efficiency of the entire system. The process should be powerful to handle the entire operations. The hard disk should have sufficient capacity to store the file and application:

1. Processor: Device with Intel i3, 7th gen processor, an equivalent AMD processor or higher
2. RAM: 8 GB
3. Peripherals: Keyboard, Compatible mouse.
4. Hard Disk: 2 GB or higher of long term storage space on a hard drive or solid state drive

2.4 SOFTWARE REQUIREMENTS

A major element in building a system is the section of compatible software since the software in the market is experiencing in geometric progression. Selected software should be acceptable by the firm and one user as well as it should be feasible for the system.

This document gives a detailed description of the software requirement specification. The study of requirement specification is focused specially on the functioning of the system. It allows the developer or analyst to understand the system, function to be carried out the performance level to be obtained and corresponding interfaces to be established.

- HTML (HyperText Markup Language)
- CSS (Cascading Style Sheets)
- JavaScript
- PHP for server connectivity
- Text Editor such as Visual Studio Code, Atom, Sublime, etc

MySQL server such as Apache, Nginx The PHP, database GUI and the server required can be utilized in the form of a development stack such as MAMP (Mac, Apache, MySQL, PHP) or XAMPP (Cross Platform, Apache, MySQL/MariaDB, Perl, PHP)

3. SYSTEM DESIGN

3.1 ARCHITECTURE

Apartment management system consists of the user unit, which is the main part responsible for the management of the system. The functions of the user are to:

1. Add, remove, update and search the apartment, buildings and tenants.
2. Add, remove, update and search for a tenant who has rented or leased an apartment.

The architecture of the system is shown below.

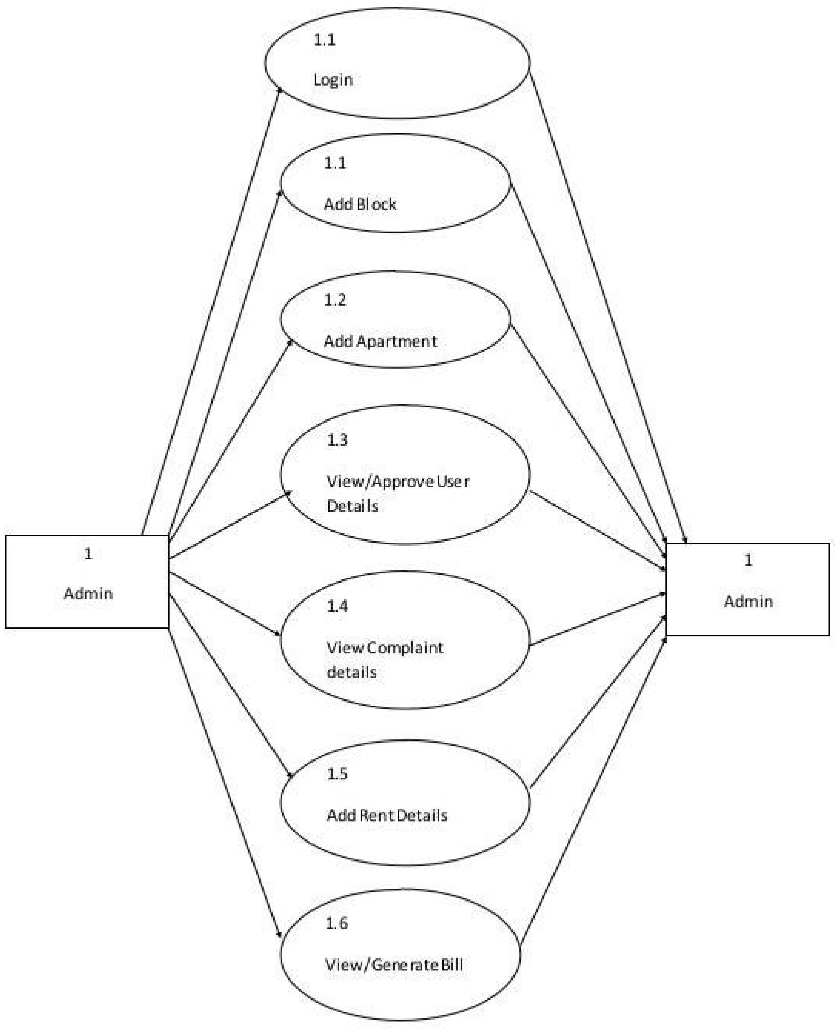


Fig: Data flow diagram

The above diagram shows the workflow of the system when a user is using the system.

3.2 FLOWCHART

A Flowchart is simply a graphical representation of steps. It shows steps in sequential order and is widely used in presenting the Now of algorithms, workflow or processes. Typically, a flowchart shows the steps as boxes of various kinds, and their order by connecting them with arrows.

The flowchart for the apartment management system is shown below.

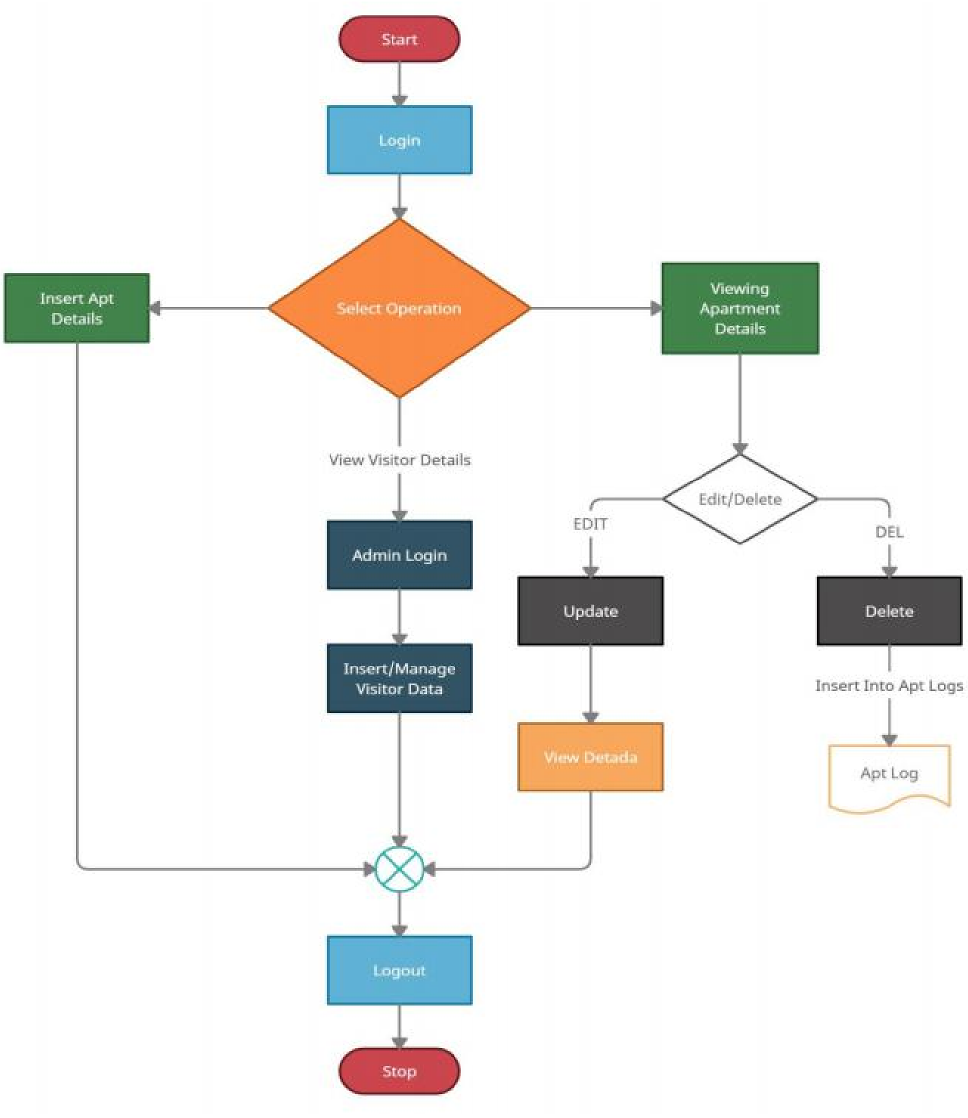


Fig: Flowchart of Project

In the above flowchart, we can see how the user interacts with the various components of the software to get the required data from the database.

3.3 ENTITY RELATIONSHIP DIAGRAM

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

It allows us to describe the data involved in a real-world enterprise in terms of objects and their relationship, widely used to develop an initial database design. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. ER diagrams are used to sketch out the design of a database.

The ER Diagram for the apartment management system is shown below.

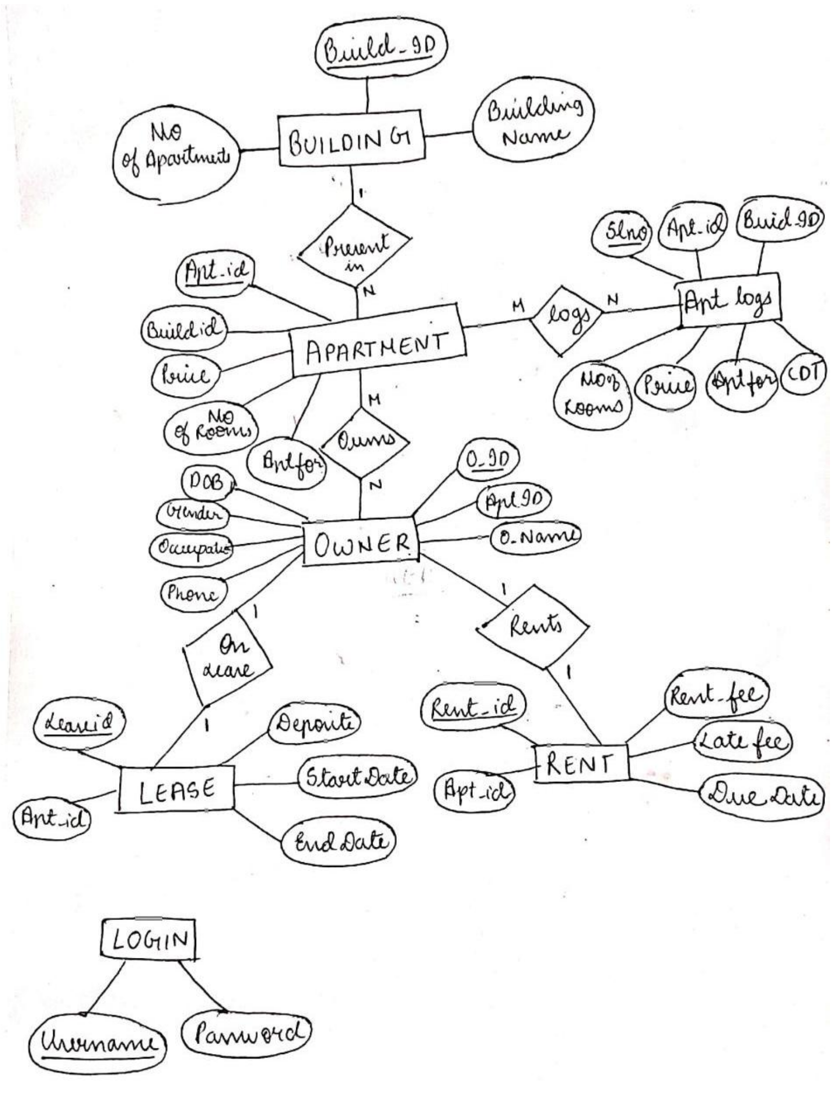


Fig: Entity Relational Diagram

In this entity relationship diagram shown above, the entities are rent, owner, apartment, building and lease. Here the entities are represented by rectangles, attributes of the tables are represented by ovals and relationships are represented using diamonds.

3.4 SCHEMA DIAGRAM

The design of the database is called a schema. A database schema can be represented in a visual diagram, which shows the database object and their relationship, which represents the logical view of the database and how the relationships among them are represented.

A database schema defines how the data is organized using the schema diagram. A schema diagram only shows us the database design.

The schema diagram of the apartment management system is shown below.

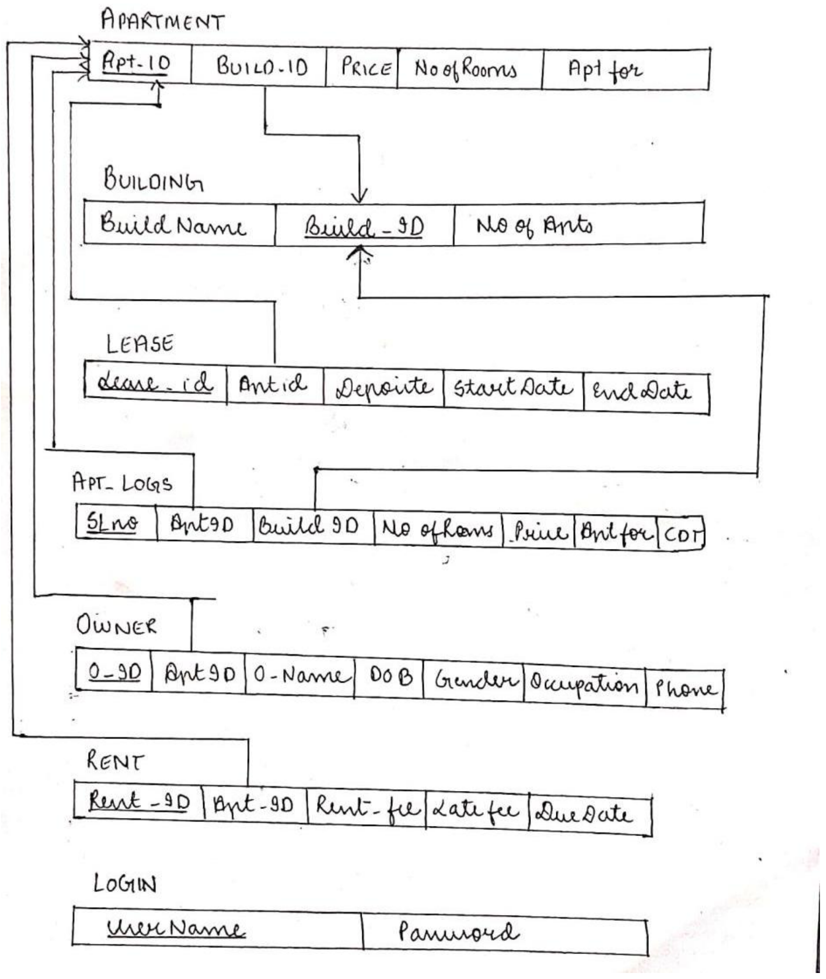


Fig: Schema Diagram

The schema diagram above represents different tables used, and underlined attributes are primary keys and arrows are used to represent foreign keys.

3.5 RELATIONAL SCHEMA DIAGRAM

A relational schema is a blueprint used in database design to represent the data to be entered into the database and describe how that data is structured in tables (called relations in relational schemas). The schema describes how those tables relate to each other.

In the relational schema, the table, or relation, consists of a set of named, but unsorted, columns (called attributes in relational schemas) and an undefined number of unnamed and unsorted rows (called tuples in relational schemas). Each row is unique, but the rows can be moved around as needed and stored in any order, modified, or deleted without impacting the efficient operation of the database.

The relational schema diagram of the apartment management system is shown below.

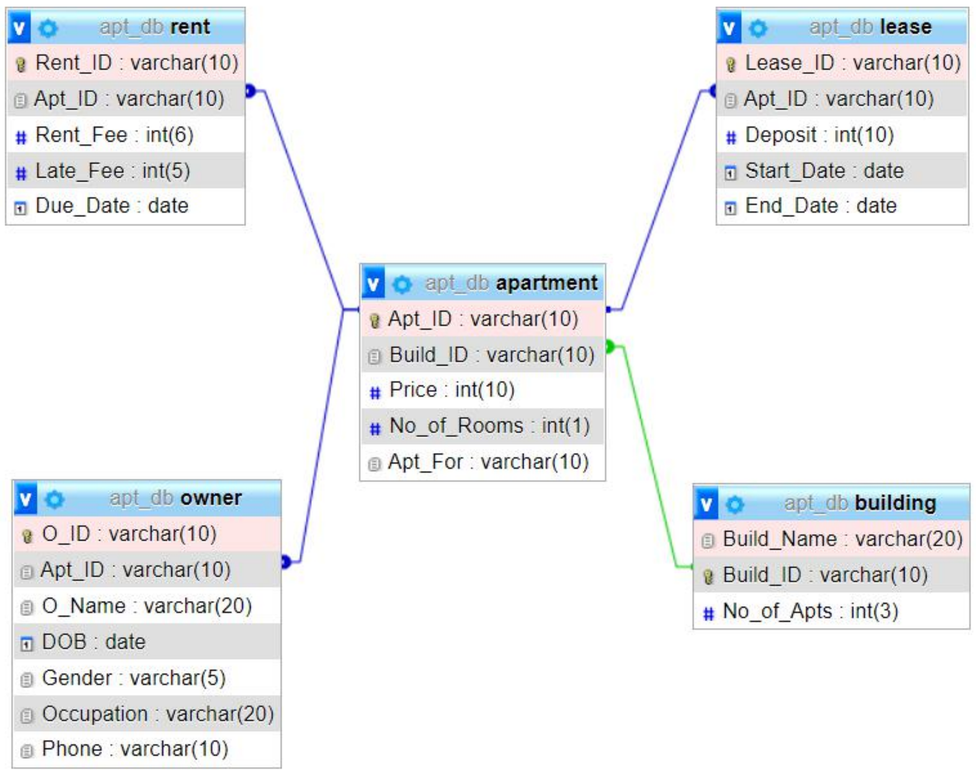


Fig: Relational Schema diagram

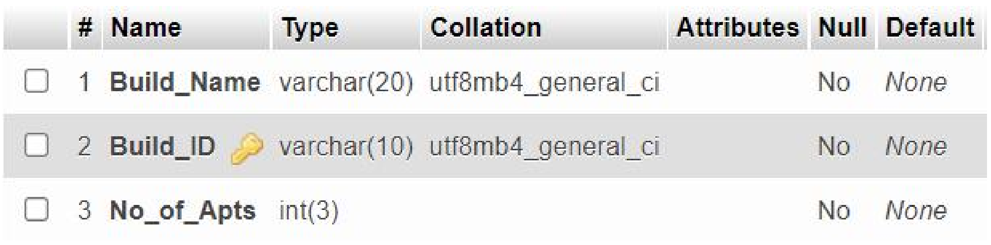
The relational schema diagram above represents different tables used, and first attributes are primary keys and arrows are used to represent foreign keys.

4. IMPLEMENTATIONS

4.1 TABLE DESCRIPTION

4.1.1 BUILDINGS TABLE

The buildings table has the attributes build_name, build_id and no_of_apt. The build_id is used as primary key as shown below.

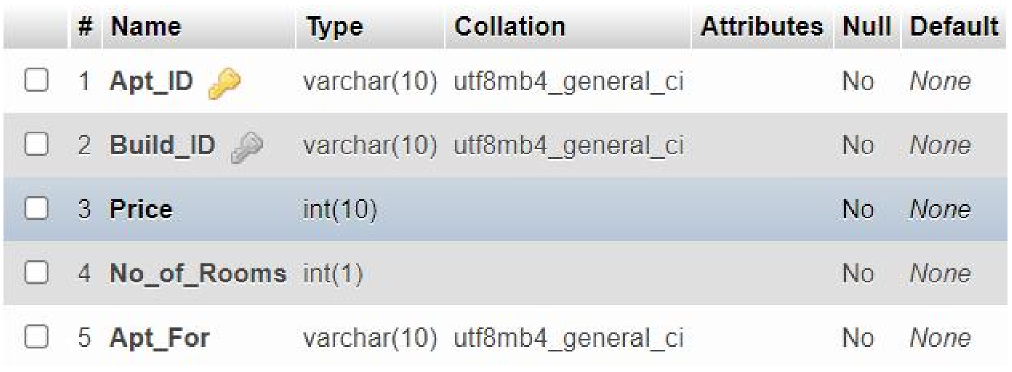


	#	Name	Type	Collation	Attributes	Null	Default
☐	1	Build_Name	varchar(20)	utf8mb4_general_ci		No	None
☐	2	Build_ID 🔑	varchar(10)	utf8mb4_general_ci		No	None
☐	3	No_of_Apts	int(3)			No	None

Fig: Building Table

4.1.2 APARTMENT TABLE

The apartment table has the attributes apt_id, build_id, price, no_of_rooms and apt_for. The apt_id is used as primary key as shown below.

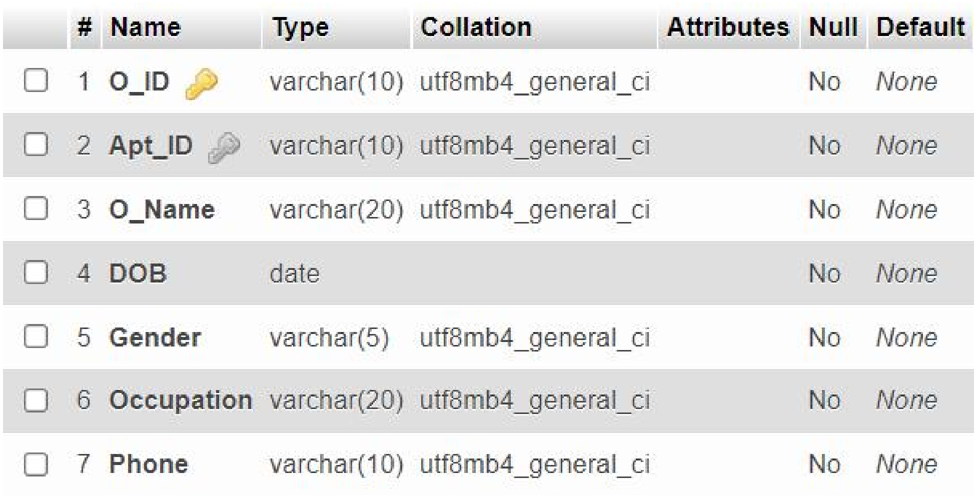


	#	Name	Type	Collation	Attributes	Null	Default
☐	1	Apt_ID 🔑	varchar(10)	utf8mb4_general_ci		No	None
☐	2	Build_ID 🔑	varchar(10)	utf8mb4_general_ci		No	None
☐	3	Price	int(10)			No	None
☐	4	No_of_Rooms	int(1)			No	None
☐	5	Apt_For	varchar(10)	utf8mb4_general_ci		No	None

Fig: Apartment Table

4.1.3 OWNER TABLE

The owner table has the attributes o_id, apt_id, o_name, dob, gender, occupation and phone. The o_id is used as primary key and apt_id is used as foreign key as shown below.

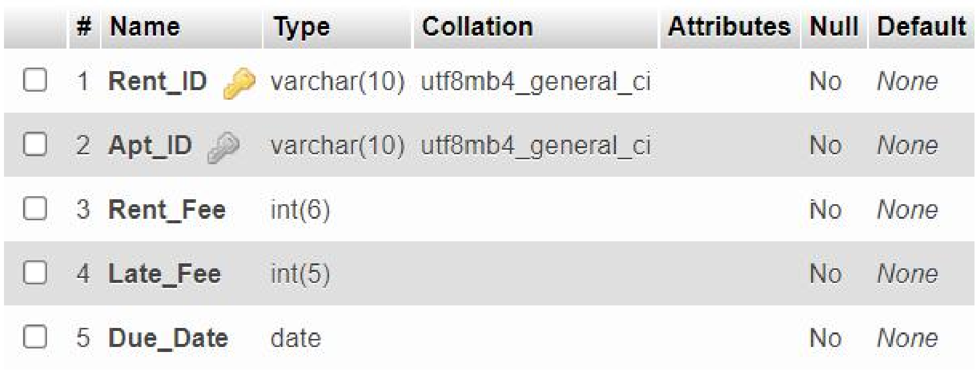


	#	Name	Type	Collation	Attributes	Null	Default
☐	1	O_ID	varchar(10)	utf8mb4_general_ci		No	None
☐	2	Apt_ID	varchar(10)	utf8mb4_general_ci		No	None
☐	3	O_Name	varchar(20)	utf8mb4_general_ci		No	None
☐	4	DOB	date			No	None
☐	5	Gender	varchar(5)	utf8mb4_general_ci		No	None
☐	6	Occupation	varchar(20)	utf8mb4_general_ci		No	None
☐	7	Phone	varchar(10)	utf8mb4_general_ci		No	None

Fig: Owner table

4.1.4 RENT TABLE

The rent table has the attributes rent_id, apt_id, rent_fee, late_fee and due_fee. The rent_id is used as primary key and apt_id is used as foreign key as shown below.

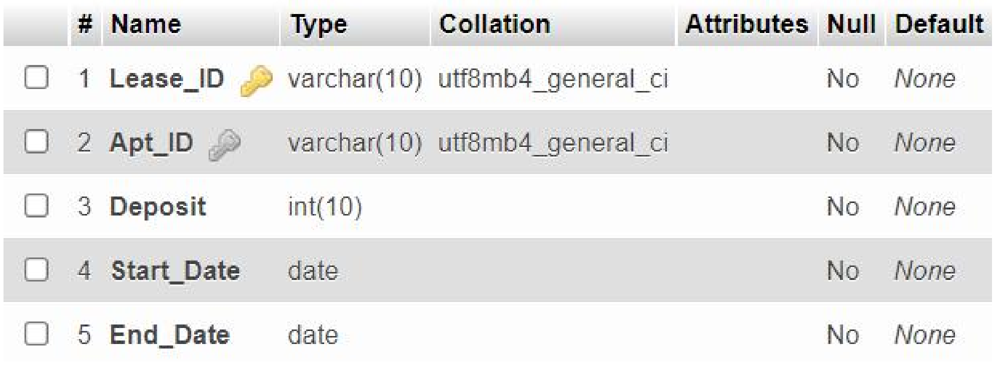


	#	Name	Type	Collation	Attributes	Null	Default
☐	1	Rent_ID	varchar(10)	utf8mb4_general_ci		No	None
☐	2	Apt_ID	varchar(10)	utf8mb4_general_ci		No	None
☐	3	Rent_Fee	int(6)			No	None
☐	4	Late_Fee	int(5)			No	None
☐	5	Due_Date	date			No	None

Fig: Rent table

4.1.5 LEASE TABLE

The lease table has the attributes lease_id, apt_id, deposit, start_date and end_date. The lease_id is used as primary key and apt_id is used as foreign key as shown below.



#	Name	Type	Collation	Attributes	Null	Default
1	Lease_ID	varchar(10)	utf8mb4_general_ci		No	None
2	Apt_ID	varchar(10)	utf8mb4_general_ci		No	None
3	Deposit	int(10)			No	None
4	Start_Date	date			No	None
5	End_Date	date			No	None

Fig: Lease table

4.2 DATABASE IMPLEMENTATIONS

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101  SET  @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `apt_db`
--

DELIMITER $$
--
-- Procedures
--
CREATE DEFINER=`root`@`localhost` PROCEDURE `apt_on_sale` ()   NO SQL
SELECT * FROM apartment WHERE Apt_for = 'Sale'$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `getLogDetails` ()   NO SQL
SELECT * FROM apt_logs$$

DELIMITER ;

-- --------------------------------------------------------

--
-- Table structure for table `apartment`
--
```

```
CREATE TABLE `apartment` (
  `Apt_ID` varchar(10) NOT NULL,
  `Build_ID` varchar(10) NOT NULL,
  `Price` int(10) NOT NULL,
  `No_of_Rooms` int(1) NOT NULL,
  `Apt_For` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `apartment`
--

INSERT INTO `apartment` (`Apt_ID`, `Build_ID`, `Price`, `No_of_Rooms`, `Apt_For`)
VALUES
('N1001', 'B101', 3000000, 3, 'Sale'),
('N1002', 'B101', 4000000, 4, 'Rent'),
('N1003', 'B101', 2500000, 2, 'Lease'),
('P1001', 'B103', 5000000, 2, 'Sale'),
('P1002', 'B103', 6500000, 3, 'Rent'),
('P1003', 'B103', 7000000, 4, 'Lease'),
('P1004', 'B103', 10000000, 4, 'Sale'),
('S1001', 'B102', 5000000, 4, 'Sale'),
('S1002', 'B102', 3000000, 3, 'Rent'),
('S1003', 'B102', 2000000, 2, 'Lease');

--
-- Triggers `apartment`
--
DELIMITER $$
CREATE TRIGGER `del_trigger` BEFORE DELETE ON `apartment` FOR EACH ROW
INSERT INTO apt_logs VALUES (null, OLD.Apt_ID, OLD.Build_ID, OLD.No_of_Rooms,
OLD.Price, OLD.Apt_For, NOW())
$$
DELIMITER ;

-- --------------------------------------------------------

--
-- Table structure for table `apt_logs`
--

CREATE TABLE `apt_logs` (
  `SL_No` int(11) NOT NULL,
  `Apt_ID` varchar(10) NOT NULL,
  `Build_ID` varchar(10) NOT NULL,
  `No_of_Rooms` int(11) NOT NULL,
  `Price` int(10) NOT NULL,
  `Apt_For` varchar(10) NOT NULL,
  `CDT` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
```

```
-- Dumping data for table `apt_logs`
--

INSERT INTO `apt_logs` (`SL_No`, `Apt_ID`, `Build_ID`, `No_of_Rooms`, `Price`, `Apt_For`,
`CDT`) VALUES
(4, 'A1001', 'B100', 3, 3000000, 'Sale', '2019-11-29 12:51:09');

-- --------------------------------------------------------

--
-- Table structure for table `building`
--

CREATE TABLE `building` (
  `Build_Name` varchar(20) NOT NULL,
  `Build_ID` varchar(10) NOT NULL,
  `No_of_Apts` int(3) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `building`
--

INSERT INTO `building` (`Build_Name`, `Build_ID`, `No_of_Apts`) VALUES
('Noble Apartment', 'B101', 10),
('Shakuntala Nilaya', 'B102', 5),
('Prestige', 'B103', 10);

-- --------------------------------------------------------

--
-- Table structure for table `lease`
--

CREATE TABLE `lease` (
  `Lease_ID` varchar(10) NOT NULL,
  `Apt_ID` varchar(10) NOT NULL,
  `Deposit` int(10) NOT NULL,
  `Start_Date` date NOT NULL,
  `End_Date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `lease`
--

INSERT INTO `lease` (`Lease_ID`, `Apt_ID`, `Deposit`, `Start_Date`, `End_Date`) VALUES
('L1001', 'N1003', 1000000, '2017-11-30', '2020-11-30'),
('L1002', 'P1003', 1500000, '2015-09-01', '2019-09-01'),
('L1003', 'S1003', 1200000, '2019-06-05', '2021-07-05');

-- --------------------------------------------------------
```

```
--
-- Table structure for table `login`
--

CREATE TABLE `login` (
  `UserName` varchar(10) NOT NULL,
  `Password` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `login`
--

INSERT INTO `login` (`UserName`, `Password`) VALUES
('shoaib121', 'num');

-- --------------------------------------------------------

--
-- Table structure for table `owner`
--

CREATE TABLE `owner` (
  `O_ID` varchar(10) NOT NULL,
  `Apt_ID` varchar(10) NOT NULL,
  `O_Name` varchar(20) NOT NULL,
  `DOB` date NOT NULL,
  `Gender` varchar(5) NOT NULL,
  `Occupation` varchar(20) NOT NULL,
  `Phone` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `owner`
--

INSERT INTO `owner` (`O_ID`, `Apt_ID`, `O_Name`, `DOB`, `Gender`, `Occupation`,
`Phone`) VALUES
('1001', 'N1001', 'Imaad', '1999-10-20', 'Male', 'CSE Engg', '1231231234'),
('1002', 'N1002', 'Avinash', '1999-11-06', 'Male', 'CSE Engg', '1451451456'),
('1003', 'N1003', 'Vaishnavi', '1998-08-29', 'Femal', 'CSE Engg', '1789178914'),
('1004', 'S1001', 'Hansie', '1999-12-09', 'Male', 'Chef', '4455445544'),
('1005', 'S1002', 'Saicharan', '1999-05-05', 'Male', 'Petroleum Engg', '1785178514'),
('1006', 'S1003', 'Shubham', '1999-12-13', 'Male', 'EE Engg', '1456145614'),
('1007', 'P1001', 'Mamatha', '1988-02-18', 'Femal', 'Bussiness', '3213213214'),
('1008', 'P1002', 'Aamina', '1981-06-12', 'Femal', 'Civil Engg', '6546546541'),
('1009', 'P1003', 'Charls', '1989-06-14', 'Male', 'CSE Engg', '9876532145'),
('1010', 'P1004', 'Numaan', '1997-06-11', 'Male', 'Bussiness', '9008560282');

-- --------------------------------------------------------
```

```
--
-- Table structure for table `rent`
--

CREATE TABLE `rent` (
  `Rent_ID` varchar(10) NOT NULL,
  `Apt_ID` varchar(10) NOT NULL,
  `Rent_Fee` int(6) NOT NULL,
  `Late_Fee` int(5) NOT NULL,
  `Due_Date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `rent`
--

INSERT INTO `rent` (`Rent_ID`, `Apt_ID`, `Rent_Fee`, `Late_Fee`, `Due_Date`) VALUES
('R1001', 'N1002', 240000, 1000, '2021-06-16'),
('R1002', 'S1002', 200000, 1000, '2020-06-17'),
('R1003', 'P1002', 500000, 50000, '2019-12-30');

--
-- Indexes for dumped tables
--

--
-- Indexes for table `apartment`
--
ALTER TABLE `apartment`
  ADD PRIMARY KEY (`Apt_ID`),
  ADD KEY `Build_ID` (`Build_ID`);

--
-- Indexes for table `apt_logs`
--
ALTER TABLE `apt_logs`
  ADD PRIMARY KEY (`SL_No`);

--
-- Indexes for table `building`
--
ALTER TABLE `building`
  ADD PRIMARY KEY (`Build_ID`);

--
-- Indexes for table `lease`
--
ALTER TABLE `lease`
  ADD PRIMARY KEY (`Lease_ID`),
  ADD KEY `Apt_ID` (`Apt_ID`);

--
```

```
-- Indexes for table `login`
--
ALTER TABLE `login`
  ADD PRIMARY KEY (`UserName`);

--
-- Indexes for table `owner`
--
ALTER TABLE `owner`
  ADD PRIMARY KEY (`O_ID`),
  ADD KEY `Apt_ID` (`Apt_ID`);

--
-- Indexes for table `rent`
--
ALTER TABLE `rent`
  ADD PRIMARY KEY (`Rent_ID`),
  ADD KEY `Apt_ID` (`Apt_ID`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `apt_logs`
--
ALTER TABLE `apt_logs`
  MODIFY `SL_No` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `apartment`
--
ALTER TABLE `apartment`
  ADD CONSTRAINT `building_FK` FOREIGN KEY (`Build_ID`) REFERENCES `building`
(`Build_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `lease`
--
ALTER TABLE `lease`
  ADD CONSTRAINT `apartment_FK` FOREIGN KEY (`Apt_ID`) REFERENCES
`apartment` (`Apt_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `owner`
--
ALTER TABLE `owner`
  ADD CONSTRAINT `owner_ibfk_1` FOREIGN KEY (`Apt_ID`) REFERENCES
```

```
`apartment` (`Apt_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `rent`
--
ALTER TABLE `rent`
  ADD CONSTRAINT `apmt_FK` FOREIGN KEY (`Apt_ID`) REFERENCES `apartment`
(`Apt_ID`) ON DELETE CASCADE ON UPDATE CASCADE;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

4.3 PSEUDO CODE

4.3.1 HTML Code Snippet

HTML CODE FOR LOGIN PAGE

```
<!DOCTYPE html>
<html>
<head>
<title>Apartment DB</title>
<link rel="stylesheet" type="text/css" href="tablestyle.css"> </head>
<body>
<h1 title="By Shoaib And Imaad"> <bold>APARTMENT
MANAGEMENT SYSTEM</bold>
</h1>
<img src="Img/logo.jpg" alt="LOGO">
<div class="background">
<form action="login.php" method="post">
<input type="text" name="username" placeholder="ENTER USER ID" required> <br>
<input type="password" name="password" placeholder="ENTER PASSWORD"
required><br>
<button type="submit">LOGIN</button>
</form>
</div>
</body>
</html>
```

4.3.2 PHP Code Snippet

PHP CODE FOR LOGIN PAGE:

```
<?php
session_start();
// Connect to Database
$con = mysqli_connect("localhost", "root", ""); if(isset($_POST['username'])){
$username = $_POST['username'];
}
$password = "";
if(isset($_POST['username'])){
$password = $_POST['password'];
} APARTMENT MANAGEMENT SYSTEM
2020-21
//Select Database
mysqli_select_db($con,"apt_db");
// Query the database for user
$result = mysqli_query($con,"select * from login where UserName = '$username' and
Password = '$password'")
or die('Failed to query database'.mysqli_error());
$row = mysqli_fetch_array($result);
// user and password verification
if ( $row['UserName'] == $username && $row['Password'] == $password )
{
$_SESSION['uname'] = $username;
$message="Login Succesfull. Welcome ".$_SESSION['uname']."";
echo"<script > { alert('$message');}
window.location.replace('Menu.html');</script>";
}
else
{
$message="Credentials mismatch please try again";
echo"<script > { alert('$message');}
window.location.replace('login.html');</script>";
}
?>
```

4.3.3 CSS Code Snippet

```
body{
        margin: 0;
        padding: 0;
        font-family: sans-serif;
        background: url("Img/backg.jpg");
        background-repeat: no-repeat;
        background-size: cover;
}

.box h1{
        color: #fff;
        text-transform: uppercase;
        font-weight: 500;
}

.box{
        width: 500px;
        padding: 50px;
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%,-50%);
        background: #413529;
        text-align: center;
        border-radius: 100px;
}

.box button{
        width: 50%;
        border:0;
        background: none;
        margin: 20px auto;
        text-align: center;
        border: 2px solid #2ecc71;
        padding: 14px 40px;
        outline: none;
        color: white;
        border-radius: 24px;
        transition: 0.25s;
        cursor: pointer;
}
.box button:hover {
        width: 80%;
        background: #2ecc71;
}

.box input{
        border: 0;
        background: none;
        margin: 10px auto;
        text-align: center;
        border: 2px solid #3498db;
        padding: 14px 10px;
        width:200px;
        outline: none;
        color: white;
        border-radius: 24px;
```

```
    transition: 0.25s;
}

.box input:focus {
width:300px;
border-color: #191919;
}

.box label{
    color: grey;
}

button{
    background-color: #c4c3b899;
    border: none;
    color: white;
    padding: 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    border-radius: 10px;
}

button:hover{
    opacity: 0.8;
    background-color: #413529;
}
```

4.4 Source Code for Database Backend

4.4.1 Stored Procedure

A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs. Stored Procedures are used in the database to run a certain line or lines of code multiple times by calling that procedure in any number of .php files, any number of times. Our database has a stored procedure named 'apt_on_sale'.

```
--
-- Procedures
--
CREATE DEFINER=`root`@`localhost` PROCEDURE `apt_on_sale` ()  NO SQL
SELECT * FROM apartment WHERE Apt_for = 'Sale'$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `getLogDetails` ()  NO SQL
SELECT * FROM apt_logs$$

DELIMITER ;
```

4.4.2 Triggers

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

```
--
-- Triggers `apartment`
--
CREATE TRIGGER `del_trigger` BEFORE DELETE ON `apartment` FOR EACH ROW INSERT INTO apt_logs VALUES (null,
OLD.Apt_ID, OLD.Build_ID, OLD.No_of_Rooms, OLD.Price, OLD.Apt_For, NOW())
$$
DELIMITER ;
```

4.5 Normalization

Normalization is the process of reorganizing data in a database so that it meets two basic requirements: There is no redundancy of data, all data is stored in only one place. Data dependencies are logical,all related data items are stored together.

● Normalization is the process of organizing the data in the database.

● Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.

● Normalization divides the larger table into the smaller table and links them using relationships.

● The normal form is used to reduce redundancy from the database table.

1NF (First Normal Form) Rules

● Each table cell should contain a single value.

● Each record needs to be unique.

2NF (Second Normal Form) Rules

● Rule 1- Be in 1NF

● Rule 2- Single Column Primary Key that does not functionally dependant on any subset of candidate key Relation

3NF (Third Normal Form) Rules

● Rule 1- Be in 2NF

● Rule 2- Has no transitive functional dependencies.

Our project has a clear implementation of 2NF. There exists no multi-valued attributes as all the attributes stand independent from the other entities present in the database. There exists a primary key that is not functionally dependent to the candidate key of the table.

5. RESULTS

5.1 Screenshots

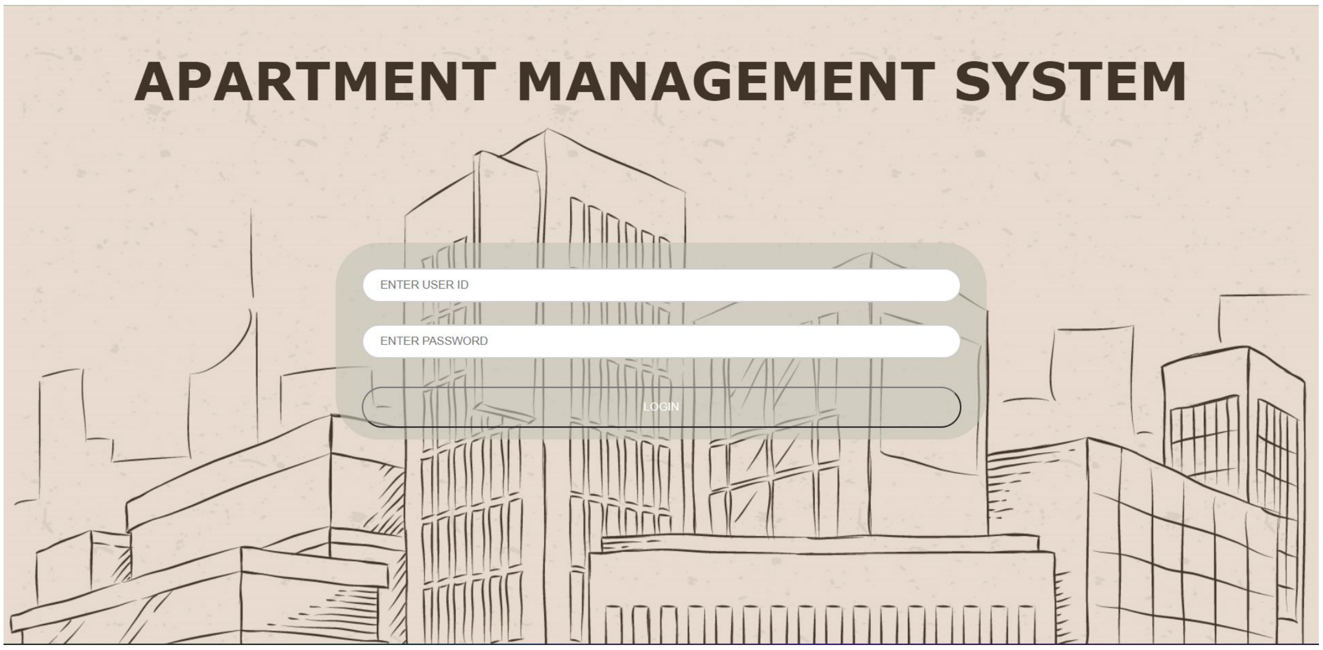


Fig: Login Page

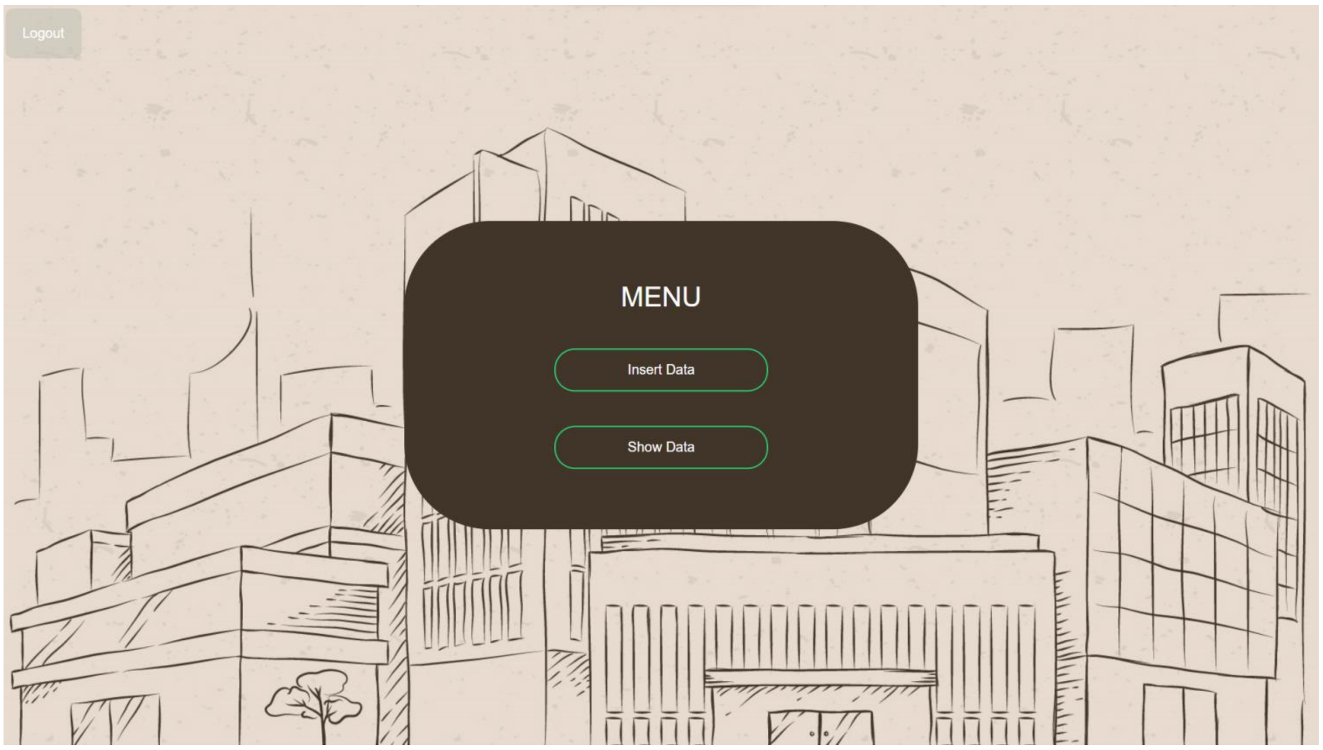


Fig: Home Page

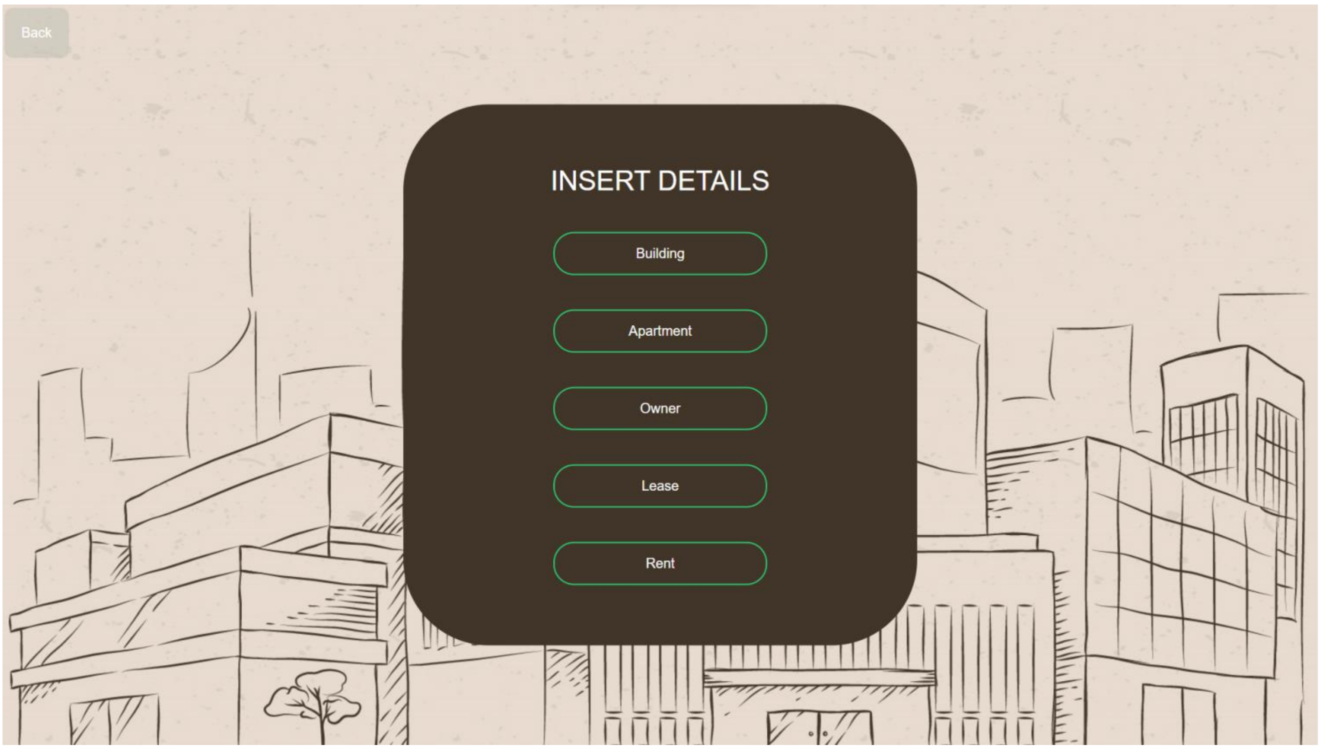


Fig: Insertion Page

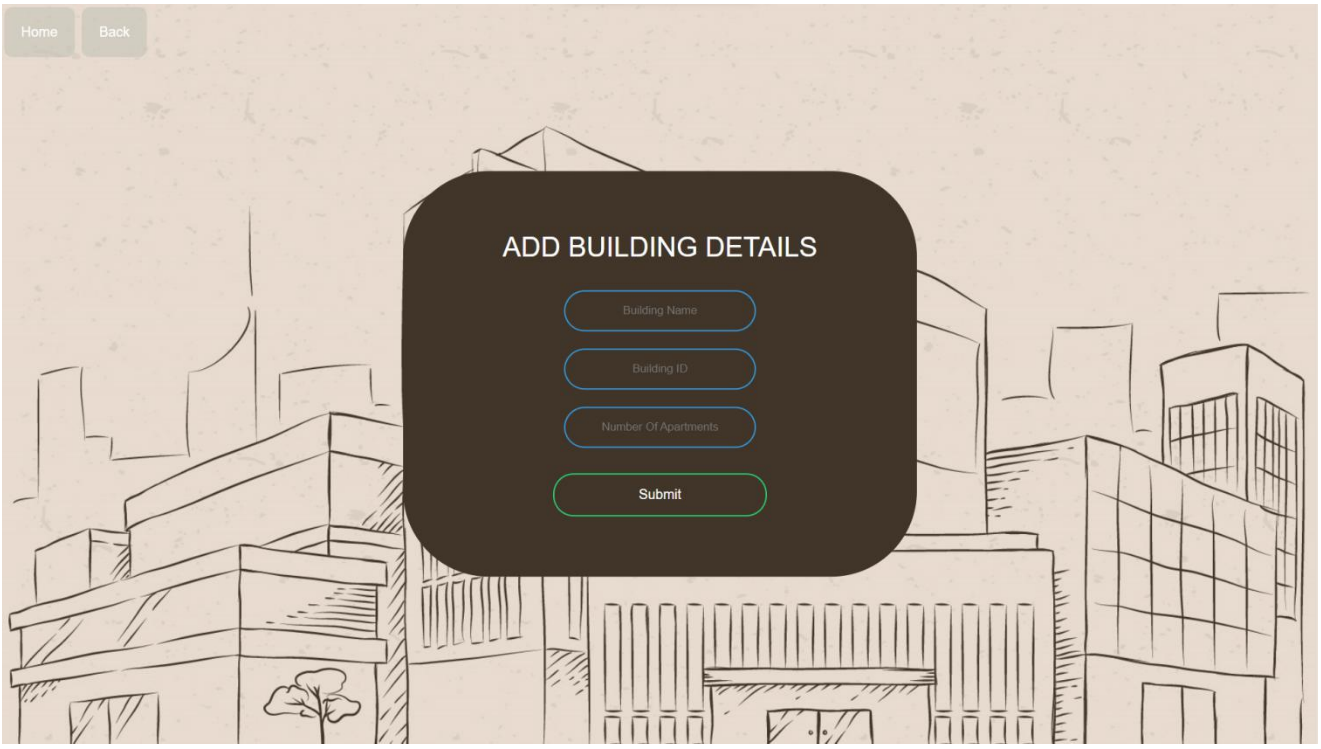
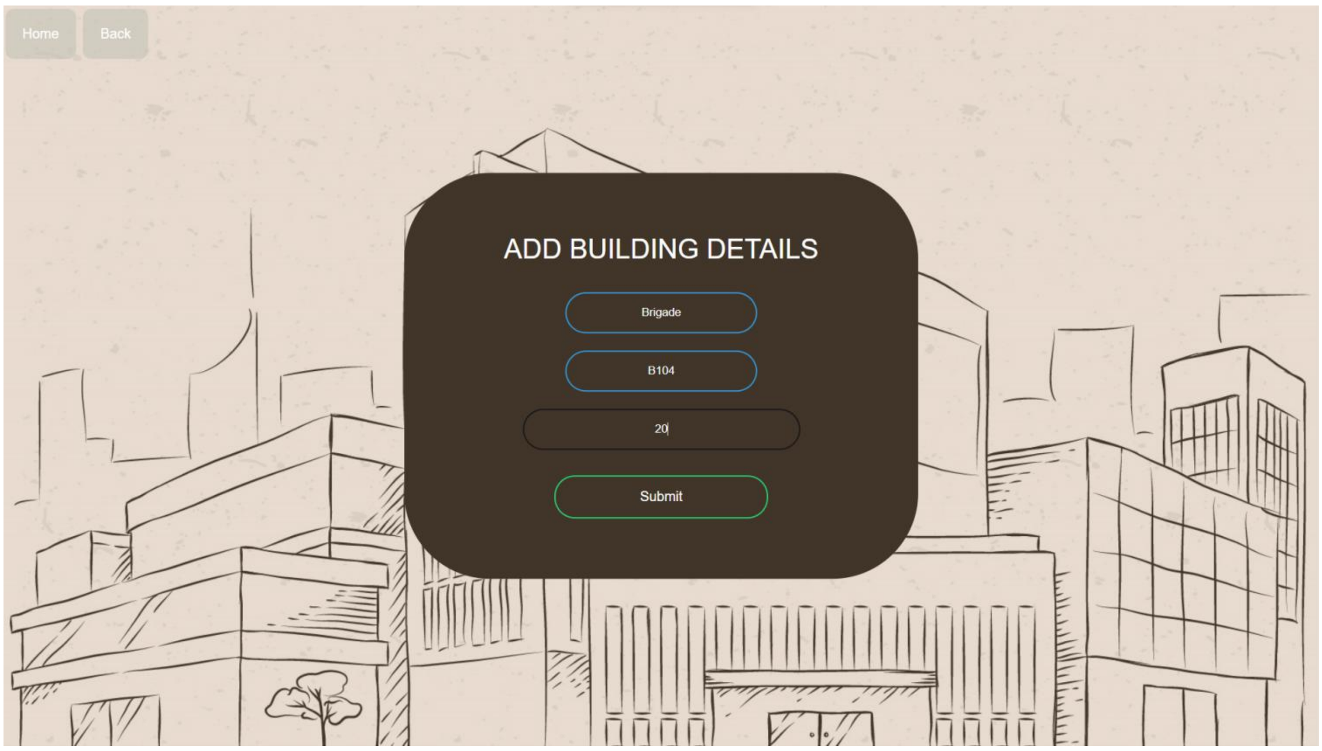


Fig: Insertion of Building Details

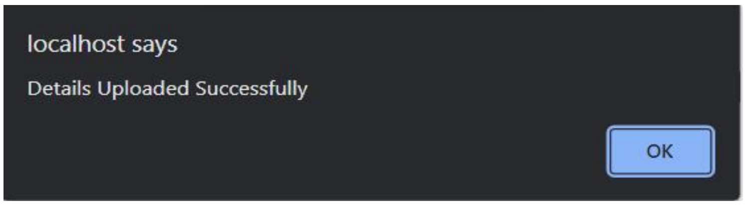


ADD BUILDING DETAILS

Brigade

B104

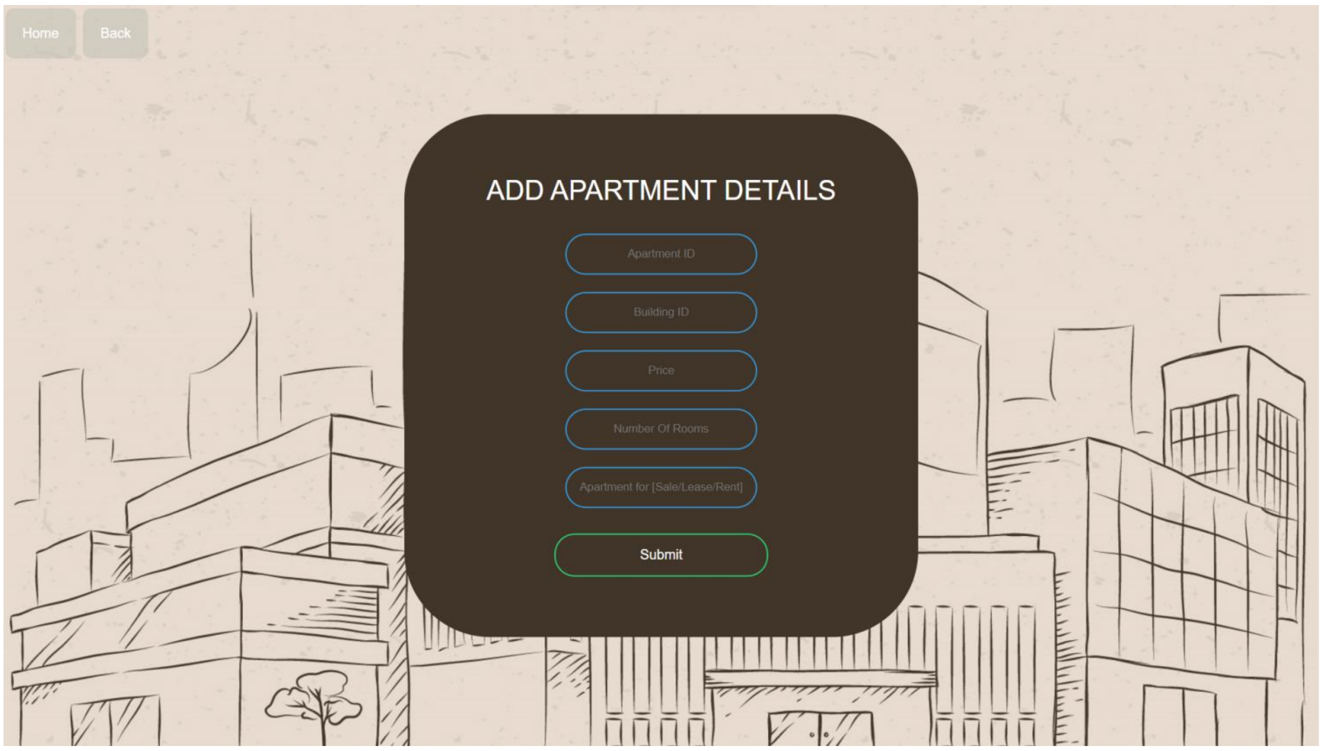
20



localhost says

Details Uploaded Successfully

Fig: Building Details Uploaded Successfully



ADD APARTMENT DETAILS

Apartment ID

Building ID

Price

Number Of Rooms

Apartment for [Sale/Lease/Rent]

Fig: Insertion of Apartment Details

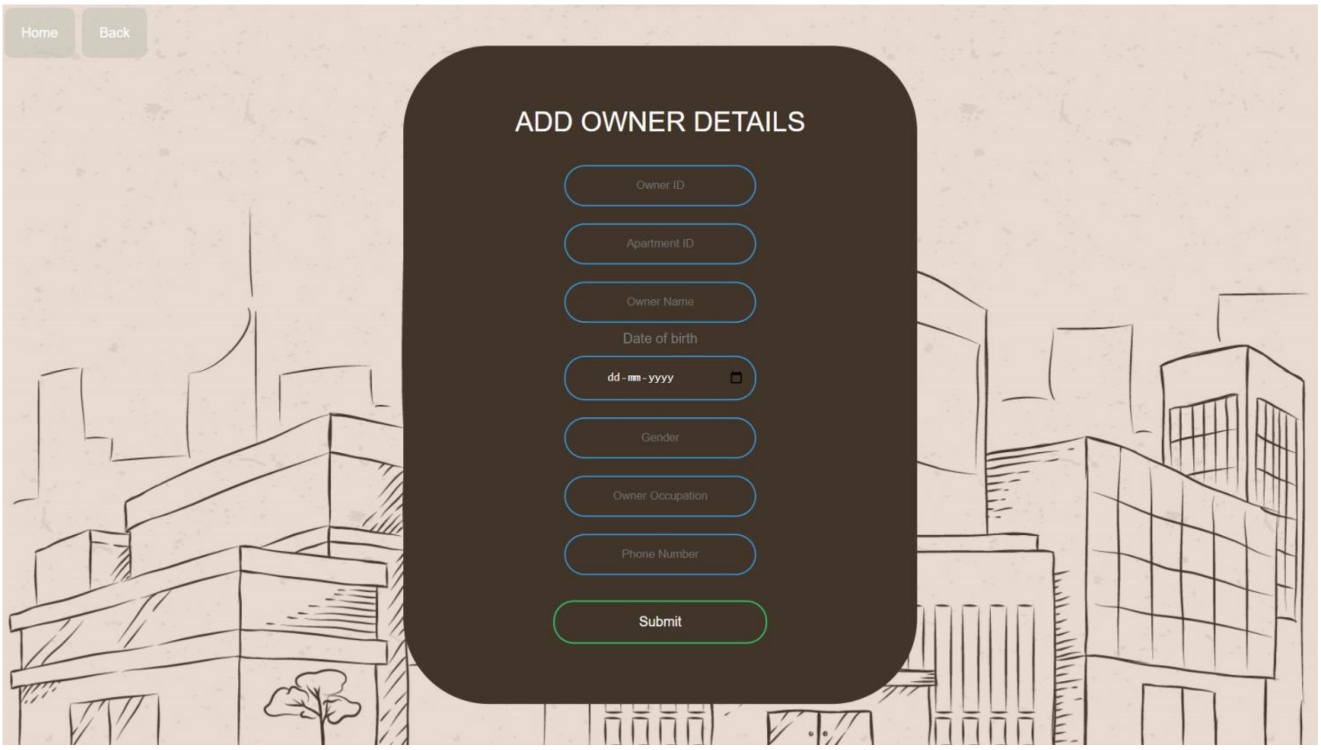


Fig: Insertion of Owner Details

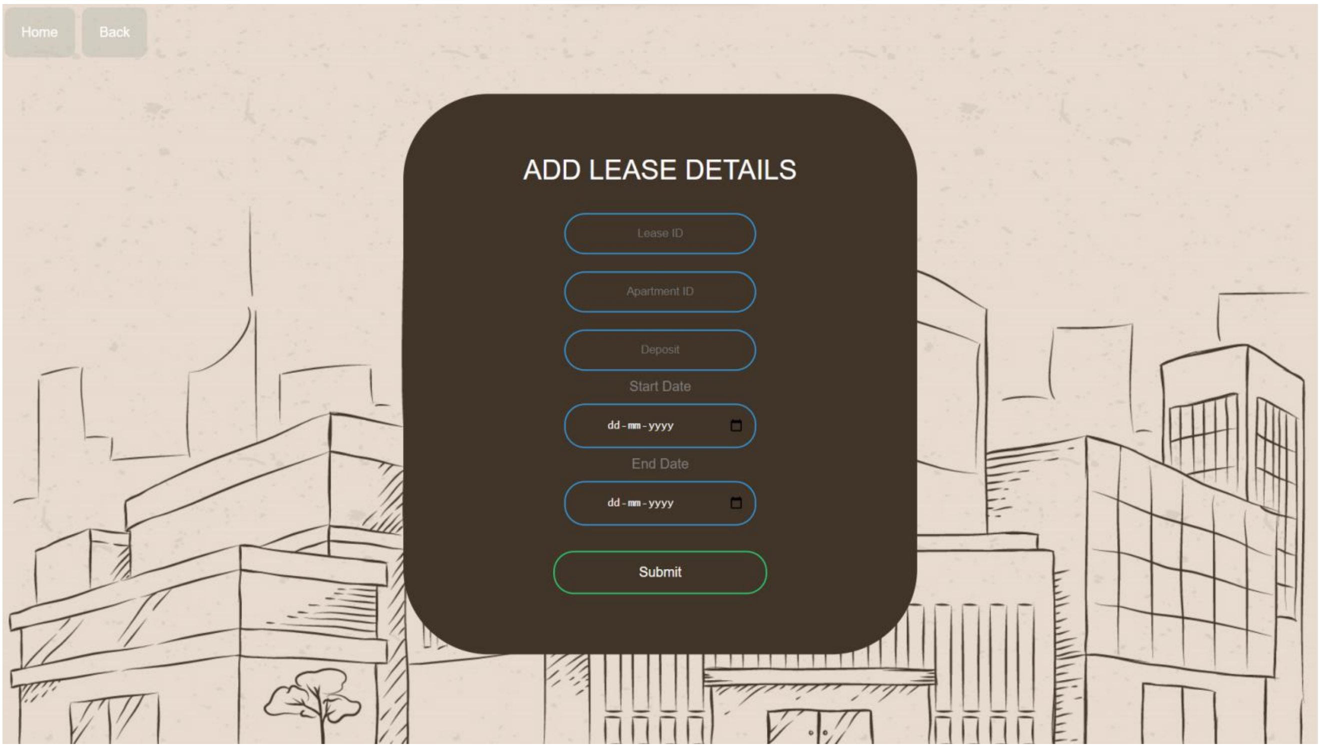


Fig: Insertion of Lease Details

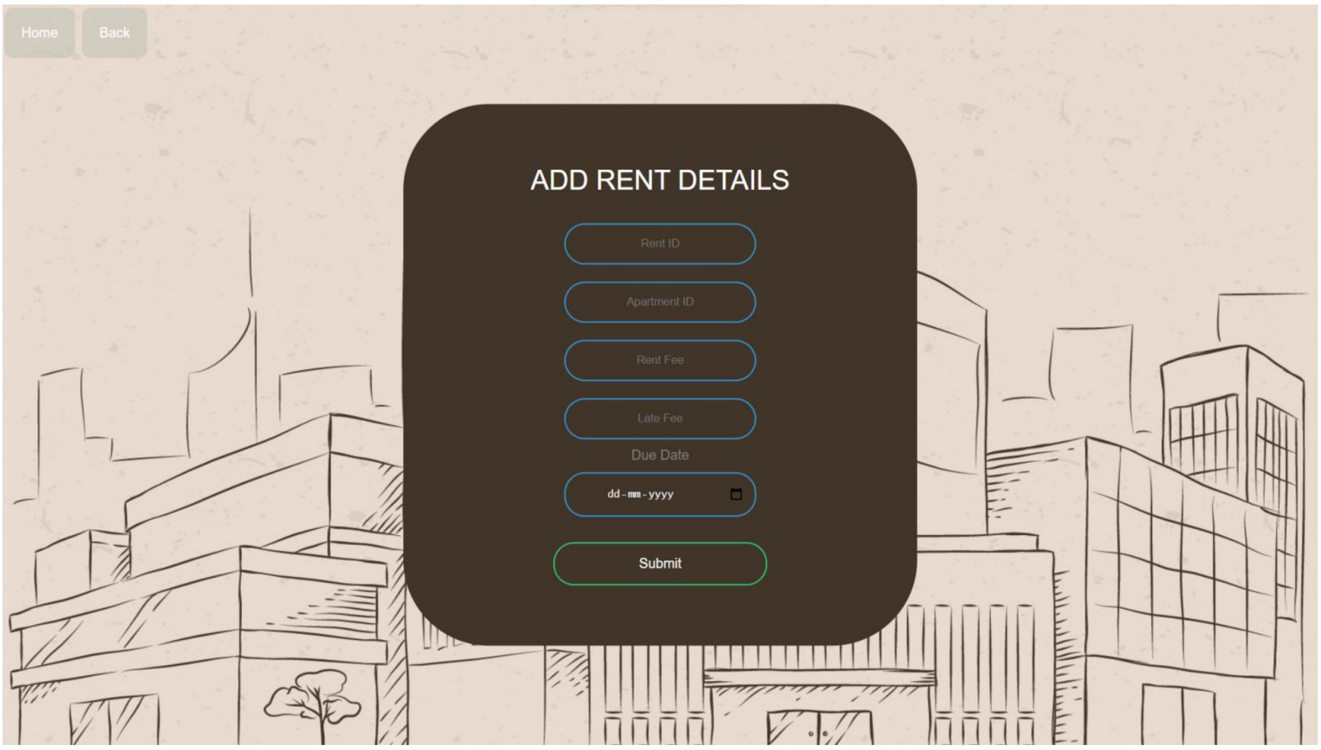


Fig: Insertion of Rent Details

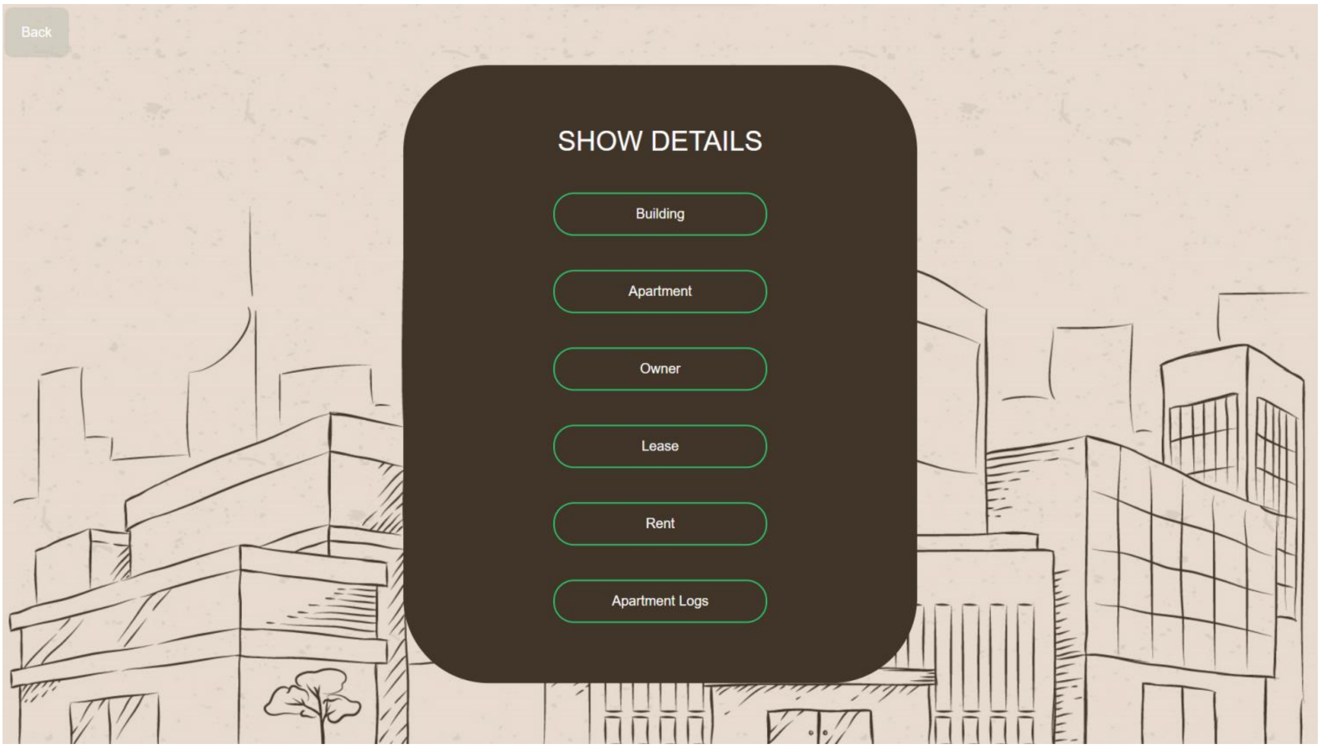
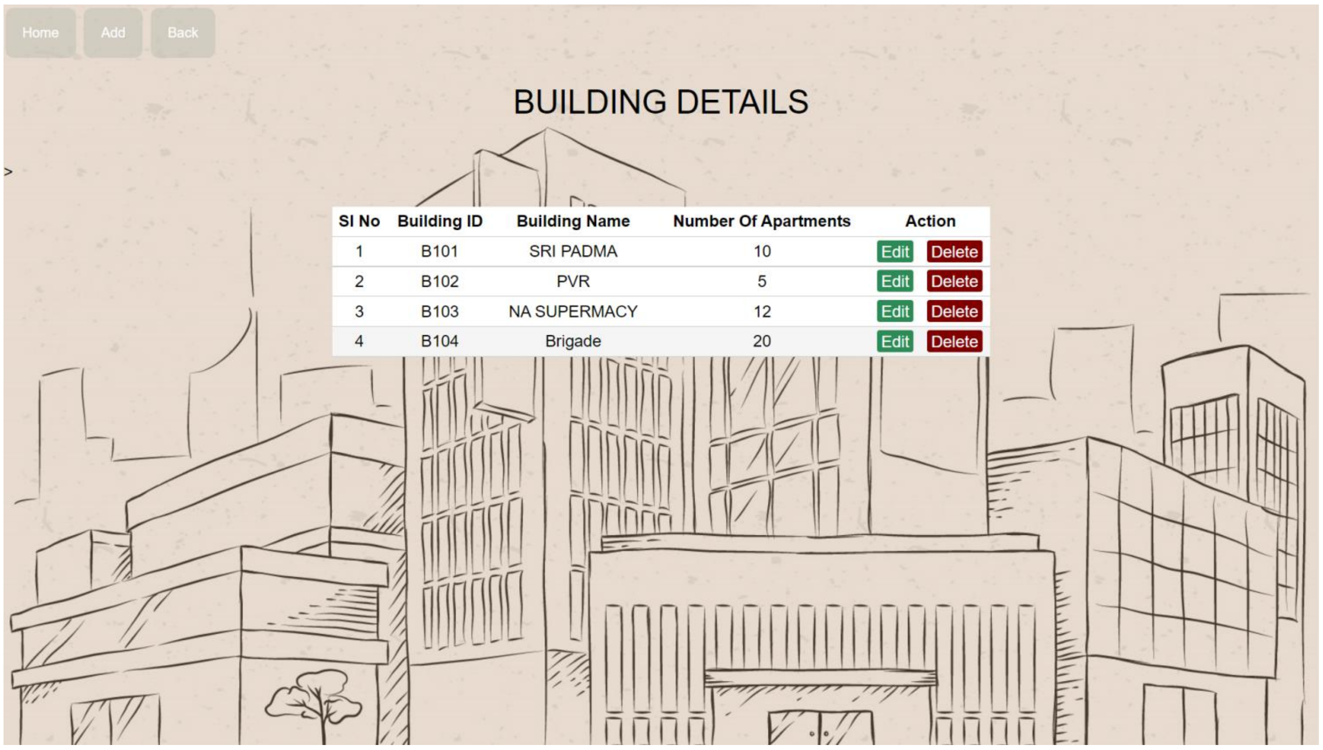


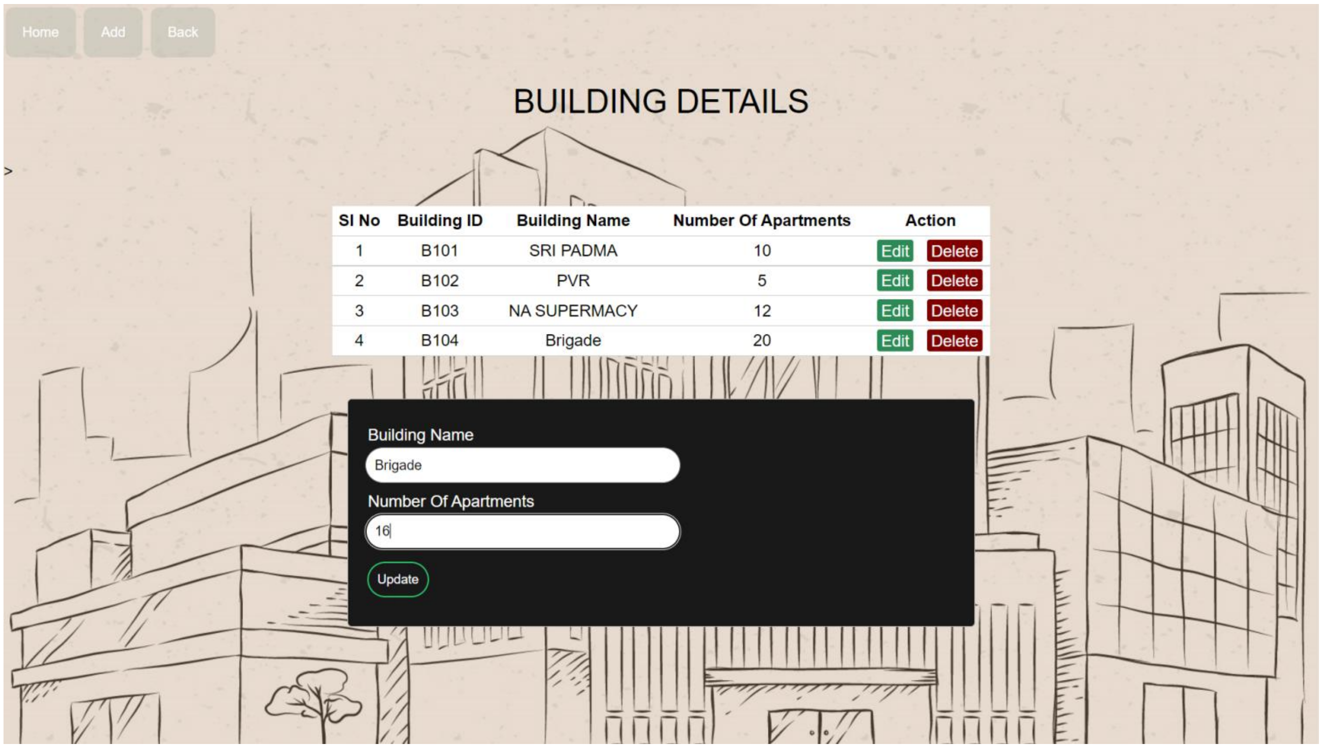
Fig: Reading of Inserted Details



BUILDING DETAILS

Sl No	Building ID	Building Name	Number Of Apartments	Action
1	B101	SRI PADMA	10	Edit Delete
2	B102	PVR	5	Edit Delete
3	B103	NA SUPERMACY	12	Edit Delete
4	B104	Brigade	20	Edit Delete

Fig: Reading of Building Details



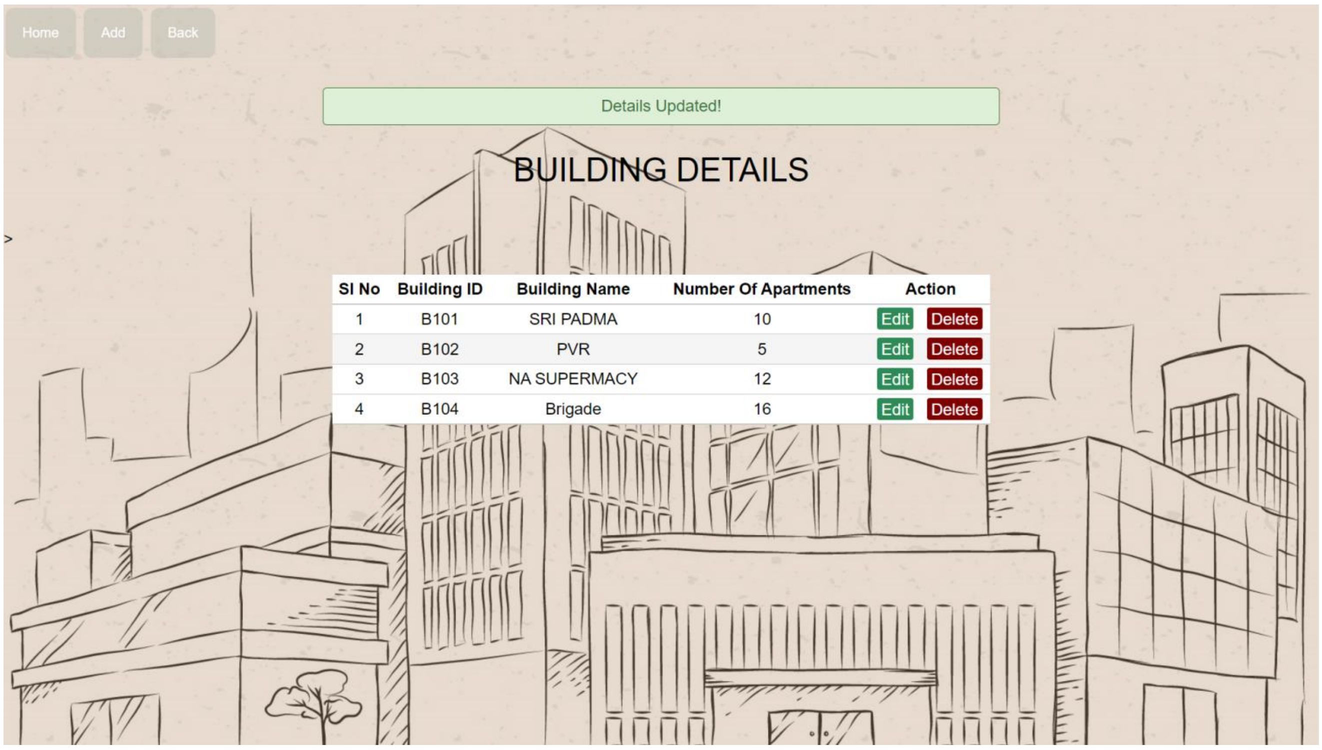
BUILDING DETAILS

Sl No	Building ID	Building Name	Number Of Apartments	Action
1	B101	SRI PADMA	10	Edit Delete
2	B102	PVR	5	Edit Delete
3	B103	NA SUPERMACY	12	Edit Delete
4	B104	Brigade	20	Edit Delete

Building Name: Brigade

Number Of Apartments: 16

Fig: Updation of Building Details

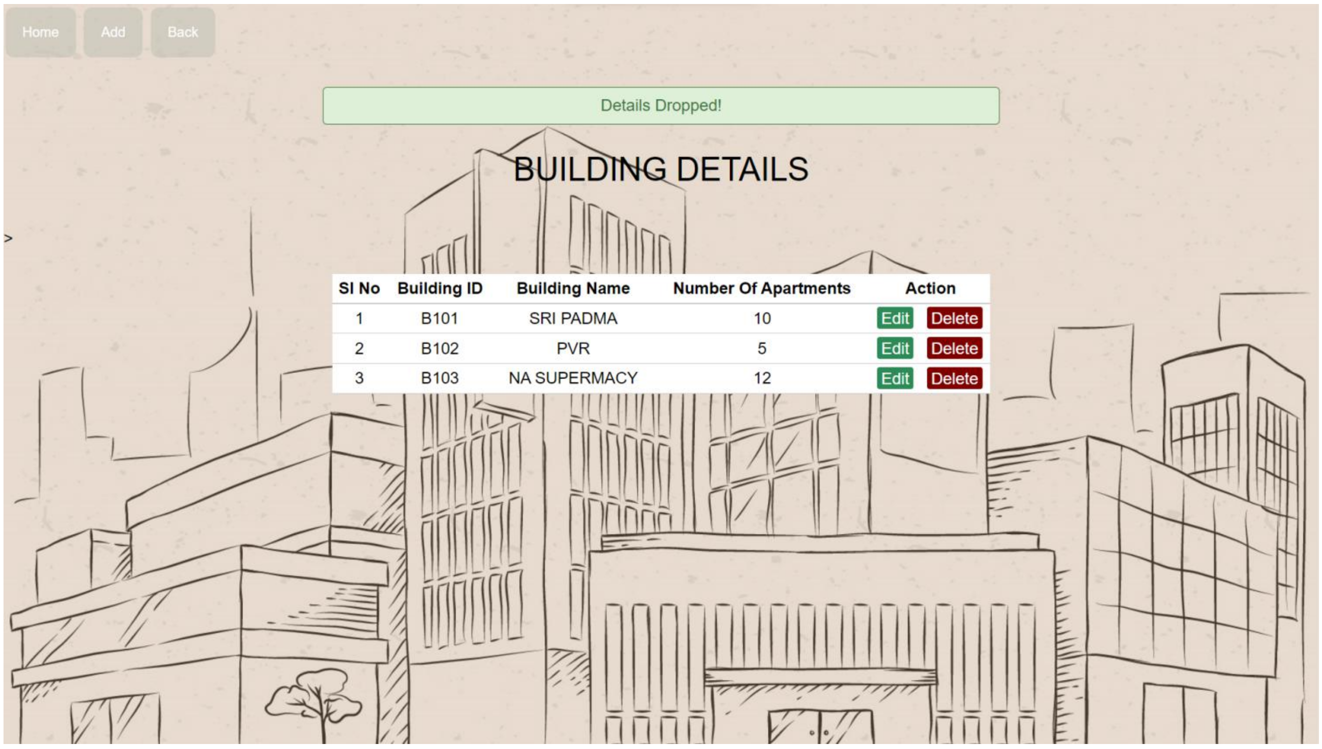


Details Updated!

BUILDING DETAILS

Sl No	Building ID	Building Name	Number Of Apartments	Action
1	B101	SRI PADMA	10	Edit Delete
2	B102	PVR	5	Edit Delete
3	B103	NA SUPERMACY	12	Edit Delete
4	B104	Brigade	16	Edit Delete

Fig: Building Details Updated Successfully

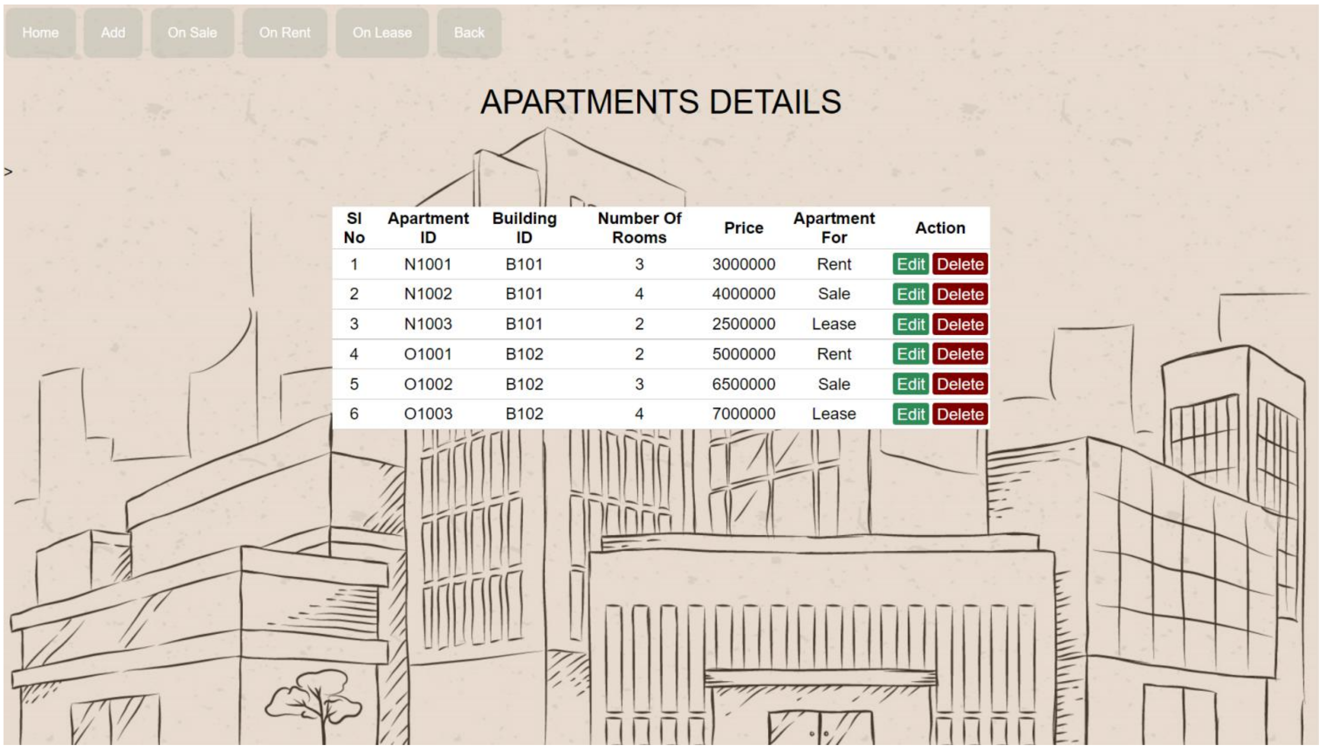


Details Dropped!

BUILDING DETAILS

Sl No	Building ID	Building Name	Number Of Apartments	Action
1	B101	SRI PADMA	10	Edit Delete
2	B102	PVR	5	Edit Delete
3	B103	NA SUPERMACY	12	Edit Delete

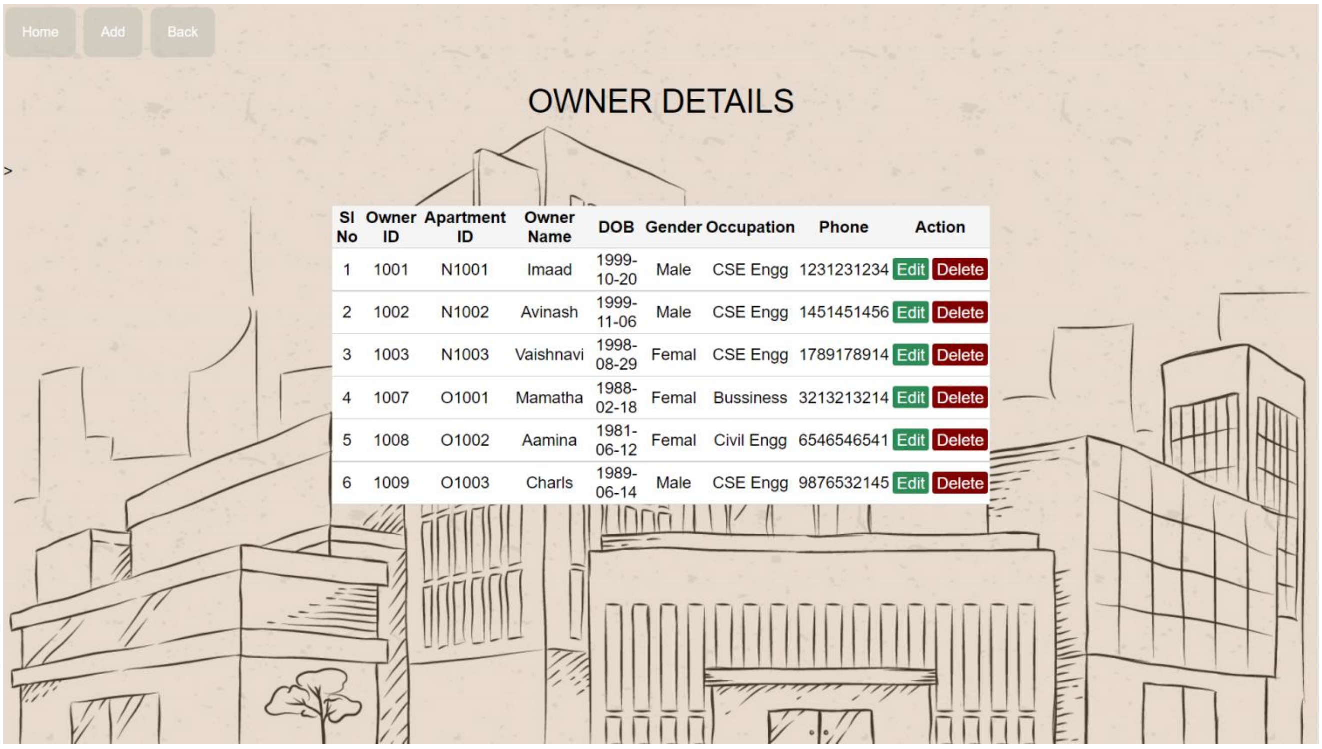
Fig: Deletion of Brigade Tuple Successfully



APARTMENTS DETAILS

Sl No	Apartment ID	Building ID	Number Of Rooms	Price	Apartment For	Action
1	N1001	B101	3	3000000	Rent	Edit Delete
2	N1002	B101	4	4000000	Sale	Edit Delete
3	N1003	B101	2	2500000	Lease	Edit Delete
4	O1001	B102	2	5000000	Rent	Edit Delete
5	O1002	B102	3	6500000	Sale	Edit Delete
6	O1003	B102	4	7000000	Lease	Edit Delete

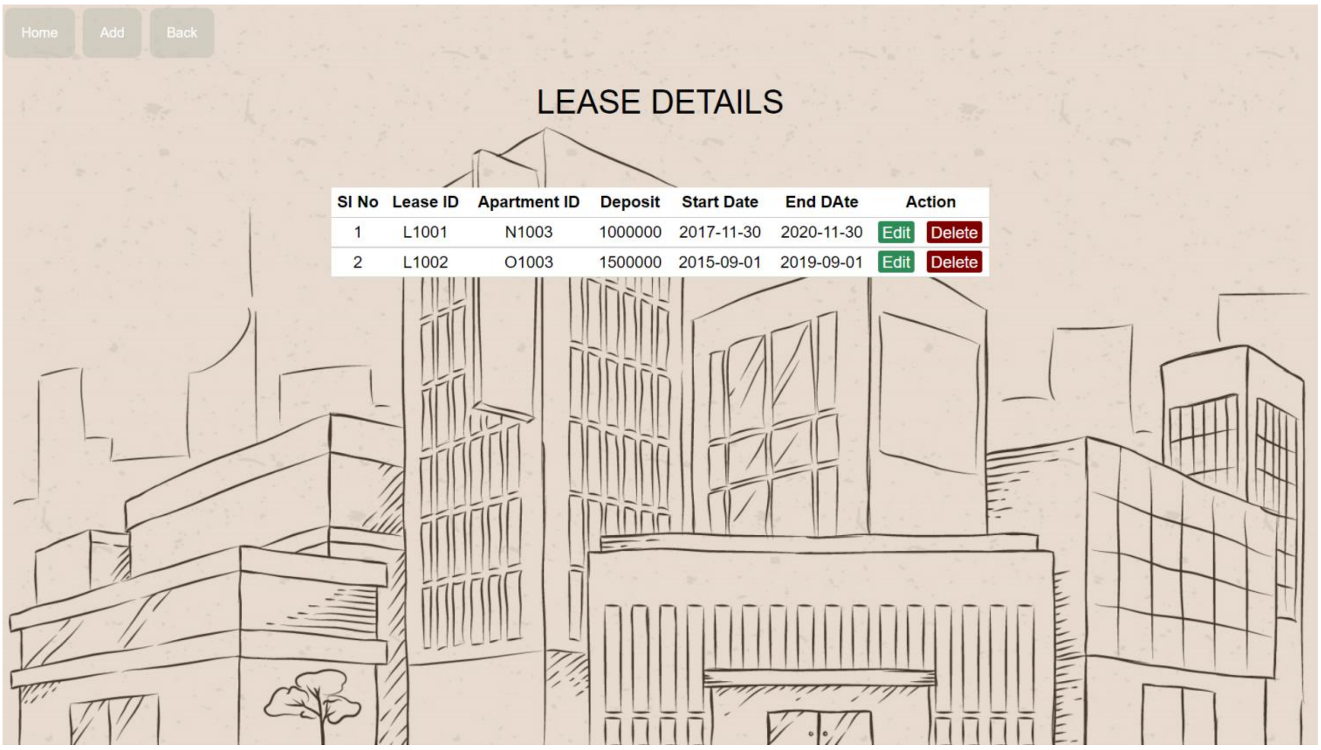
Fig: Reading of Apartment Details



OWNER DETAILS

Sl No	Owner ID	Apartment ID	Owner Name	DOB	Gender	Occupation	Phone	Action
1	1001	N1001	Imaad	1999-10-20	Male	CSE Engg	1231231234	Edit Delete
2	1002	N1002	Avinash	1999-11-06	Male	CSE Engg	1451451456	Edit Delete
3	1003	N1003	Vaishnavi	1998-08-29	Femal	CSE Engg	1789178914	Edit Delete
4	1007	O1001	Mamatha	1988-02-18	Femal	Bussiness	3213213214	Edit Delete
5	1008	O1002	Aamina	1981-06-12	Femal	Civil Engg	6546546541	Edit Delete
6	1009	O1003	Charls	1989-06-14	Male	CSE Engg	9876532145	Edit Delete

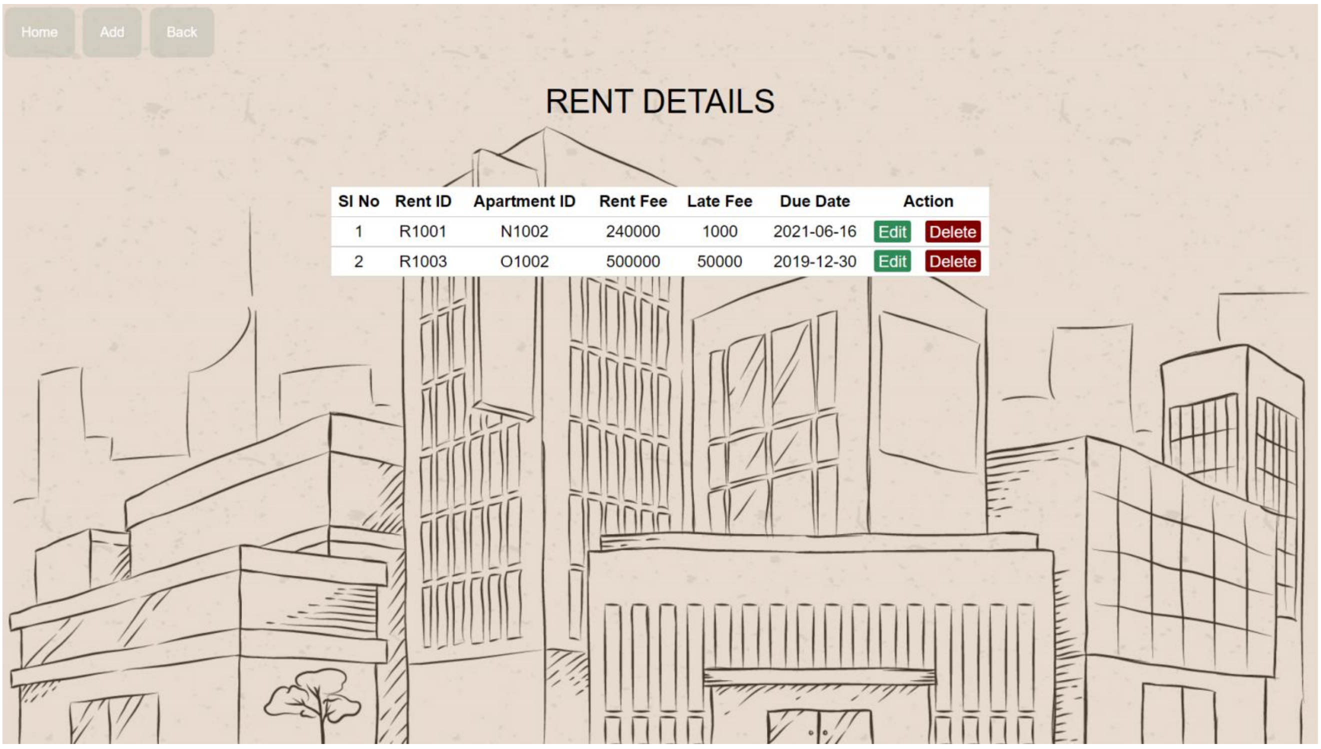
Fig: Reading of Owner Details



LEASE DETAILS

Sl No	Lease ID	Apartment ID	Deposit	Start Date	End DAte	Action
1	L1001	N1003	1000000	2017-11-30	2020-11-30	Edit Delete
2	L1002	O1003	1500000	2015-09-01	2019-09-01	Edit Delete

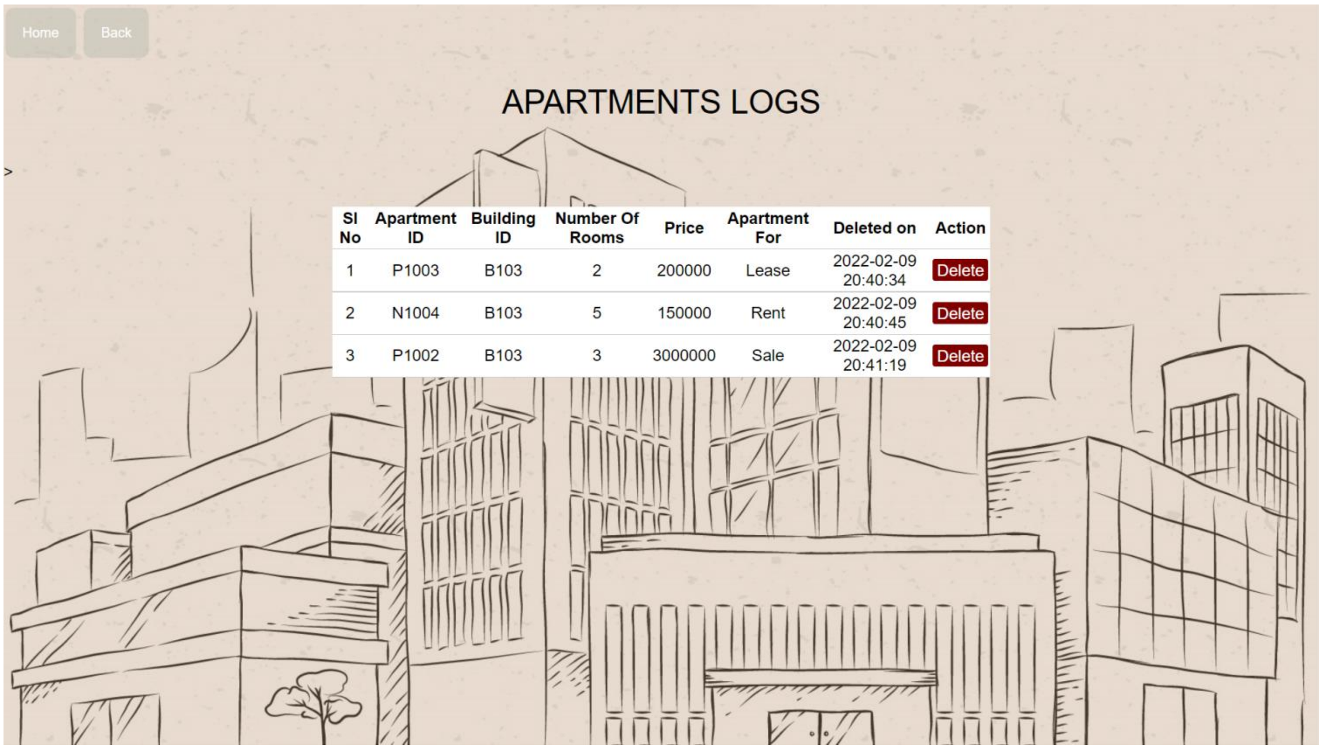
Fig: Reading of Lease Details



RENT DETAILS

Sl No	Rent ID	Apartment ID	Rent Fee	Late Fee	Due Date	Action
1	R1001	N1002	240000	1000	2021-06-16	Edit Delete
2	R1003	O1002	500000	50000	2019-12-30	Edit Delete

Fig: Reading of Rent Details



APARTMENTS LOGS

Sl No	Apartment ID	Building ID	Number Of Rooms	Price	Apartment For	Deleted on	Action
1	P1003	B103	2	200000	Lease	2022-02-09 20:40:34	Delete
2	N1004	B103	5	150000	Rent	2022-02-09 20:40:45	Delete
3	P1002	B103	3	3000000	Sale	2022-02-09 20:41:19	Delete

Fig: Reading of Apartment Logs

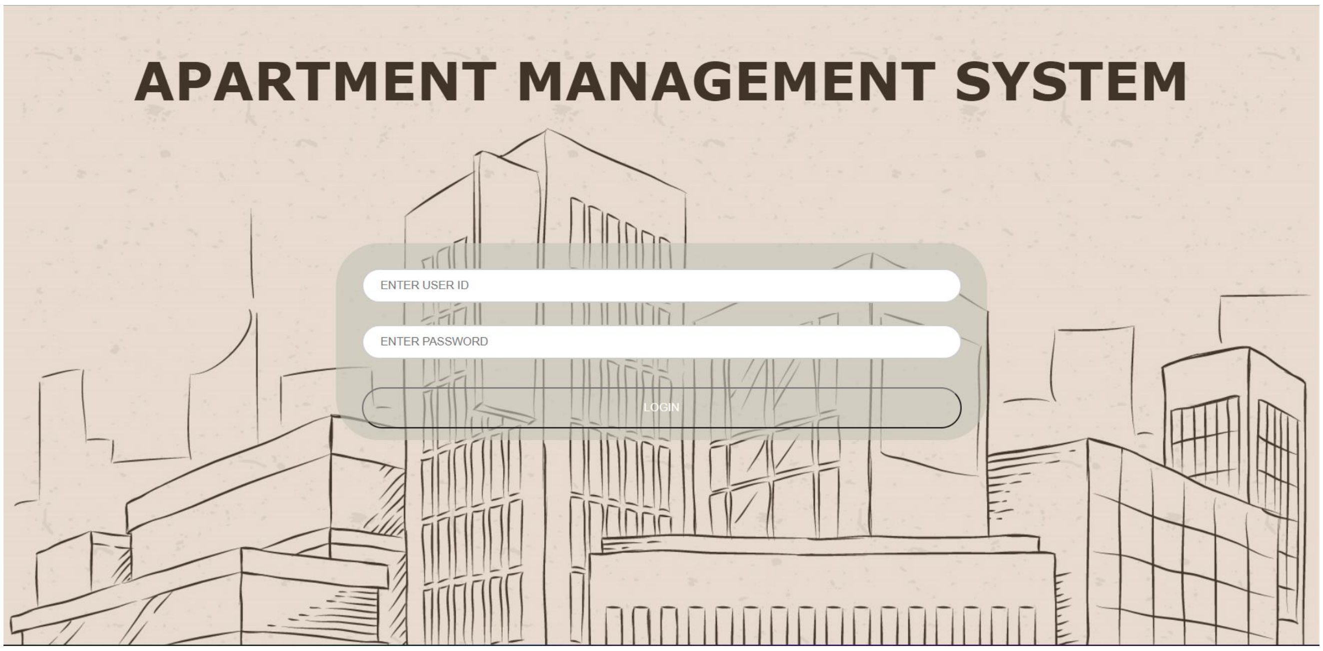

APARTMENT MANAGEMENT SYSTEM

Fig: After Successfully Logging Out

6. Conclusion And Future Enhancements

The development of this apartment management system is a great improvement over the manual system, which uses lots of manual and paper work. The computerization of the system speeds up the process.
The Apartment Management System is fast, efficient and reliable, avoids data redundancy and inconsistency. It contains all the functional features described in the objective of the project.

In the future, the various improvements that can be made are

- Since this is an admin only web application, user side of the application can be developed in the future.
- The user interface can be improved by making it more user-friendly and attractive.
- The web application currently runs in local server provided by the browser, in the future it can hosted and deployed so that a clickable link can be used to access the application.
- Security was a property taken into consideration hence, the existence of both user authentication and the presence of certain functions to prevent HTML and SQL injection but it is still not enough. More layers of security must be including to make the data tighter and more strictly controlled calls.
- The GUI can be bettered as well, to enable the CSS script to be more dynamic and include responsive design
- Rent and Lease payments of the apartment can also be added in the webpage for the future betterment of the project.

7. REFERENCES

7.1 Book References

- Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
- Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill

7.2 Web References

- https://stackoverflow.com/
- https://www.youtube.com/
- https://www.php.net/
- https://www.php.net/manual/en/book.pdo.php
- https://dev.mysql.com/doc/refman/8.0/en/
- https://www.w3schools.com/html/default.asp
- https://www.w3schools.com/css/default.asp
- https://www.w3schools.com/js/default.asp
- https://www.w3schools.com/php/default.asp
- https://en.wikipedia.org/wiki/Main_Page

7.3 Videos References

- https://www.youtube.com/watch?v=OJEQaVT45XA
- https://www.youtube.com/watch?v=qz0aGYrrlhU
- https://www.youtube.com/watch?v=W6NZfCO5SIk
- https://www.youtube.com/watch?v=I_vyo