VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAUM-590014

A DBMS Mini-Project Report

On

"RENTAL MANAGEMENT SYSTEM"

A Mini project report submitted in partial fulfillment of the requirements for the 5th semester of
Bachelor of Engineering in Computer Science and Engineering
of Visvesvaraya Technological University, Belgaum

Submitted by:

NIRMAL KUMAR(1ST18CS087)

MD HASAN(1ST18C068)

Under the Guidance of:

Prof. RAJSHEKHAR S A
Assoc Professor
Dept. of CSE

Department of Computer Science and Engineering

Sambhram Institute of Technology

M.S.Palya, Bangalore-560 097

2020-21

SAMBRAM INSTITUTE OF TECHNOLOGY
M.S. PALYA, BANGALORE-97.
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the mini project work entitled **“Rental Management System”** has been successfully carried out by Nirmal Kumar bearing USN 1ST18CS087 and Md Hasan bearing USN 1ST18CS068 bonafide students of **Sambram Institute of Technology** in partial fulfillment of the requirements for the **5th semester** of **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University**, Belgaum, during academic year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the laboratory requirements of 5th semester BE, CSE.

Guide:	**HOD:**
Prof. Rajshekhar SA	**Dr.T. John Peter**
Asst Prof.	**Prof. and Head**
Dept. of CSE	**Dept of CSE**

<u>External Viva:</u>

Name of the Examiners	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project work. We would like to take this opportunity to thank them all.

We would like to thank **Dr.H.C Chandrakanth**, Principal, SAIT, Bangalore, for his moral support towards completing my project.

We would like to thank **Dr.T. John peter**, Prof & Head, Department of Computer Science & Engineering, SaIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express my sincere gratitude to my guide **Prof. Rajshekhar S.A** Asst Prof, Department of CSE, SaIT, Bangalore, for their able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank all the teaching and non-teaching staff of Department of Computer Science & Engineering SaIT, Bangalore for their constant support and encouragement.

Date: **Students' names**

Nirmal Kumar

Place: Bangalore Md Hasan

ABSTRACT

The main objective of the Rental Management System is to provide an easy interface to maintain all the information of all tenants in a place. The project is totally built at administrative end, so that only owner of the house or apartment can view or manage all valuable information from their tenants. The system has the following features:

- Secure login to system via a login form.
- User friendly interface to add data to database and view reports at one place.
- Owner can track particular’s rent, paying dates and transaction Id.
- Account section will show overall payment paid to owner.
- Owner can generate payment receipt for tenants.

In addition, owner can analyze old data to provide better service in future. New features like face detection, WhatsApp integration can be done to this project in future. Tenants logbook connected with face detection api can be automatically filled with out-time and in-time. WhatsApp integration can be done to send payment receipt directly to tenants.

CONTENTS

CHAPTER 1

INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

1 INTRODUCTION

Databases and database technology have 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, genetics, law, education, and library science. The word database is so commonly used that we must begin by defining what a 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 a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database. The preceding definition of 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 implicit properties:

- A database represents some aspect of the real world, sometimes called the miniworld or the universe of discourse (UoD). Changes to the miniworld 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.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is general-purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications. Defining a database involves specifying the data types, structures and constraints of the data to be stored in the database.
The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called meta-data. Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS. Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and

generating reports from the data. Sharing a database allows multiple users and programs to access the database simultaneously.

1.2 HISTORY OF DBMS

A Database Management System allows a person to organize, store, and retrieve data from a computer. It is a way of communicating with a computer's "stored memory." In the very early years of computers, "punch cards" were used for input, output, and data storage. Punch cards offered a fast way to enter data, and to retrieve it. Herman Hollerith is given credit for adapting the punch cards used for weaving looms to act as the memory for a mechanical tabulating machine, in 1890. Much later, databases came along.

Databases (or DBs) have played a very important part in the recent evolution of computers. The first computer programs were developed in the early 1950s, and focused almost completely on coding languages and algorithms. At the time, computers were basically giant calculators and data (names, phone numbers) was considered the leftovers of processing information. Computers were just starting to become commercially available, and when business people started using them for real-world purposes, this leftover data suddenly became important.

Enter the Database Management System (DBMS). A database, as a collection of information, can be organized so a Database Management System can access and pull specific information. In 1960, Charles W. Bachman designed the Integrated Database System, the "first" DBMS. IBM, not wanting to be left out, created a database system of their own, known as IMS. Both database systems are described as the forerunners of navigational database.

By the mid-1960s, as computers developed speed and flexibility, and started becoming popular, many kinds of general use database systems became available. As a result, customers demanded a standard be developed, in turn leading to Bachman forming the Database Task Group. This group took responsibility for the design and standardization of a language called Common Business Oriented Language (COBOL). The Database Task Group presented this standard in 1971, which also came to be known as the "CODASYL approach."

The CODASYL approach was a very complicated system and required substantial training. It depended on a "manual" navigation technique using a linked data set, which formed a large network. Searching for records could be accomplished by one of three techniques:

- Using the primary key (also known as the CALC key)

- Moving relationships (also called sets) to one record from another
- Scanning all records in sequential order

Eventually, the CODASYL approach lost its popularity as simpler, easier-to-work-with systems came on the market.

Edgar Codd worked for IBM in the development of hard disk systems, and he was not happy with the lack of a search engine in the CODASYL approach, and the IMS model. He wrote a series of papers, in 1970, outlining novel ways to construct databases. His ideas eventually evolved into a paper titled, A Relational Model of Data for Large Shared Data Banks, which described new method for storing data and processing large databases. Records would not be stored in a free-form list of linked records, as in CODASYL navigational model, but instead used a “table with fixed-length records.” IBM had invested heavily in the IMS model, and wasn’t terribly interested in Codd’s ideas. Fortunately, some people who didn’t work for IBM “were” interested. In 1973, Michael Stonebraker and Eugene Wong (both then at UC Berkeley) made the decision to research relational database systems. The project was called INGRES (*Interactive Graphics and Retrieval System*), and successfully demonstrated a relational model could be efficient and practical.

INGRES worked with a query language known as QUEL, in turn, pressuring IBM to develop SQL in 1974, which was more advanced (SQL became ANSI and OSI standards in 1986 1nd 1987). SQL quickly replaced QUEL as the more functional query language.
RDBM Systems were an efficient way to store and process structured data. Then, processing speeds got faster, and “unstructured” data (art, photographs, music, etc.) became much more common place. Unstructured data is both non-relational and schema-less, and Relational Database Management Systems simply were not designed to handle this kind of data.

1.3 CHARACTERISTIC OF DATABASE APPROACH

A number of characteristics distinguish the database approach from the much older approach of programming with files. In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application. For example, one user, the grade reporting office, may keep files on students and their grades. Programs to print a student’s transcript and to enter new grades are implemented as part of the application. A second user, the accounting office, may keep track of students’ fees and their payments. Although both users are interested in data about students, each user maintains separate files— and programs to manipulate these files—because each requires some data not avail able from the other user’s files. This redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common up-to-date data. In the database approach, a single repository maintains data that is defined once and then accessed by various users. In file systems, each application is free to name data elements independently. In contrast, in a database, the names or labels of data are defined once, and used repeatedly by queries, transactions, and applications. The main characteristics of the database approach versus the file-processing approach are the following:

- Self-describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing.

1.3.1 Self-Describing Nature of a Database System

A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints. This definition is stored in the DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The information stored in the catalog is called meta-data, and it describes the structure of the primary database. The catalog is used by the DBMS software and also by database users who need information about the database structure. A general-purpose DBMS software package is not written for a specific database application. Therefore, it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access. The DBMS software must work equally well with any number of database applications—for example, a university database, a banking database, or a company database—as long as the database definition is stored in the catalog. In traditional file processing, data definition is typically part of the application programs themselves. Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs. For example, an application program written in C++ may have structure or class declarations, and a COBOL program has data division statements to define its files. Whereas file-processing software can access only specific databases, DBMS software can access diverse databases by extracting the database definitions from the catalog and using these definitions.

1.3.2 Insulation between Programs and Data, and Data Abstraction

In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file. By contrast, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs. We call this property program-data independence. In some types of database systems, such as object-oriented and object-relational systems users can define operations on data as part of the database definitions. An operation (also called a function or method) is specified in two parts. The interface (or signature) of an operation includes the operation name and the data types of its arguments (or parameters). The implementation (or method) of the operation is specified separately and can be changed without affecting the interface. User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This may be termed program-operation independence. The characteristic that allows program-data independence and program-operation independence is called data abstraction. A DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented. Informally, a data model is a type of data abstraction that is used to provide this conceptual representation. The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for most users to understand than computer storage concepts. Hence, the data model hides storage and implementation details that are not of interest to most database users.

1.3.3 Support of Multiple Views of the Data

A database typically has many users, each of whom may require a different perspective or view of the database. A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored. Some users may not need to be aware of whether the data they refer to is stored or derived. A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

1.3.4 Sharing of Data and Multiuser Transaction Processing

A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct. For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger. These types of applications are generally called online transaction processing (OLTP) applications. A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently. The concept of a transaction has become central to many database applications. A transaction is an executing program or process that includes one or more database accesses, such as reading or updating of database records. Each transaction is supposed to execute a logically correct database access if executed in its entirety without interference from other transactions. The DBMS must enforce several transaction properties. The isolation property ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing concurrently. The atomicity property ensures that either all the database operations in a transaction are executed or none are.

1.4 APPLICATIONS OF DBMS

Applications where we use Database Management Systems are:

- **Telecom**: There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Industry**: Where it is a manufacturing unit, warehouse or distribution center, each one needs a database to keep the records of ins and outs. For example, distribution center should keep a track of the product units that supplied into the center as well as the products that got delivered out from the distribution center on each day; this is where DBMS comes into picture.
- **Banking System**: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
- **Education sector**: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
- **Online shopping**: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

CHAPTER 2

SCHEMA AND ER DIAGRAM

2.1 ER DIAGRAM

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored.

The cardinality or fundamental principle of one data aspect with respect to another is a critical feature. The relationship of one to the other must be precise and exact between each other in order to explain how each aspect links together. In simple words Cardinality is a way to define the relationship between two entities.

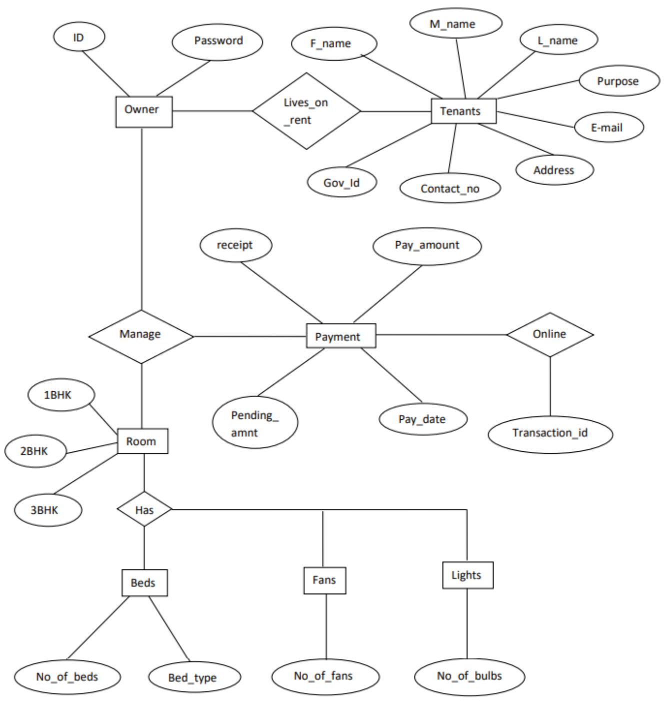


Fig 2.1 ER Diagram

2.2 SCHEMA DIAGRAM

In any data model it is important to distinguish between the description of the database and the database itself. The description of a database is called the database schema, which is specified during database design and is not expected to change frequently.

A displayed schema is called a schema diagram. A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints.

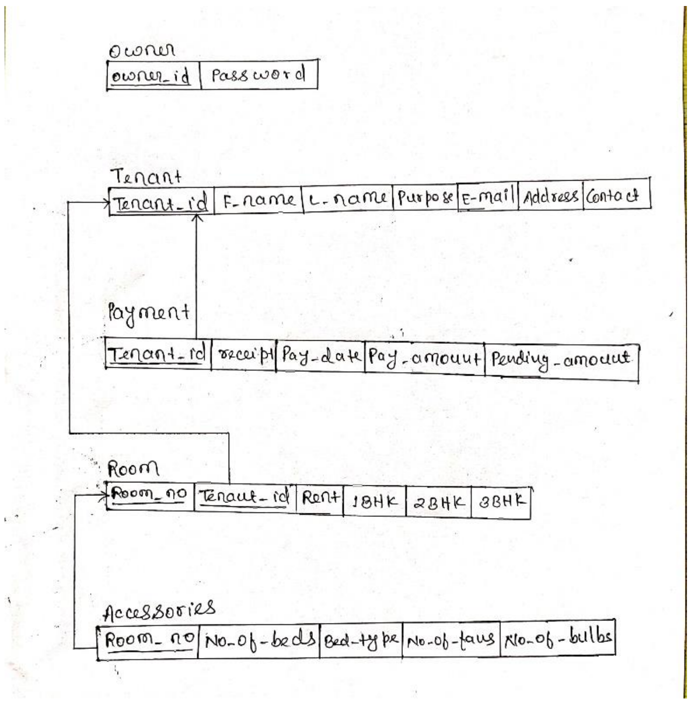


Fig 2.2 Schema Diagram

CHAPTER 3

RESOURCE REQUIREMENTS

3.1 SOFTWARE REQUIREMENT: -

- Front End tools: HTML, JavaScript, CSS, PHP, Bootstrap4
- Back End tools: phpMyAdmin
- Browser that supports HTML and JavaScript
- IIS or apache server
- MySQL database
- Xampp server

3.2 HARDWARE REQUIREMENT: -

- CPU: Pentium processor and above
- RAM: 2 GB
- HDD: 40 GB

CHAPTER 4

DESCRIPTION OF TOOLS AND TECHNOLOGIES

4.1 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them 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. It 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 <img/> and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

4.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

4.3 JAVASCRIPT

JavaScript (sometimes abbreviated JS) is a prototype-based scripting language that is dynamic, weakly typed, general purpose programming language and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

JavaScript was formalized in the ECMA Script language standard and is primarily used in the form of client-side JavaScript, implemented as part of a Web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment.

JavaScript's use in applications outside Web pages for example in PDF documents, site-specific browsers, and desktop widgets is also significant. In this application, JavaScript is used for validation purpose like text box validation, email validation, phone number validation. JavaScript is the good tool for validating the web-applications.

4.4 PHP LANGUAGE

PHP is a server-side scripting language designed primarily for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Development Team. PHP originally stood for Personal Home Page, but it now stands for the recursive acronym PHP: Hypertext Preprocessor.

PHP code may be embedded into HTML or HTML5 markup, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server software combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

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, leaving the canonical PHP interpreter as a de facto standard. Since 2014 work has gone on to create a formal PHP specification.

HP development began in 1995 when Rasmus Lerdorf wrote several Common Gateway Interface (CGI) programs in C, which he used to maintain his personal homepage. He extended them to work with web forms and to communicate with databases, and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.

PHP/FI could help to build simple, dynamic web applications. To accelerate bug reporting and to improve the code, Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP Tools) version 1.0" on the Usenet discussion group on June 8, 1995 This release already had the basic functionality that PHP has as of 2013. This included Perl-like variables, form handling, and the ability to embed HTML. The syntax resembled that of Perl but was simpler, more limited and less consistent.

4.5 IIS OR APACHE SERVER

Apache HTTP Server, colloquially called Apache, is free and open-source cross-platform web server software, released under the terms of Apache License 2.0.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, TCL, and PHP. Popular authentication modules include mod_access, mod_auth, mod_digest, and mod_auth_digest, the successor to mod_digest. A sample of other features include Secure Sockets Layer and Transport Layer Security support (mod_ssl), a proxy module (mod_proxy), a URL rewriting module (mod_rewrite), custom log files (mod_log_config), and filtering support (mod_include and mod_ext_filter).

Popular compression methods on Apache include the external extension module, mod_gzip, implemented to help with reduction of the size (weight) of Web pages served over HTTP. Mod Security is an open-source intrusion detection and prevention engine for Web applications. Apache logs can be analyzed through a Web browser using free scripts, such as AWStats/W3Perl or Visitors.

Virtual hosting allows one Apache installation to serve many different Web sites. For example, one machine with one Apache installation could simultaneously serve www.example.com, www.example.org, test47.test-server.example.edu, etc.

Apache features configurable error messages, DBMS-based authentication databases, and content negotiation. It is also supported by several graphical user interfaces (GUIs).

It supports password authentication and digital certificate authentication. Because the source code is freely available, anyone can adapt the server for specific needs, and there is a large public library of Apache add-ons.

4.6 MySQL DATABASE

MySQL is a Relational Database Management System (RDBMS). MySQL server can manage many databases at the same time. In fact, many people might have different databases managed by a single MySQL server. Each database consists of a structure to hold the data and the data itself. A data-base can exist without data, only a structure, be totally empty, twiddling its thumbs and waiting for data to be stored in it.

Data in a database is stored in one or more tables. You must create the data-base and the tables before you can add any data to the database. First you create the empty database. Then you add empty tables to the database. Database tables are organized like other tables that you're used in rows and columns. Each row represents an entity in the database, such as a customer, a book, or a project. Each column contains an item of information about the entity, such as a customer name, a book name, or a project start date. The place where a particular row and column intersect, the individual cell of the table, is called a field. Tables in databases can be related. Often a row in one table is related to several rows in another table. For instance, you might have a database containing data about books you own. You would have a book table and an author table. One row in the author table might contain information about the author of several books in the book table. When tables are related, you include a column in one table to hold data that matches data in the column of another table.

4.7 WHAT IS MySQL

MySQL, the most popular Open-Source SQL database management system, is developed, distributed, and supported by MySQL AB. MySQL AB is a commercial company, founded by the MySQL developers. It is a second-generation Open-Source company that unites Open-Source values and methodology with a successful business model.

- MySQL is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server.

Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- MySQL is a relational database management system.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. This adds speed and flexibility. The SQL part of “MySQL” stands for “Structured Query Language.” SQL is the most common standardized language used to access databases and is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. “SQL-92” refers to the standard released in 1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

- MySQL software is Open Source.

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations. The MySQL Database Server is very fast, reliable, and easy to use.

- MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

- MySQL Server works in client/server or embedded systems.

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

4.8 XAMPP SERVER

Xampp server installs a complete and ready-to-use development environment. Xampp server allows you to fit your needs and allows you to setup a local server with the same characteristics as your production.

In case of setting up the server and PHP on your own, you have two choices for the method of connecting PHP to the server. For many servers PHP has a direct module interface (also called SAPI).

These servers include Apache, Microsoft Internet Information Server, Netscape and iPlanet servers. Many other servers have support for ISAPI, the Microsoft module interface (OmniHTTPd for example). If PHP has no module support for your web server, you can always use it as a CGI or Fast CGI processor. This means you set up your server to use the CGI executable of PHP to process all PHP file requests on the server.

CHAPTER 5

PHP AND DATABASE CONNECTION

5.1 CREATING DATABASE CONNECTION

- PHP provides built-in database connectivity for a wide range of databases – MySQL, PostgreSQL, Oracle, Berkeley DB, Informix, Lotus Notes, and more
- Use either mysql_connect or mysql_pconnect to create database connection
- mysql_connect: connection is closed at end of script (end of page)
- mysql_pconnect: creates persistent connection -connection remains even after end of the page
- Connect to the MySQL server

○ $connection = mysql_connect("localhost", $username, $password);

- Access the database

○ mysql_select_db("databasename", $connection);

- Perform SQL operations

○ Example: $result = mysql_query ($query, $connection)

- Disconnect from the server

○ mysql_close($connection)

5.2 ARCHITECTURE USED (4-TIER ARCHITECTURE)

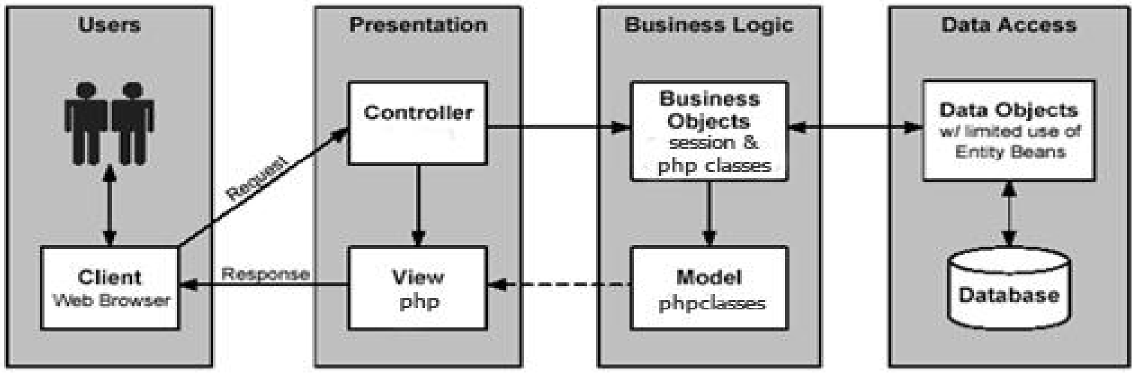


Fig: Architecture used

Four Tier architecture is a client–server architecture in which presentation, application processing, and data management functions are physically separated. Four-tier application architecture provides a model by which developers can create flexible and reusable applications. By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application.

5.2.1 PRESENTATION LAYER

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. In simple terms, it is a layer which users can access directly (such as a web page, or an operating system's GUI).

5.2.2 BUSINESS LAYER

Business layer or domain logic is the part of the program that encodes the real-world business rules that determine how data can be created, stored, and changed. It is contrasted with the remainder of the software that might be concerned with lower-level details of managing a database or displaying the user interface, system infrastructure, or generally connecting various parts of the program.

5.2.3 DATA ACCESS LAYER

A Data Access Layer (DAL) in computer software, is a layer of a computer program which provides simplified access to data stored in persistent storage.

For example, the DAL might return a reference to an object (in terms of object-oriented programming) complete with its attributes instead of a row of fields from a database table. This allows the client (or user) modules to be created with a higher level of abstraction. This kind of model could be implemented by creating a class of data access methods that directly reference a corresponding set of database stored procedures. Another implementation could potentially retrieve or write records to or from a file system. The DAL hides this complexity of the underlying data store from the external world.

5.2.4 CONTROL LAYER

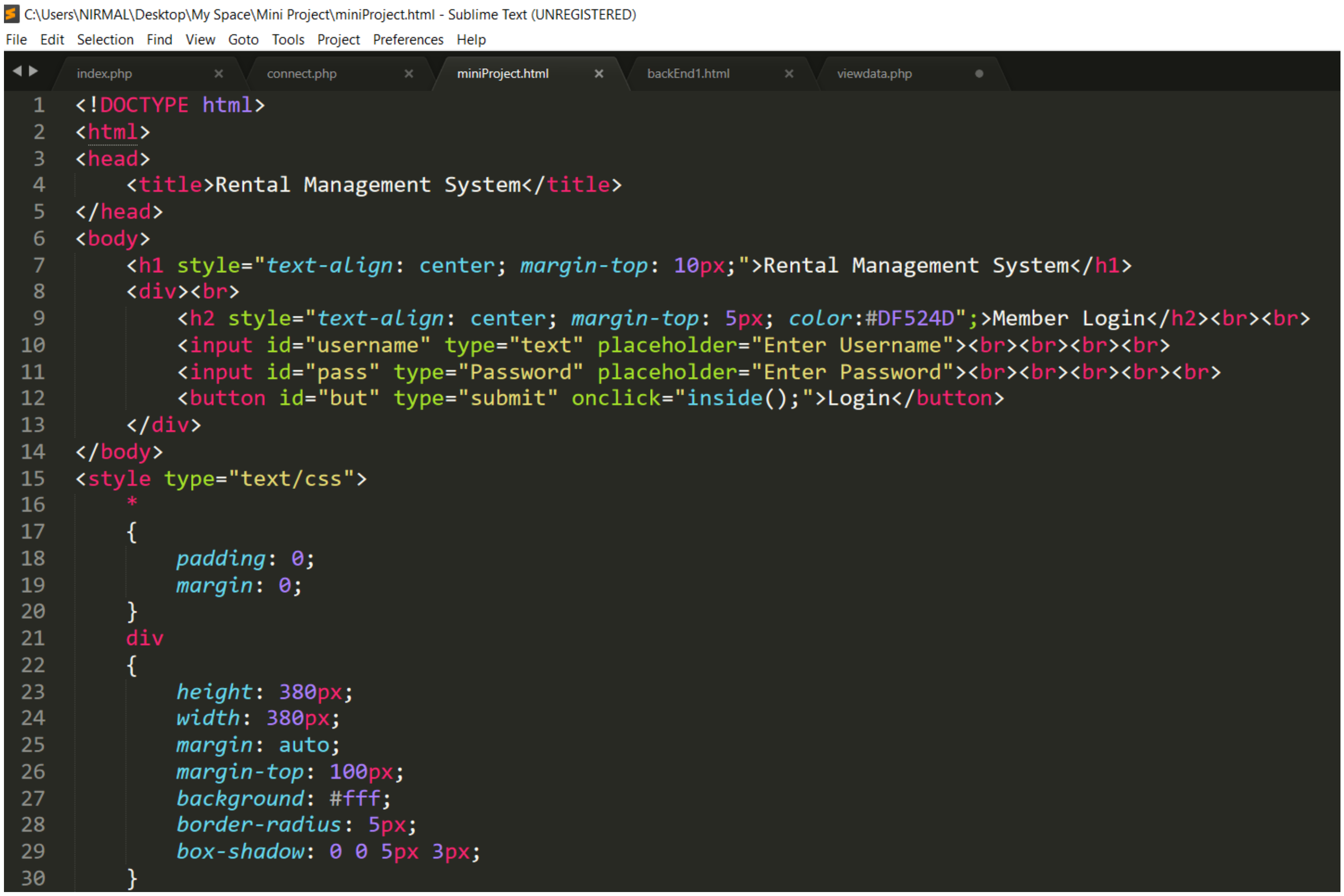
The control layer is responsible for communication between business and presentation layer. It connects the logic and data with each other and gives a better connectivity and separation between layers.

CHAPTER 6

IMPLEMENTATION

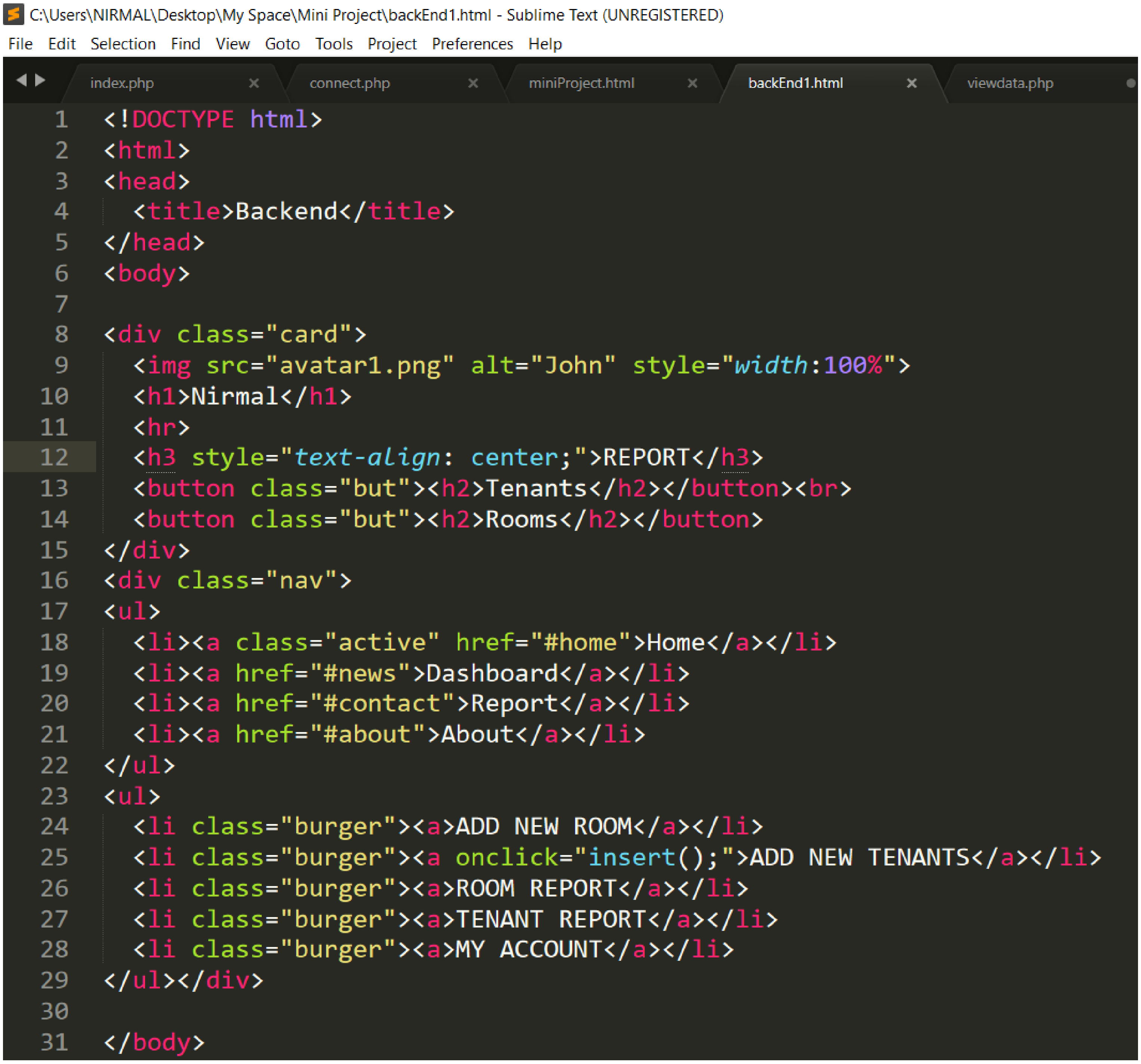
SOURCE CODE

LOGIN CODE:


C:\Users\NIRMAL\Desktop\My Space\Mini Project\miniProject.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
index.php
connect.php
miniProject.html
backEnd1.html
viewdata.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Rental Management System</title>
</head>
<body>
    <h1 style="text-align: center; margin-top: 10px;">Rental Management System</h1>
    <div><br>
        <h2 style="text-align: center; margin-top: 5px; color:#DF524D";>Member Login</h2><br><br>
        <input id="username" type="text" placeholder="Enter Username"><br><br><br><br>
        <input id="pass" type="Password" placeholder="Enter Password"><br><br><br><br><br>
        <button id="but" type="submit" onclick="inside();">Login</button>
    </div>
</body>
<style type="text/css">
    *
    {
        padding: 0;
        margin: 0;
    }
    div
    {
        height: 380px;
        width: 380px;
        margin: auto;
        margin-top: 100px;
        background: #fff;
        border-radius: 5px;
        box-shadow: 0 0 5px 3px;
    }
```

DASHBOARD CODE:

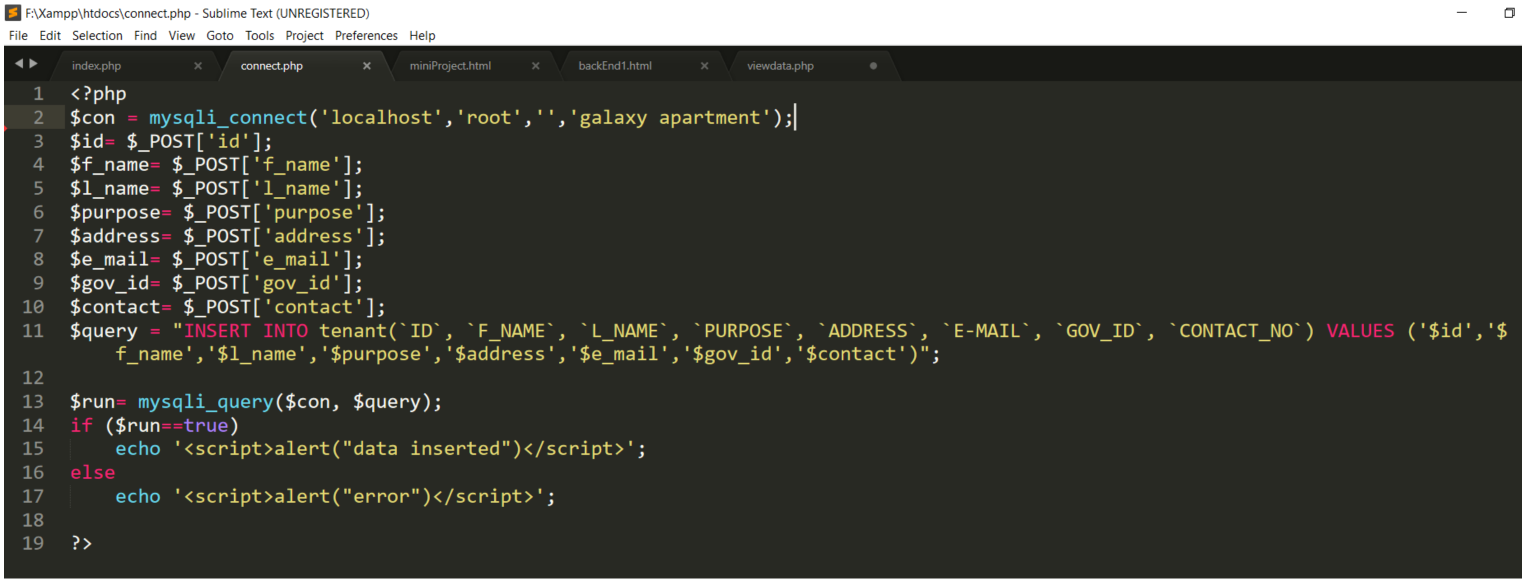

C:\Users\NIRMAL\Desktop\My Space\Mini Project\backEnd1.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
index.php
connect.php
miniProject.html
backEnd1.html
viewdata.php

```html
<!DOCTYPE html>
<html>
<head>
  <title>Backend</title>
</head>
<body>

<div class="card">
  <img src="avatar1.png" alt="John" style="width:100%">
  <h1>Nirmal</h1>
  <hr>
  <h3 style="text-align: center;">REPORT</h3>
  <button class="but"><h2>Tenants</h2></button><br>
  <button class="but"><h2>Rooms</h2></button>
</div>
<div class="nav">
<ul>
  <li><a class="active" href="#home">Home</a></li>
  <li><a href="#news">Dashboard</a></li>
  <li><a href="#contact">Report</a></li>
  <li><a href="#about">About</a></li>
</ul>
<ul>
  <li class="burger"><a>ADD NEW ROOM</a></li>
  <li class="burger"><a onclick="insert();">ADD NEW TENANTS</a></li>
  <li class="burger"><a>ROOM REPORT</a></li>
  <li class="burger"><a>TENANT REPORT</a></li>
  <li class="burger"><a>MY ACCOUNT</a></li>
</ul></div>

</body>
```

PHP CONNECTION CODE:

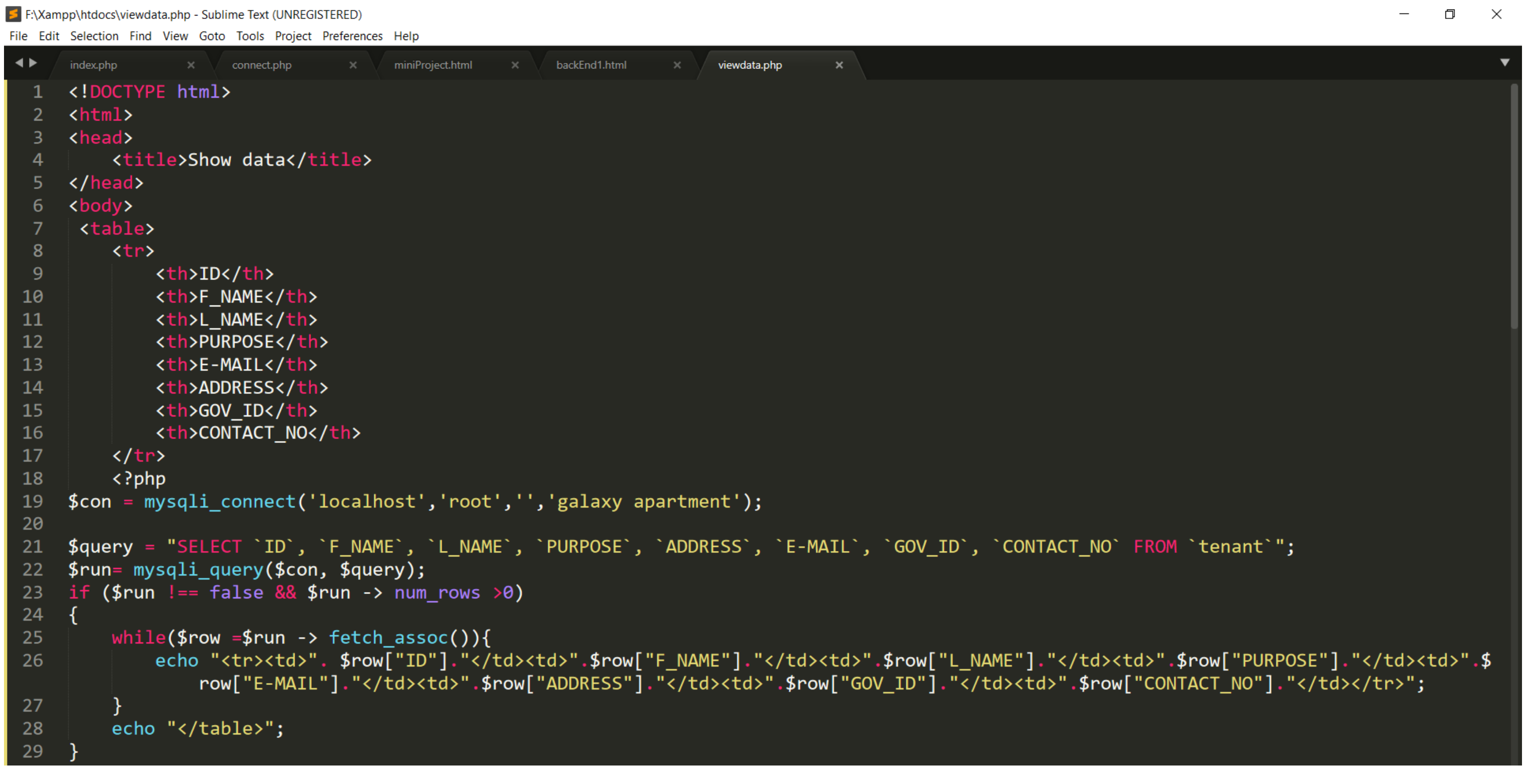

F:\Xampp\htdocs\connect.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
index.php
connect.php
miniProject.html
backEnd1.html
viewdata.php

```php
<?php
$con = mysqli_connect('localhost','root','','galaxy apartment');
$id= $_POST['id'];
$f_name= $_POST['f_name'];
$l_name= $_POST['l_name'];
$purpose= $_POST['purpose'];
$address= $_POST['address'];
$e_mail= $_POST['e_mail'];
$gov_id= $_POST['gov_id'];
$contact= $_POST['contact'];
$query = "INSERT INTO tenant(`ID`, `F_NAME`, `L_NAME`, `PURPOSE`, `ADDRESS`, `E-MAIL`, `GOV_ID`, `CONTACT_NO`) VALUES ('$id','$f_name','$l_name','$purpose','$address','$e_mail','$gov_id','$contact')";

$run= mysqli_query($con, $query);
if ($run==true)
    echo '<script>alert("data inserted")</script>';
else
    echo '<script>alert("error")</script>';

?>
```

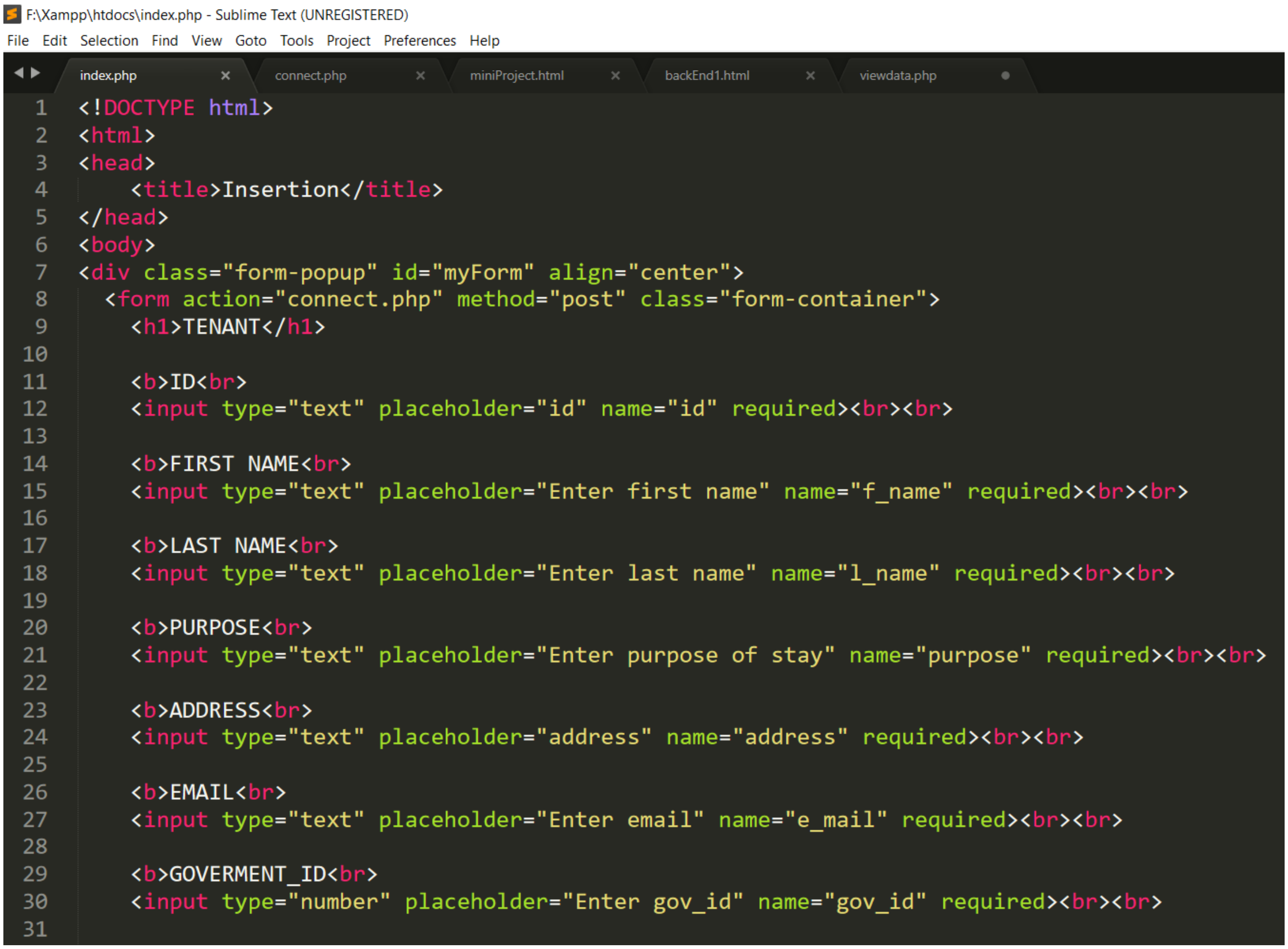
PHP SHOW DATA:


F:\Xampp\htdocs\viewdata.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
index.php
connect.php
miniProject.html
backEnd1.html
viewdata.php

```php
<!DOCTYPE html>
<html>
<head>
    <title>Show data</title>
</head>
<body>
 <table>
    <tr>
        <th>ID</th>
        <th>F_NAME</th>
        <th>L_NAME</th>
        <th>PURPOSE</th>
        <th>E-MAIL</th>
        <th>ADDRESS</th>
        <th>GOV_ID</th>
        <th>CONTACT_NO</th>
    </tr>
    <?php
$con = mysqli_connect('localhost','root','','galaxy apartment');

$query = "SELECT `ID`, `F_NAME`, `L_NAME`, `PURPOSE`, `ADDRESS`, `E-MAIL`, `GOV_ID`, `CONTACT_NO` FROM `tenant`";
$run= mysqli_query($con, $query);
if ($run !== false && $run -> num_rows >0)
{
    while($row =$run -> fetch_assoc()){
        echo "<tr><td>". $row["ID"]."</td><td>".$row["F_NAME"]."</td><td>".$row["L_NAME"]."</td><td>".$row["PURPOSE"]."</td><td>".$row["E-MAIL"]."</td><td>".$row["ADDRESS"]."</td><td>".$row["GOV_ID"]."</td><td>".$row["CONTACT_NO"]."</td></tr>";
    }
    echo "</table>";
}
```

DATA INSERTION FORM CODE:


F:\Xampp\htdocs\index.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
index.php
connect.php
miniProject.html
backEnd1.html
viewdata.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Insertion</title>
</head>
<body>
<div class="form-popup" id="myForm" align="center">
  <form action="connect.php" method="post" class="form-container">
    <h1>TENANT</h1>

    <b>ID<br>
    <input type="text" placeholder="id" name="id" required><br><br>

    <b>FIRST NAME<br>
    <input type="text" placeholder="Enter first name" name="f_name" required><br><br>

    <b>LAST NAME<br>
    <input type="text" placeholder="Enter last name" name="l_name" required><br><br>

    <b>PURPOSE<br>
    <input type="text" placeholder="Enter purpose of stay" name="purpose" required><br><br>

    <b>ADDRESS<br>
    <input type="text" placeholder="address" name="address" required><br><br>

    <b>EMAIL<br>
    <input type="text" placeholder="Enter email" name="e_mail" required><br><br>

    <b>GOVERMENT_ID<br>
    <input type="number" placeholder="Enter gov_id" name="gov_id" required><br><br>

```

CHAPTER 7

SNAPSHOTS

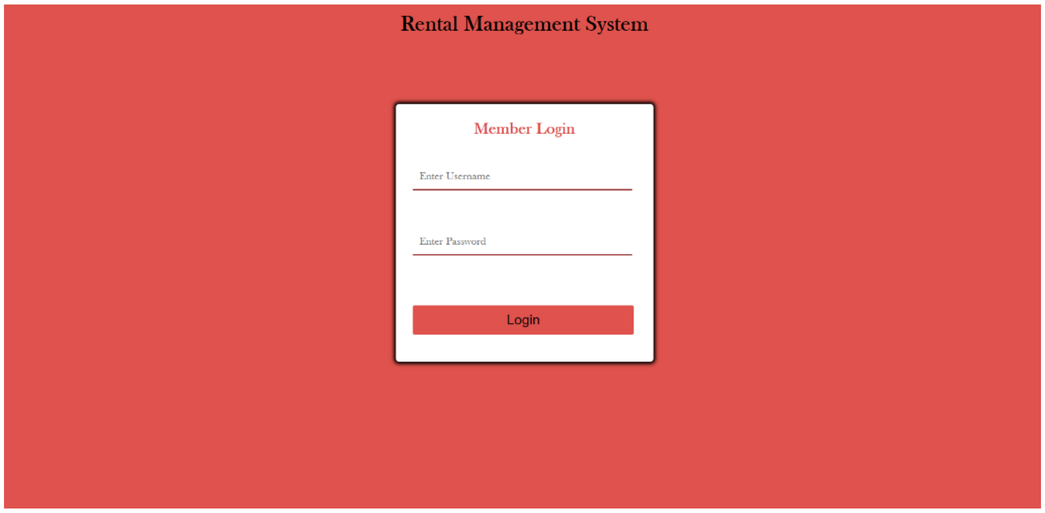


Fig 7.2 Login Page

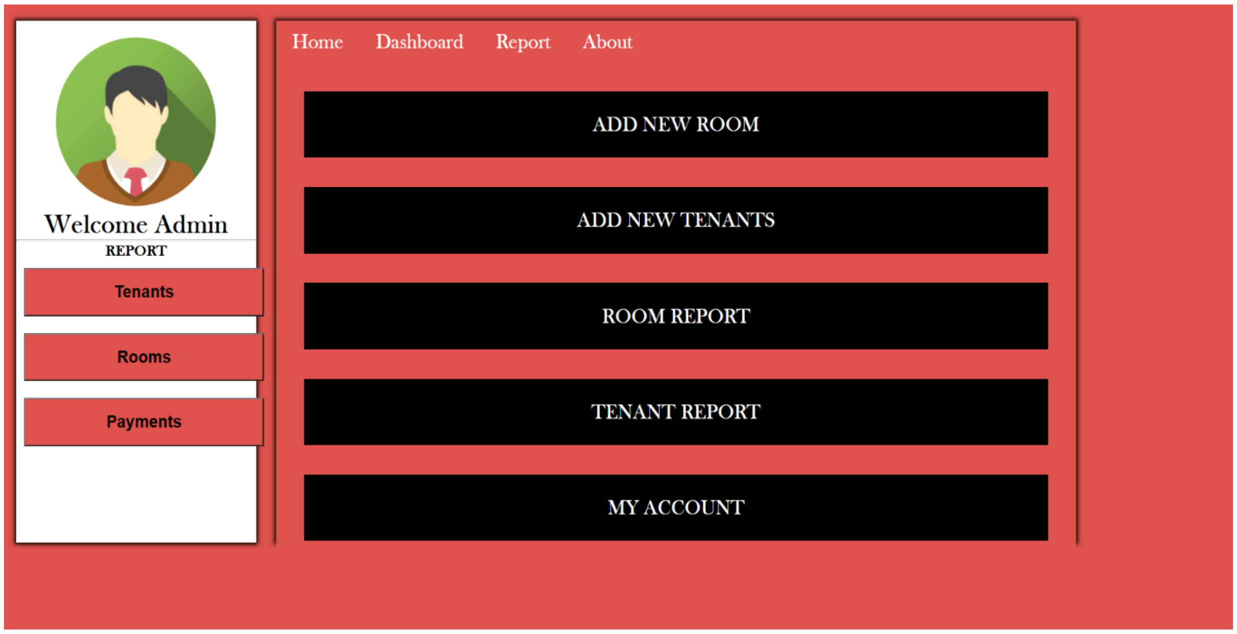
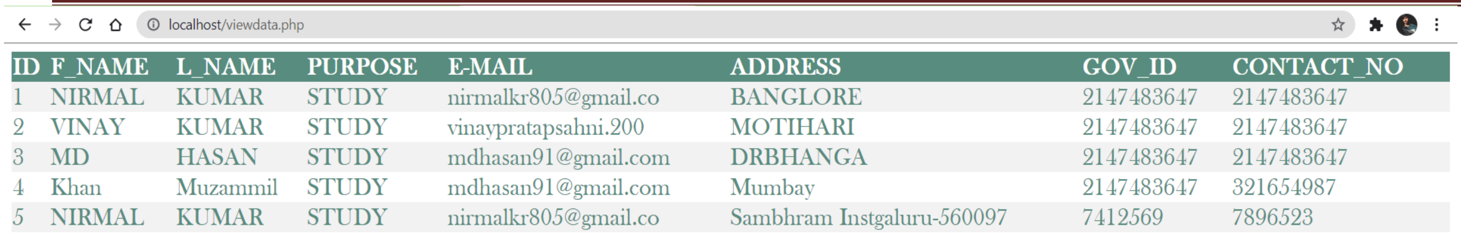


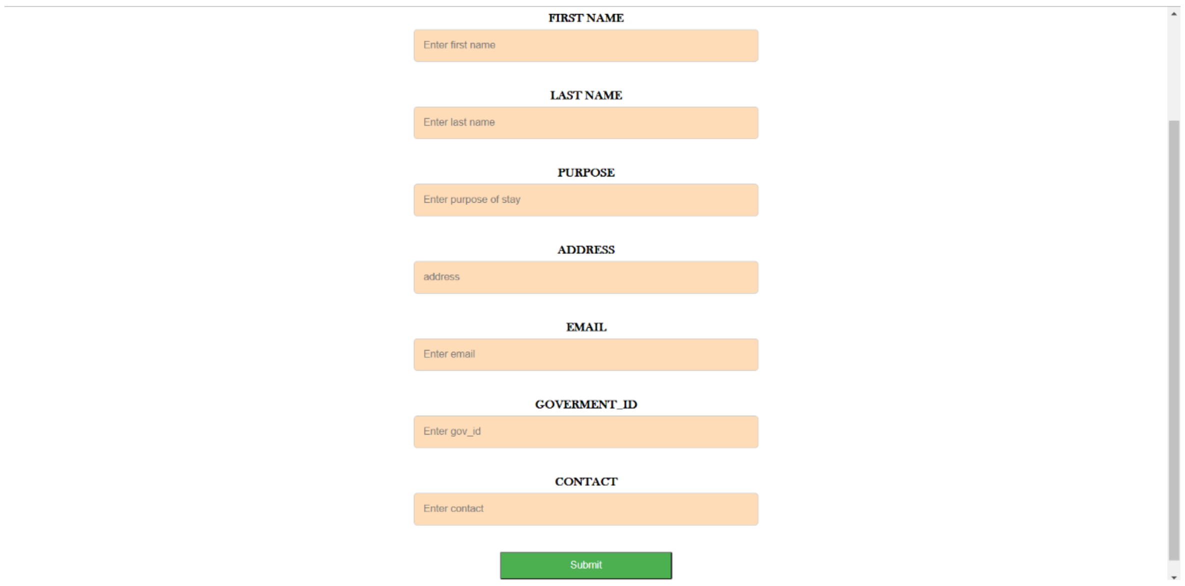
Fig 7.3 Dashboard



localhost/viewdata.php

ID	F_NAME	L_NAME	PURPOSE	E-MAIL	ADDRESS	GOV_ID	CONTACT_NO
1	NIRMAL	KUMAR	STUDY	nirmalkr805@gmail.co	BANGLORE	2147483647	2147483647
2	VINAY	KUMAR	STUDY	vinaypratapsahni.200	MOTIHARI	2147483647	2147483647
3	MD	HASAN	STUDY	mdhasan91@gmail.com	DRBHANGA	2147483647	2147483647
4	Khan	Muzammil	STUDY	mdhasan91@gmail.com	Mumbay	2147483647	321654987
5	NIRMAL	KUMAR	STUDY	nirmalkr805@gmail.co	Sambhram Instgaluru-560097	7412569	7896523

Fig 7.4 Show Tenant Details



FIRST NAME	Enter first name
LAST NAME	Enter last name
PURPOSE	Enter purpose of stay
ADDRESS	address
EMAIL	Enter email
GOVERMENT_ID	Enter gov_id
CONTACT	Enter contact

Fig 7.4 Add Tenant Detail

CHAPTER 8

FUTURE ENHANCEMENT

There are also few features which can be integrated with the system to make it more flexible.

Below list shows the future points to be considered:

- We can implement face detection API for automatic logbook entry.
- Receipt generation after each payment.
- WhatsApp integration to send payment receipt directly to the tenant.
- Complaint section can be added.

CHAPTER 9

CONCLUSION

Since all details are entered electronically in the “Rental Management System” , data will be secured and unharmed. Using this system, we can retrieve tenants information, payment history, room report with a single click. Thus, processing information will be faster and easy unlike traditional book style. It easily eliminate the book keeping task thus reduces human effort.

This helps the owner to retrieve the correct details of the each and every tenants correctly. The owner can analyze and can provide better service in future. Large amount of data stored in the database can be used to analyze many more things in future like behavior of tenants, need of tenants according to changing world.

CHAPTER 10

BIBLIOGRAPHY

BOOK REFERENCES:

[1] Database System Models, Languages, Rameez Elmasri and Sham Kant B. Navathe, 7th Edition, 2017 Pearson.

[2] Learning PHP, MySQL, JavScript, CSS & HTML5 by Robin Nixon.

WEBSITES:

- **www.w3schools.com**
- **www.youtube.com**
- **www.geekforgeek.com**