1. **SYNOPSIS**

A home rental website is an online platform that facilitates the rental process of residential properties between property owners and tenants. It provides a convenient way for property owners to list their properties and for tenants to search for suitable rental homes. Home rental websites typically include a database of available properties with details such as rental rates, location, amenities, and photos.

Users can search for properties by specifying their requirements, such as the number of bedrooms, location, rental price, and other relevant features. They can also contact the property owners or managers directly through the website to schedule a viewing or make an offer.

Home rental websites provide a range of benefits for both property owners and tenants. For property owners, they offer a convenient way to market their properties and find suitable tenants, while for tenants, they provide a platform for finding a suitable rental home with ease. Additionally, these websites often offer features such as online payment options, background checks, and lease agreements, making the rental process more streamlined and secure.

Overall, home rental websites have revolutionized the rental market, making it easier and more efficient for property owners and tenants to connect and rent homes.

**2. SYSTEM ANALYSIS**

**2.1 Existing System**

The existing system of home rental is primarily offline, where property owners and tenants rely on traditional methods to find and rent properties. Property owners usually advertise their properties through classified ads, word of mouth, or through real estate agents. Tenants, on the other hand, search for available properties through newspapers, bulletin boards, or by driving around neighborhoods.

This traditional approach to home rental can be time-consuming, costly, and less efficient compared to modern online rental systems. It often requires physical meetings between property owners and tenants, which can be difficult to schedule, especially for those who live far from the property.

However, with the advent of home rental websites and mobile applications, the rental process has become more convenient, efficient, and cost-effective. These online platforms offer a centralized database of available rental properties, allowing tenants to search and filter properties based on their preferences, such as location, price, and amenities.

Furthermore, home rental websites offer features such as online rental payments, digital lease agreements, and background checks, which make the rental process more secure and streamlined. This online rental system has also made it possible for property owners to manage their properties remotely, which saves time and money while increasing efficiency.

In summary, while the existing system of home rental relies heavily on traditional offline methods, the emergence of online home rental platforms has revolutionized the industry and provided a more efficient, cost-effective, and convenient alternative for property owners and tenants alike.

**2.2 Proposed System**

A proposed system for home rental online would be an advanced online platform that connects property owners and tenants seamlessly. It would offer a centralized database of available rental properties with detailed descriptions, photos, and video tours of each property. The platform would also provide advanced search and filtering options, allowing tenants to find properties that match their preferences and requirements quickly.

The proposed system would also integrate online rental payments, digital lease agreements, and background checks, which would streamline the rental process, eliminate the need for physical meetings, and make the rental process more secure.

Additionally, the proposed system would feature an advanced messaging system that enables tenants and property owners to communicate directly, allowing them to discuss property details, ask questions, and negotiate rental terms easily.

The platform would also provide property owners with a range of management tools to help them manage their properties remotely. These tools would allow property owners to monitor their properties, track rental payments, schedule maintenance, and communicate with tenants easily.

Finally, the proposed system would provide a review and rating system, which would enable tenants to rate and provide feedback on properties they have rented. This feature would help other tenants make more informed decisions when searching for rental properties.

In conclusion, the proposed online system for home rental would be a comprehensive, advanced platform that connects property owners and tenants seamlessly, making the rental process more efficient, convenient, and secure for all parties involved**.**

**Feasibility Study**

Three key considerations involved in the feasibility analysis are,

* Economical Feasibility
* Techinical Feasibiliy
* Social Feasibility

**Economical Feasibility**

The economical feasibility of an online home rental platform would depend on various factors such as the initial investment, operational costs, and revenue generation.

The initial investment required for developing an online home rental platform can be significant, including expenses for software development, web hosting, marketing, and legal fees. However, with the increasing demand for online rental services and the potential for revenue generation, the investment can be considered worthwhile.

The operational costs of an online home rental platform would include expenses for website maintenance, customer service, and payment processing. These costs can be relatively low compared to traditional rental systems since online rental platforms can automate many of the processes involved in property management, such as rent collection and maintenance scheduling.

Revenue can be generated from various sources, including subscription fees, commission-based fees, and advertising revenue. Subscription fees can be charged to property owners who want to list their properties on the platform, while commission-based fees can be charged for each successful rental transaction. Additionally, the platform can generate revenue from advertising fees by allowing third-party advertisers to display ads on the platform.

Overall, the economical feasibility of an online home rental platform depends on the market demand, competition, and the platform's ability to attract and retain customers. However, with the increasing trend towards online rental services, the potential for revenue generation is high, making an online home rental platform an economically feasible option.

**Technical Feasibility**

The technical feasibility of a home rental online system is an important consideration to determine whether it can be developed and implemented successfully. The following are some key technical factors to consider:

Web development: The online home rental system would require web development expertise to build and maintain the platform. This would include web design, coding, database development, and server management.

Data management: The system would require a robust and secure data management infrastructure to handle large volumes of data, including property listings, tenant information, and financial transactions.

Mobile compatibility: The system should be mobile-friendly, allowing users to access the platform from their mobile devices. This would require the development of responsive design and mobile applications.

Security: The system must be designed with strong security features to protect user data, prevent unauthorized access, and ensure secure financial transactions.

Scalability: The system must be designed to handle large volumes of traffic and data as it grows. It should be able to scale up its infrastructure and resources to meet demand without compromising performance.

Integration: The system should be able to integrate with third-party services, such as payment gateways, email providers, and customer relationship management (CRM) tools.

In conclusion, a home rental online system is technically feasible, but it requires a team of experienced developers, robust data management infrastructure, mobile compatibility, security features, scalability, and the ability to integrate with third-party services.

**Social Feasibility**

The social feasibility of a home rental online system is an essential factor to consider, as it determines whether the system can be accepted and adopted by the target audience. The following are some key social factors to consider:

User adoption: The system must be user-friendly, intuitive, and easy to navigate, ensuring that users of all ages and backgrounds can use the platform with ease.

Trust and credibility: The system must be designed to build trust and credibility with users, ensuring that their personal information is secure, financial transactions are safe, and the properties listed on the platform are legitimate.

Social impact: The system must consider the social impact of its services on the local community, including the impact on the housing market, affordability of rental homes, and the rights and responsibilities of landlords and tenants.

Community engagement: The system must engage with the local community to ensure that its services are aligned with their needs and concerns. This can include working with local organizations, seeking feedback from users, and participating in community events.

Ethical considerations: The system must be designed with ethical considerations in mind, including ensuring that properties listed on the platform meet ethical and legal standards, and that tenants and landlords are treated fairly and equitably.

In conclusion, a home rental online system must consider social factors, including user adoption, trust and credibility, social impact, community engagement, and ethical considerations, to ensure its acceptance and adoption by the target audience.

**2.3 Scope of the Project**

The scope of the project for a home rental online system would include the development, implementation, and maintenance of a platform that connects property owners and tenants seamlessly. The platform would offer a centralized database of available rental properties with detailed descriptions, photos, and video tours of each property. It would also provide advanced search and filtering options, online rental payments, digital lease agreements, background checks, and an advanced messaging system.

**2.4 Objectives of the Project**

* To provide a convenient and secure online platform for property owners and tenants to connect and facilitate the rental process.

**3. SYSTEM SPECIFICATION**

**3.1 Hardware requirements**

* PROCESSOR : PREFERABLY 1.5 GHZ O GREATER
* RAM : 1GB OR GREATER
* MOITOR : 15” COLOR
* HARD DISK : 10GB
* KEYBOARD : STANDARD 102 KEYS
* MOUSE : 2 BUTSTONS

**3.2 Software Requirements**

* OPERATING SYSTEM : WINDOWS 8,10
* LANGUAGE : Frontend- PHP

Backend- MySQL

* ENVIRONMENT : Apache Tomcat

**3.3 SOFTWARE SPECIFICATION**

**PHP**

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

* PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
* PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
* It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
* PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
* PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
* PHP is forgiving: PHP language tries to be as forgiving as possible.
* PHP Syntax is C-Like.

**Common uses of PHP**

* PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
* PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
* You add, delete, modify elements within your database through PHP.
* Access cookies variables and set cookies.
* Using PHP, you can restrict users to access some pages of your website.
* It can encrypt data.

**Characteristics of PHP**

Five important characteristics make PHP's practical nature possible −

* Simplicity
* Efficiency
* Security
* Flexibility
* Familiarity

"Hello World" Script in PHP

Live Demo

<html>

<head>

<title>Hello World</title>

</head>

<body>

<?php echo "Hello, World!";?>

</body>

</html>

The PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output. All PHP code must be included inside one of the three special markup tags ATE are recognised by the PHP Parser.

<?php PHP code goes here ?>

<? PHP code goes here ?>

<script language = "php"> PHP code goes here </script>

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

**Web Server**

PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here − <https://httpd.apache.org/download.cgi>

**Database**

PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here − <https://www.mysql.com/downloads/>

**PHP Parser**

In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

**PHP Parser Installation**

Before you proceed it is important to make sure that you have proper environment setup on your machine to develop your web programs using PHP.

Type the following address into your browser's address box.

<http://127.0.0.1/info.php>

If this displays a page showing your PHP installation related information then it means you have PHP and Webserver installed properly. Otherwise you have to follow given procedure to install PHP on your computer.

**Apache Configuration**

If you are using Apache as a Web Server then this section will guide you to edit Apache Configuration Files.

PHP.INI File Configuration

The PHP configuration file, php.ini, is the final and most immediate way to affect PHP's functionality.

**Windows IIS Configuration**

To configure IIS on your Windows machine you can refer your IIS Reference Manual shipped along with IIS.

**MySQL**

**Database**

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

A Relational DataBase Management System (RDBMS) is a software that −

* Enables you to implement a database with tables, columns and indexes.
* Guarantees the Referential Integrity between rows of various tables.
* Updates the indexes automatically.
* Interprets an SQL query and combines information from various tables.

**RDBMS Terminology**

A few definitions related to the database.

* Database − A database is a collection of tables, with related data.
* Table − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
* Column − One column (data element) contains data of one and the same kind, for example the column postcode.
* Row − A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
* Redundancy − Storing data twice, redundantly to make the system faster.
* Primary Key − A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
* Foreign Key − A foreign key is the linking pin between two tables.
* Compound Key − A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
* Index − An index in a database resembles an index at the back of a book.
* Referential Integrity − Referential Integrity makes sure that a foreign key value always points to an existing row.

**MySQL Database**

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

**Apache Tomcat**

Apache Tomcat (called "Tomcat" for short) is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) implementation of the [Java Servlet](https://en.wikipedia.org/wiki/Java_Servlet), JavaServer Pages, [Java Expression Language](https://en.wikipedia.org/wiki/Unified_Expression_Language) and [WebSocket](https://en.wikipedia.org/wiki/WebSocket) technologies. Tomcat provides a "pure Java" [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) [web server](https://en.wikipedia.org/wiki/Web_server) environment in which [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) code can run.Tomcat is developed and maintained by an open community of developers under the auspices of the [Apache Software Foundation](https://en.wikipedia.org/wiki/Apache_Software_Foundation), released under the [Apache License](https://en.wikipedia.org/wiki/Apache_License) 2.0 license.

Tomcat 4.x was released with Catalina (a servlet container), Coyote (an HTTP connector) and Jasper (a JSP engine).

**Catalina**

Catalina is Tomcat's servlet container. Catalina implements Sun Microsystems' specifications for servlet and JavaServer Pages (JSP). In Tomcat, a Realm element represents a "database" of usernames, passwords, and roles (similar to Unix groups) assigned to those users. Different implementations of Realm allow Catalina to be integrated into environments where such authentication information is already being created and maintained, and then use that information to implement Container Managed Security as described in the Servlet Specification.

**Coyote**

Coyote is a Connector component for Tomcat that supports the HTTP 1.1 protocol as a web server. This allows Catalina, nominally a Java Servlet or JSP container, to also act as a plain web server that serves local files as HTTP documents.[4] Coyote listens for incoming connections to the server on a specific TCP port and forwards the request to the Tomcat Engine to process the request and send back a response to the requesting client. Another Coyote Connector, Coyote JK, listens similarly but instead forwards its requests to another web server, such as Apache, using the JK Protocol.This usually offers better performance.[citation needed]

**Jasper**

Jasper is Tomcat's JSP Engine. Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them.

As of version 5, Tomcat uses Jasper 2, which is an implementation of the Sun Microsystems' JSP 2.0 specification. From Jasper to Jasper 2, important features were added:

JSP Tag library pooling – Each tag markup in JSP file is handled by a tag handler class. Tag handler class objects can be pooled and reused in the whole JSP servlet.

Background JSP compilation – While recompiling modified JSP Java code, the older version is still available for server requests. The older JSP servlet is deleted once the new JSP servlet has finished being recompiled.

Recompile JSP when included page changes – pages can be inserted and included into a JSP at runtime. The JSP will not only be recompiled with JSP file changes but also with included page changes.

JDT Java compiler – Jasper 2 can use the Eclipse JDT (Java Development Tools) Java compiler instead of Ant and javac.

Three new components were added with the release of Tomcat 7:

**Cluster**

This component has been added to manage large applications. It is used for load balancing that can be achieved through many techniques. Clustering support currently requires the JDK version 1.5 or higher.

**High availability**

A high-availability feature has been added to facilitate the scheduling of system upgrades (e.g. new releases, change requests) without affecting the live environment. This is done by dispatching live traffic requests to a temporary server on a different port while the main server is upgraded on the main port. It is very useful in handling user requests on high-traffic web applications.

**Web application**

It has also added user- as well as system-based web applications enhancement to add support for deployment across the variety of environments. It also tries to manage sessions as well as applications across the network. Tomcat is building additional components. A number of additional components may be used with Apache Tomcat. These components may be built by users should they need them or they can be downloaded from one of the mirrors.

**Features**

Tomcat 7.x implements the Servlet 3.0 and JSP 2.2 specifications. It requires Java version 1.6, although previous versions have run on Java 1.1 through 1.5. Versions 5 through 6 saw improvements in garbage collection, JSP parsing, performance and scalability. Native wrappers, known as "Tomcat Native", are available for Microsoft Windows and Unix for platform integration.

Tomcat 8.x implements the Servlet 3.1 and JSP 2.3 Specifications. Apache Tomcat 8.5.x is intended to replace 8.0.x and includes new features pulled forward from Tomcat 9.0.x. The minimum Java version and implemented specification versions remain unchanged.

**4. PROJECT DESCRIPTION**

**4.1 About the Project**

The project is a proposed online platform for home rental services. The platform would aim to connect property owners and tenants seamlessly by providing a centralized database of available rental properties, advanced search and filtering options, online rental payments, digital lease agreements, background checks, and an advanced messaging system.

The project's objectives include streamlining the rental process, providing property owners with management tools, building trust and credibility with users, considering the social impact of the services on the local community, and ensuring user-friendliness and security.

The project would require a team of experienced developers, web designers, and data management experts to develop and maintain the platform. The platform would also require a robust data management infrastructure, mobile compatibility, security features, scalability, and the ability to integrate with third-party services.

In addition to technical considerations, the project would also require social considerations, including user adoption, trust and credibility, social impact, community engagement, and ethical considerations.

Overall, the project aims to provide a convenient, secure, and user-friendly platform for property owners and tenants to connect and facilitate the rental process while ensuring social responsibility and ethical considerations.

**Module Description**

* **User Management:**

This module manages user registration, login, and profile management. It includes features such as email verification, password reset, and user roles and permissions.

* **Property Management:**

This module handles property listings, including property details, photos, availability, and rental rates. Property owners can manage their listings, including adding or removing properties and updating details.

* **Property Search:**

This module allows tenants to search for available properties based on their preferences, such as location, price range, number of bedrooms, and amenities. The search results should provide details about the properties and contact information for the property owner or manager.

* **Booking and Payment:**

This module handles the booking and payment process, allowing tenants to book and pay for the property online. The system should include a payment gateway that supports multiple payment methods, such as credit/debit cards, bank transfers, and e-wallets.

* **Messaging:**

This module provides a messaging system that allows tenants to communicate directly with property owners or managers. This will facilitate quick and easy communication, making it easier for tenants to ask questions and get clarification about the properties.

**5.SYSTEM DESIGN**

**5.1 Architectural Design**

The most creative and challenging face of the system development is System Design. It provides the understanding and procedural details necessary foe implementing the system recommended in the feasibility study. Design goes through the logical and physical stages of development.

In designing a new system, the system analyst must have a clear understanding of the objectives, which the design is aiming to fulfil. The first step is to determine how the output is to be produced and in what format. Second, input data and master files have to be designed to meet the requirements of the proposed output. The operational phases are handled through program construction and testing.

Design of a system can be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Thus, system design is a solution to “hoe to” approach to the creation of a new system. Thus, important phase provides the understanding and the procedural details necessary for implementing the system recommended in the feasibility study. The design step provides a data design, architectural design, and a procedural design.

**5.2 Input Design**

In the input design, user-oriented inputs are converted into a computer-based system format. It also includes determining the record media, method of input, speed of capture and entry on to the screen. Online data entry accepts commands and data through a keyboard through a keyboard. The major approach to input design is the menu and the prompt design. In each alternative, the user’s options are predefined. The data flow diagram indicates logical data flow, data stores, source and destination. Input data are collected and organized into a group of similar data. Once identified input media are selected for processing.

In this software, importance is given to develop Graphical User Interface (GUI), which is an important factor in developing efficient and user-friendly software. For inputting user data, attractive forms are designed. User can also select desired options from the menu, which provide all possible facilities.

Also, the important input format is designed in such a way that accidental errors are avoided. The user has to input only just the minimum data required, which also helps in avoiding the errors that the users may make. Accurate designing of the input format is very important in developing efficient software. The goal or input design is to make entry as easy logical and free from errors.

**5.3 Output Design**

In the output design, the emphasis is on producing a hard copy of the information requested or displaying the output on the CRT screen in a predetermined format. Two of the most output media today are printers and the screen. Most users now access their reports from a hard copy on screen display. Computer’s output is the most important and direct source of information to the user, efficient, logical, output design should improve the systems relations with the user and help in decision-making.

As the outputs are the most important source of information to the user, better design should improve the system’s relation and also should help in decision-making. The output device’s capability, print capability, response time requirements etc. should also be considered from design elaborates the way output is presented and layout available for capturing information. It’s very helpful to produce the clear, accurate and speedy information for end users

**6. SYSTEM TESTING AND MAINTENANCE**

Testing is a set activity that can be planned and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it is vital success of the system.

**Testing Objectives:**

There are several rules that can serve as testing objectives, they are

* Testing is a process of executing a program with the intent of finding and error
* A good test case is one that has high probability of finding an undiscovered error.
* A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrates that software functions appear to the working according to the specification, that performance requirements appear to have been met.

There are three ways to test a program

* For Correctness
* For Implementation efficiency
* For Computational Complexity.

Tests for correctness are supposed to verify that a program does exactly that it was designed to do. This is much more difficult that it may at first appear, especially for large programs.

Tests for implementation efficiency attempt to find ways to make a correct program faster or use less storage. It is a code-refining process, which re-examines the implementation phase of algorithm development.

Tests for computational complexity amount to an experimental analysis of the complexity of an algorithm or an experimental comparison of two or more algorithms, which solve the same problem.

* Testing Correctness

The following ideas should be a part of any testing plan:

* Preventive Measures
* Spot checks
* Testing all parts of the program
* Test Data
* Looking for trouble
* Time for testing
* Re Testing

The data is entered in all forms separately and whenever an error occurred it is corrected immediately. A quality team deputed by the management verified all the necessary documents and tested the Software while entering the data at all levels. The entire testing process can be divided into 3 phases

* Unit Testing
* Integrated Testing
* Final/System Testing

**6.1 Unit Testing**

As this system was partially GUI based WINDOWS application, the following were tested in this phase

* Tab Order
* Reverse Tab Order
* Field length
* Front end validations

In our system, Unit testing has been successfully handled. The test data was given to each and every module in all respects and got the desired output. Each module has been tested found working properly.

**6.2 Integration Testing**

Test data should be prepared carefully since the data only determines the efficiency and accuracy of the system. Artificial data are prepared solely for testing. Every program valdates the input data.

**6.3 Validation Testing**

In this, all the Code Modules were tested individually one after the other. The following were tested in all modules

* Loop testing
* Boundary Value analysis
* Equivalence Partitioning Testing

In our case all the modules were combined and given the test data. The combined module worked successfully without any side effect on other programs. Everything was found fine working.

**6.4 Maintenance**

The Objectives of this maintenance work are to make sure that the system gets into work all time without any bug. Provision must be for environmental changes which may affect the computer system or the software system. This is called the maintenance of the system. Now a days there is a rapid change in the software world. Due to this rapid change, the system should be capable of adapting these changes. In our project the process can be added with affecting other parts of the system.

Maintenance plays a vital role. The system liable to accept a modification after its implementation. This system has been designed to all new changes. Doing this will not affect the system performance or its accuracy.

In the project system testing is made as follows:

The procedure level testing is made first. By giving improper inputs, the error occurred are noted and eliminated. Then web form level testing is made. For example, storage of data to the table in the manner

In the form, the zero-length username and password are given and checked. Also, the duplication username is given and checked. The client-side validations are made.

The dates are entered in wrong manner and checked. Wrong email-id is given and checked

This is the first step in the system life cycle. Here we implement the tested error-free system in to real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors like runtime errors, long run errors and other maintenance like table verifications and report.

**7. FUTURE ENCHANCEMENT**

* Virtual reality property tours: This could involve the use of virtual reality technology to provide tenants with a fully immersive experience of the property before they rent it.
* Artificial intelligence-powered chatbots: An AI-powered chatbot could provide tenants with instant responses to frequently asked questions and support them throughout the rental process.
* Enhanced data analytics: Advanced data analytics could be used to provide insights into market trends, property performance, and customer behavior, enabling property owners to make data-driven decisions.
* Personalized recommendations: Personalized recommendations could be offered to tenants based on their search history and preferences, increasing the likelihood of finding the perfect rental property.

**8. CONCLUSION**

In conclusion, the proposed home rental online system aims to provide a convenient, secure, and user-friendly platform for property owners and tenants to connect and facilitate the rental process. The platform would include a centralized database of available rental properties, advanced search and filtering options, online rental payments, digital lease agreements, background checks, and an advanced messaging system. The project's objectives include streamlining the rental process, providing property owners with management tools, building trust and credibility with users, considering the social impact of the services on the local community, and ensuring user-friendliness and security.

The project's feasibility was also evaluated, including the economical, technical, and social feasibility. Future enhancements were also discussed, such as virtual reality property tours, artificial intelligence-powered chatbots, enhanced data analytics, personalized recommendations, smart home integration, blockchain-powered rental agreements, and multilingual support.

Overall, the proposed home rental online system has the potential to offer significant benefits to property owners and tenants by streamlining the rental process, providing advanced search and filtering options, and offering management tools to property owners. The success of the project will depend on the ability of the development team to build a user-friendly, secure, and scalable platform that considers the social impact of the services on the local community while keeping up with technological advancements and user demands.

**9.BIBLIOGRAPHY**

**References:**

* "PHP and MySQL Web Development" by Luke Welling and Laura Thomson - This book is a comprehensive guide to developing web applications using PHP and MySQL. It covers everything from basic PHP syntax to advanced topics such as security and performance optimization.
* "Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5" by Robin Nixon - This book provides a beginner-friendly introduction to web development using PHP, MySQL, and other web technologies such as JavaScript, jQuery, CSS, and HTML5.

**Sites**

* "CSS: The Definitive Guide" by Eric Meyer - This book is a comprehensive reference guide to CSS. It covers everything from basic syntax to advanced layout techniques and offers tips and tricks for working with CSS across different browsers and platforms.
* W3Schools (https://www.w3schools.com/) - W3Schools is a popular website for learning web development. It offers free online tutorials and interactive exercises for learning HTML, CSS, JavaScript, PHP, and other web technologies.

**10. ANNEXURE**

**10.1 Level 0 DFD**

HOME RENTAL

ONLINE SYSTEM

Data Store :

Tenants

Process: TENANT

Data Store:

Rental Properties

Process : Owner

Data Store:

Owners Info

External Entity:

Payment Gate

**10.2 Architectural Diagram**

**WEB SERVER**

**Load Balancer**

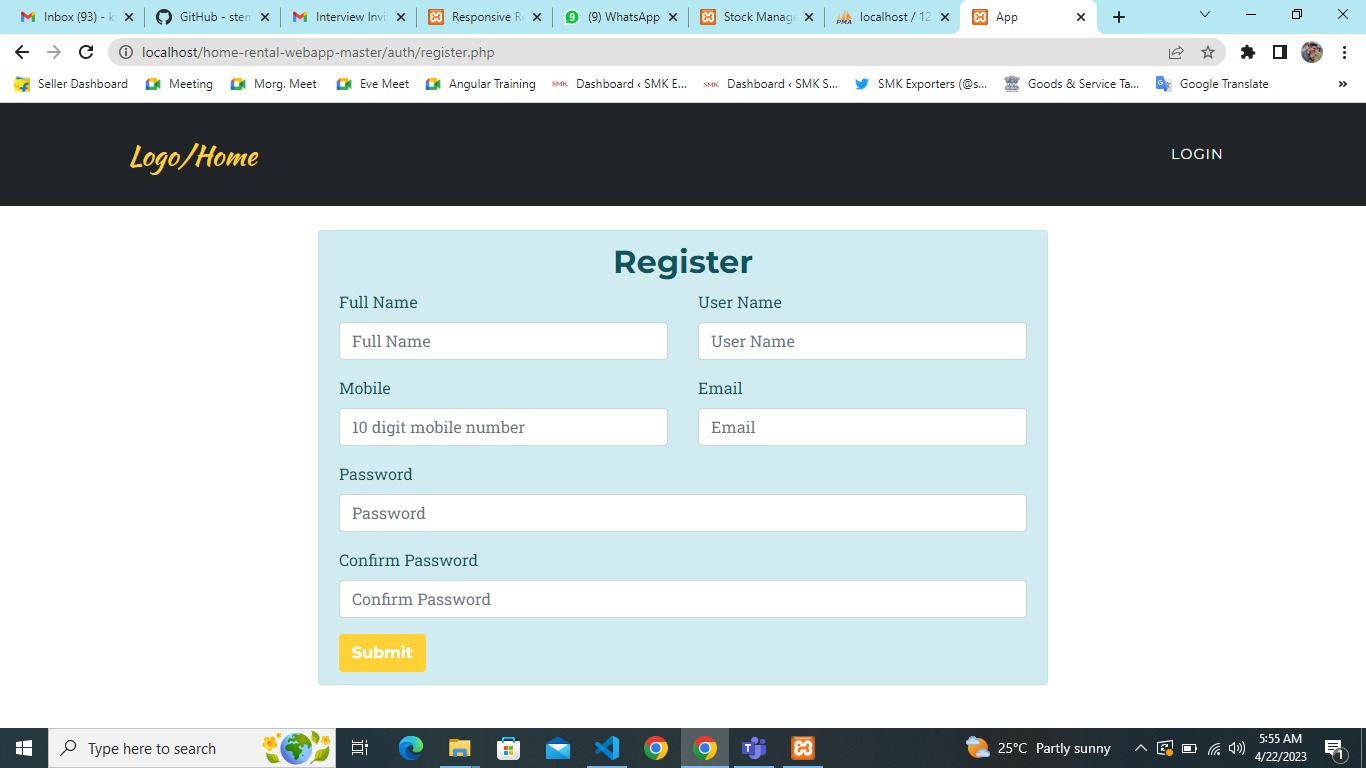
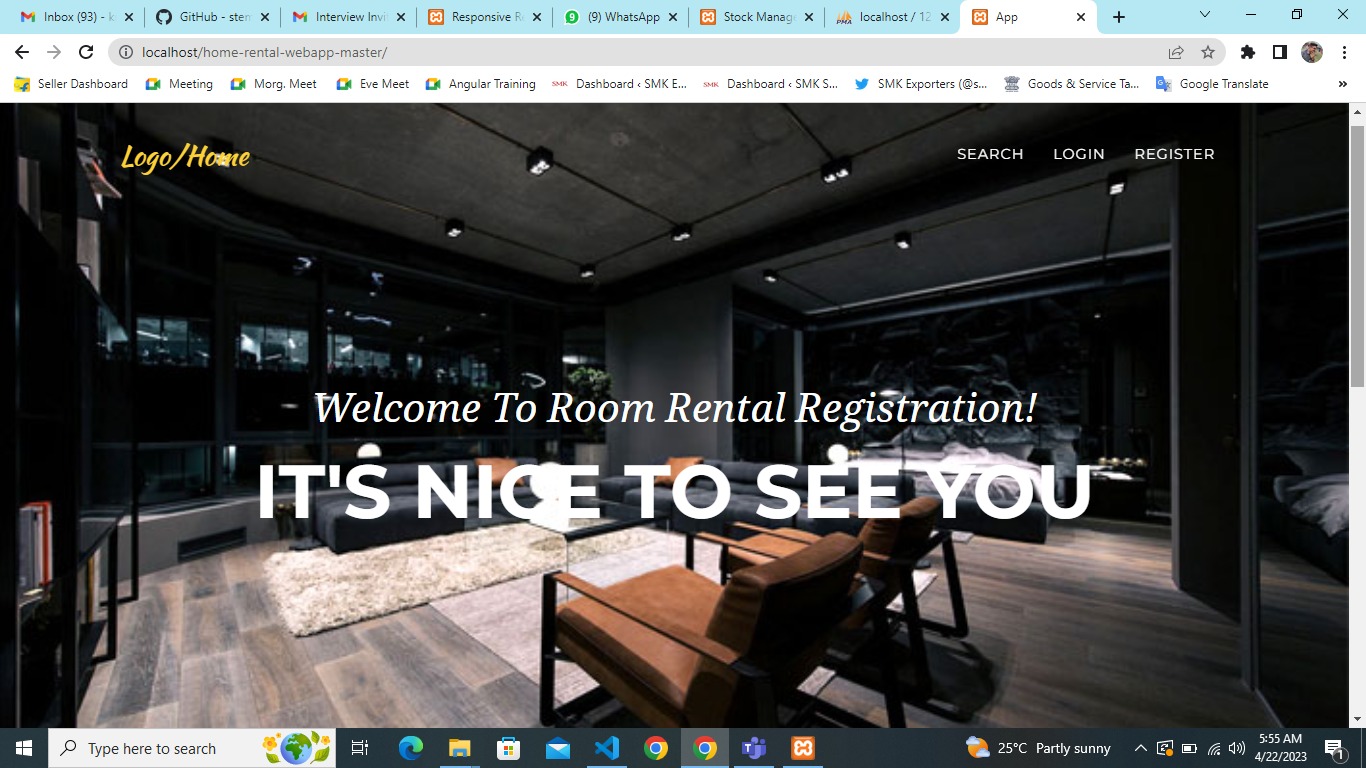
**Application Server**

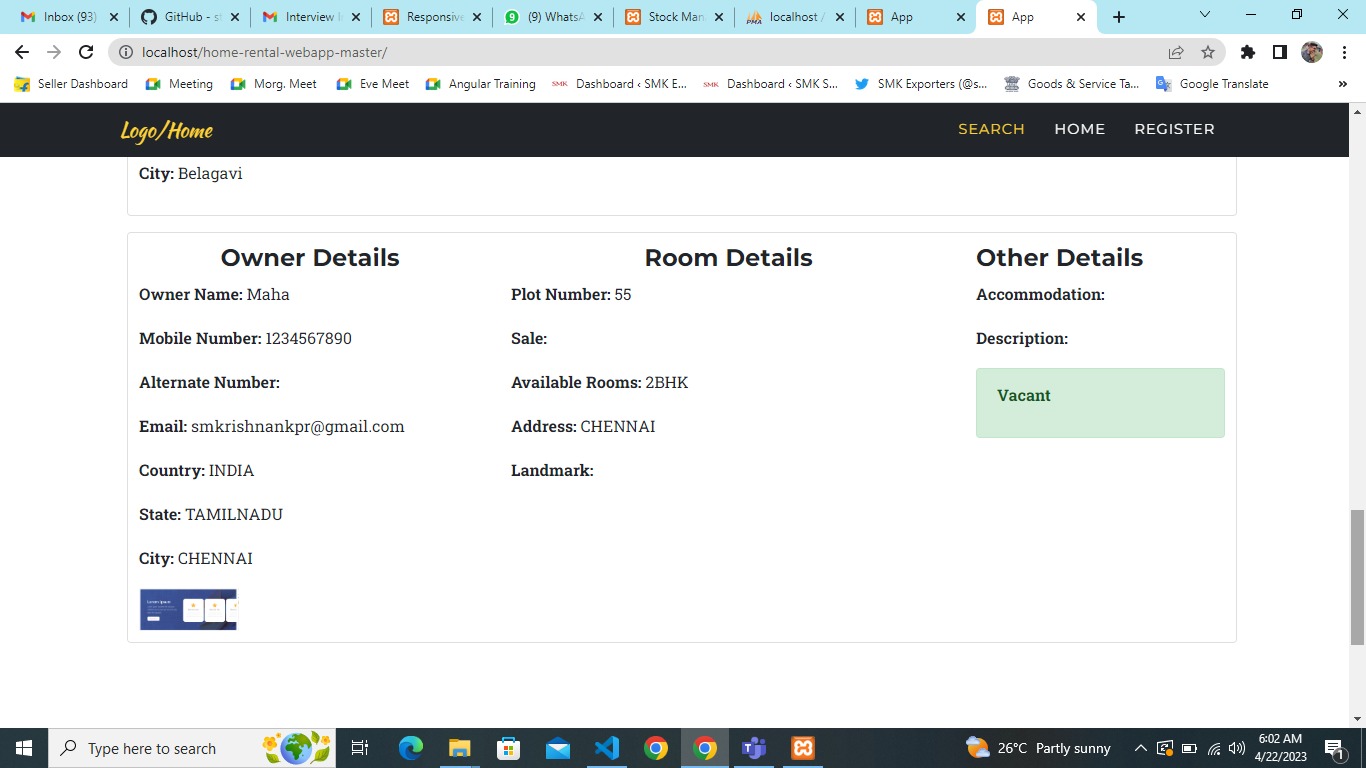
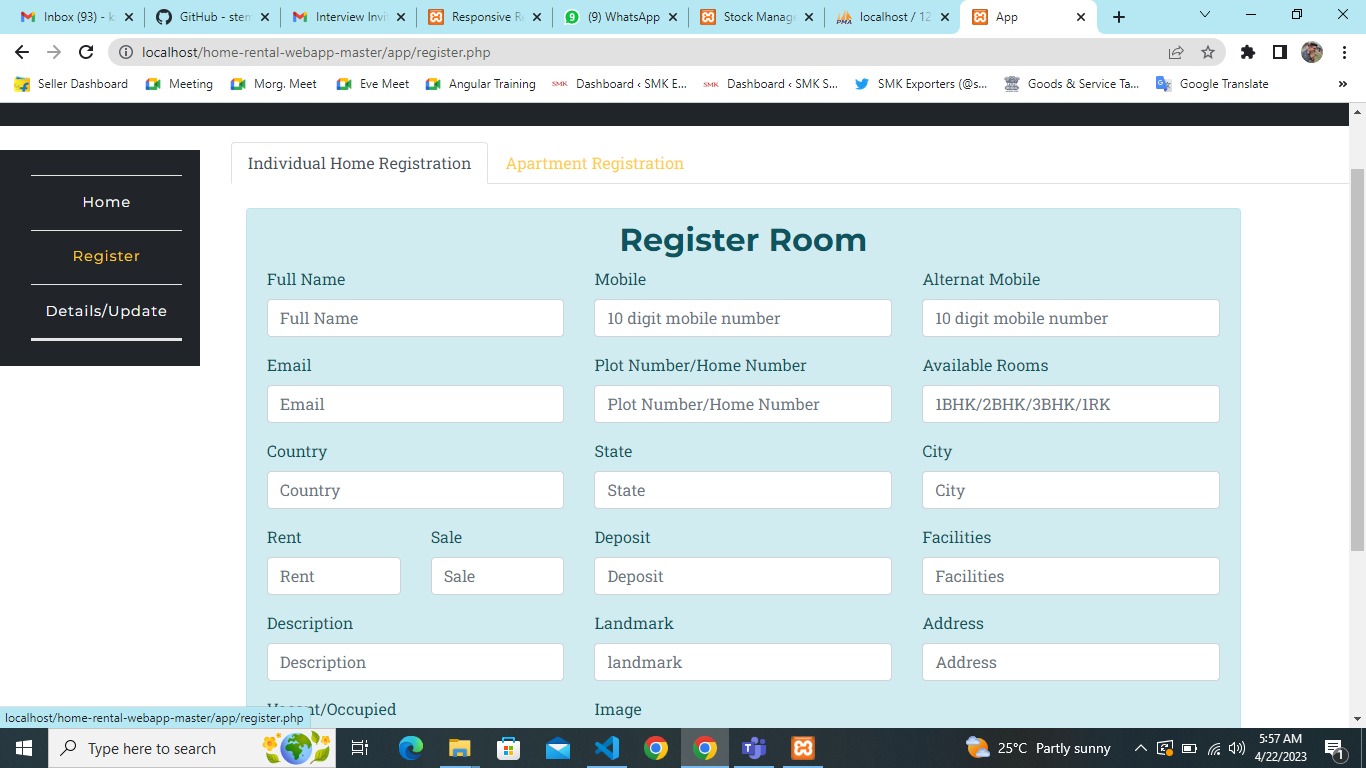
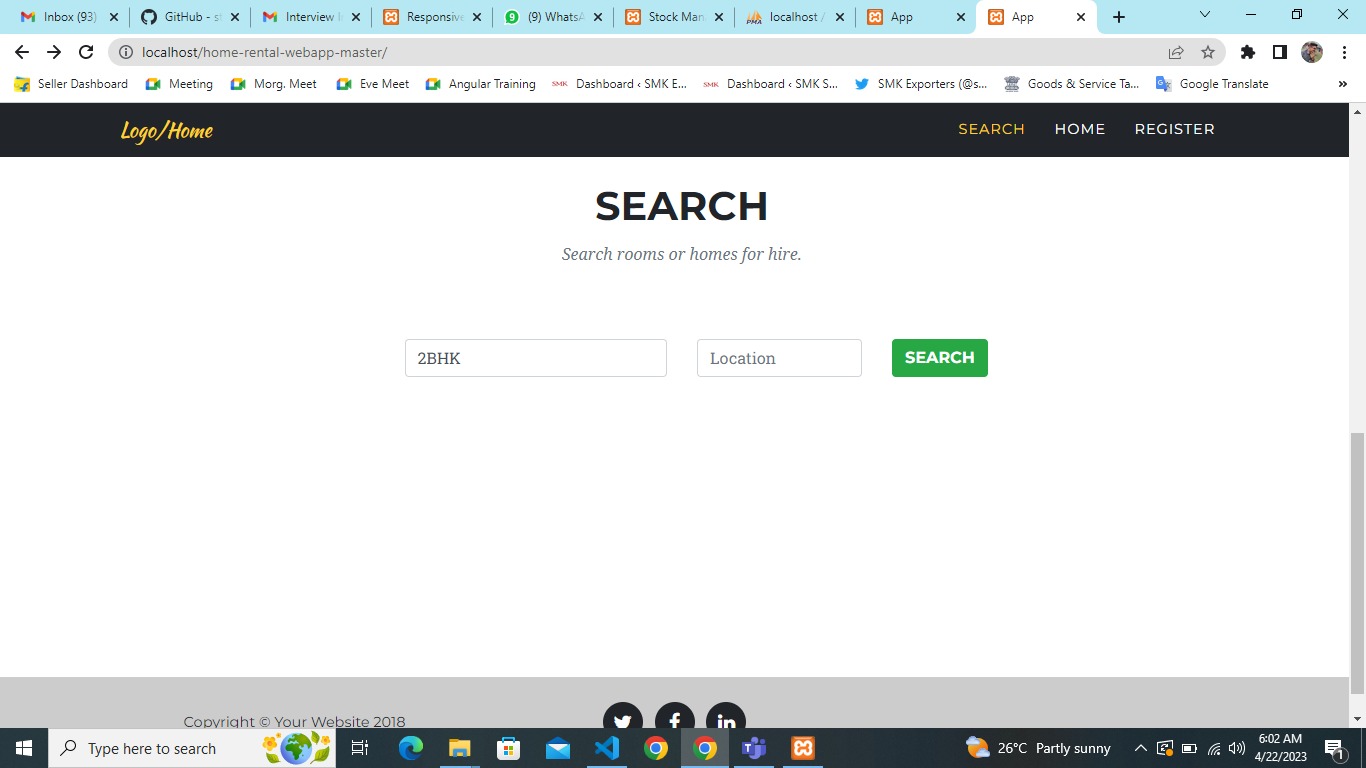
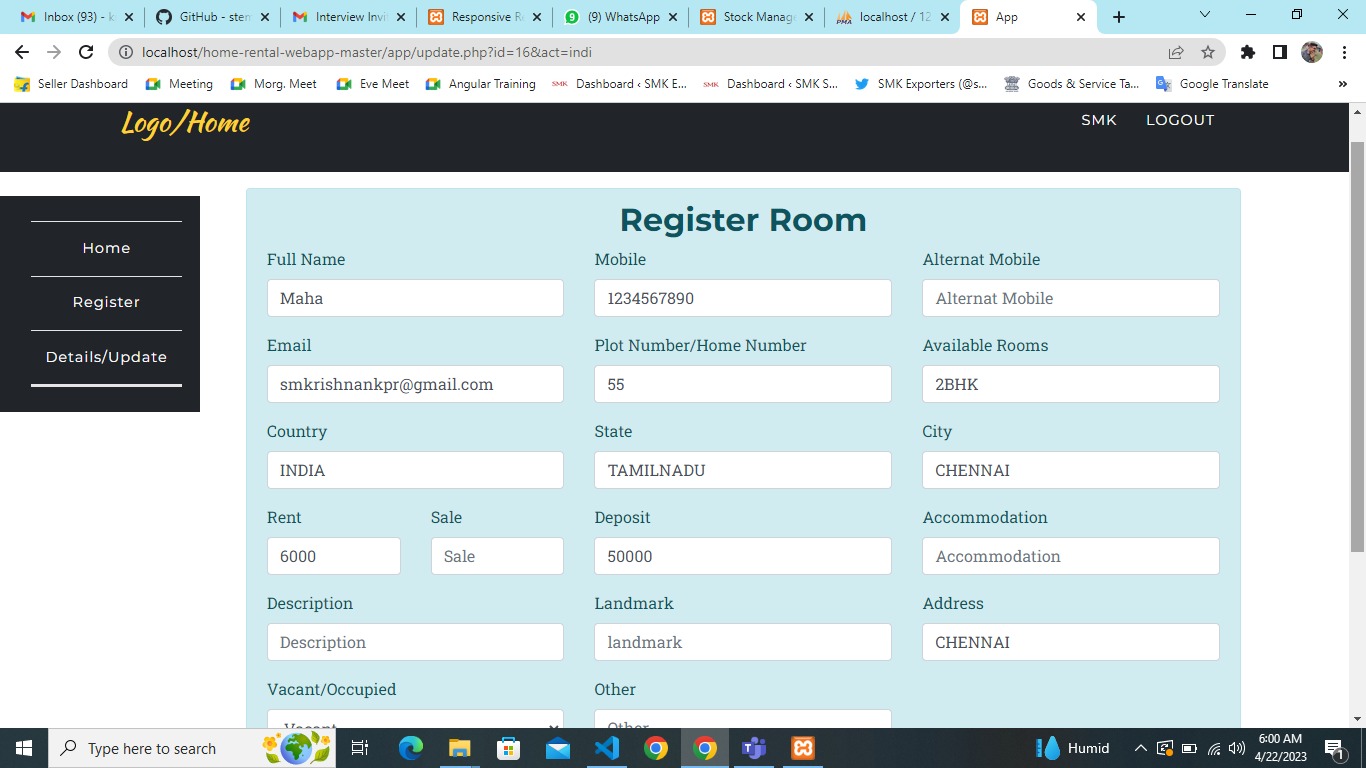
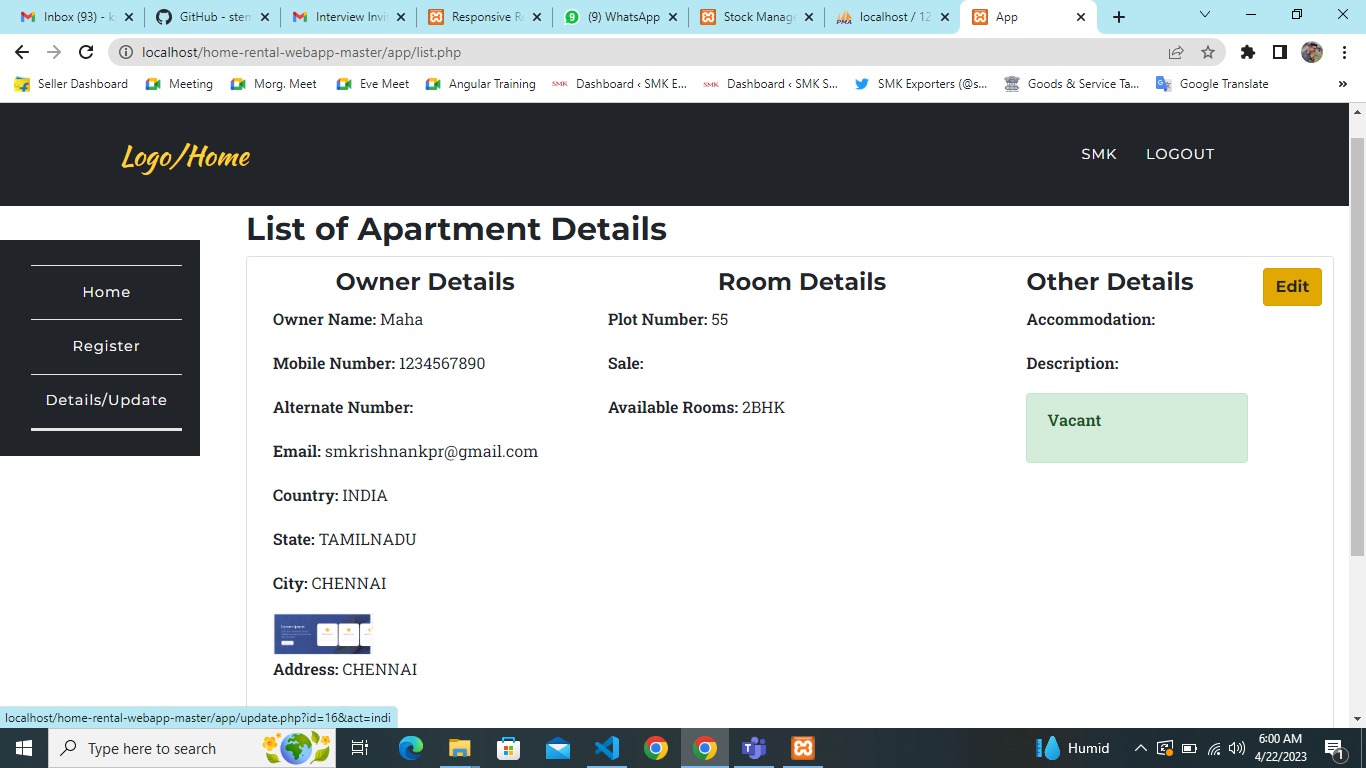
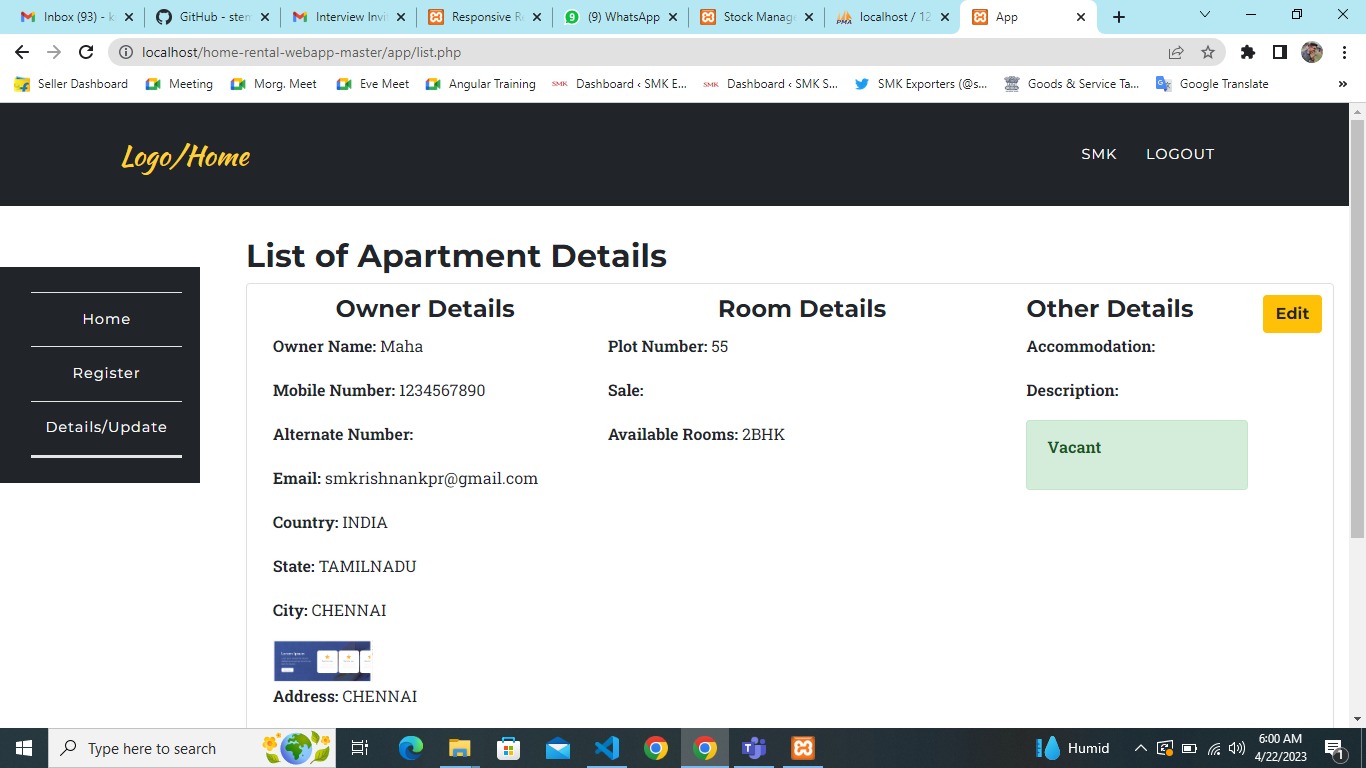
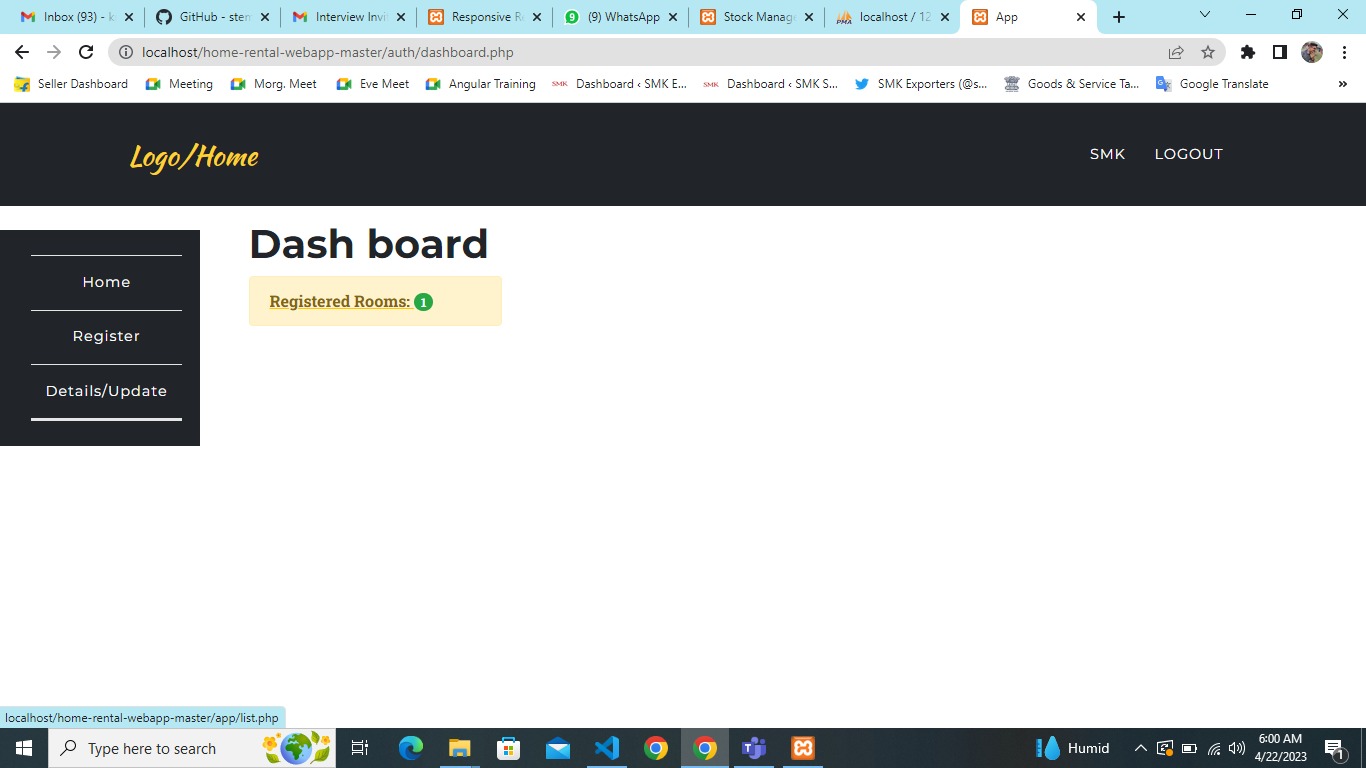
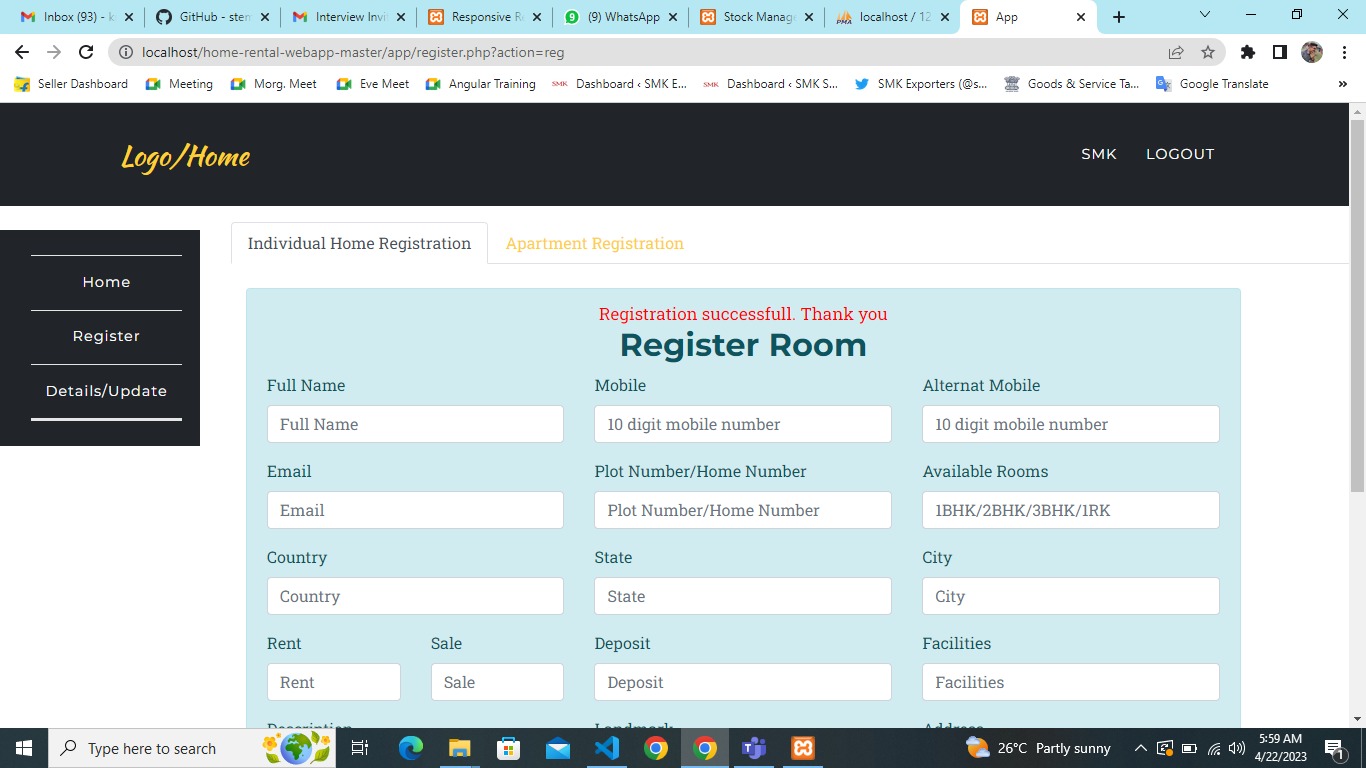
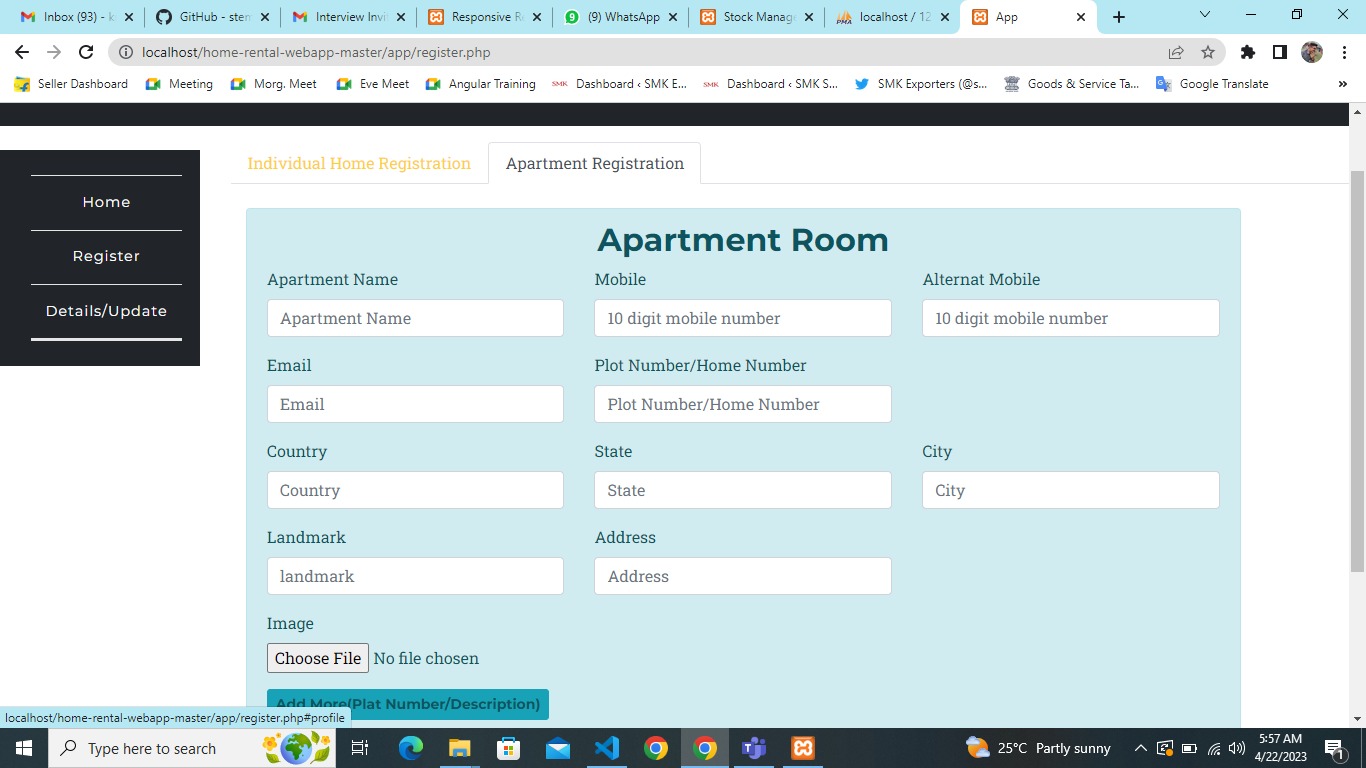
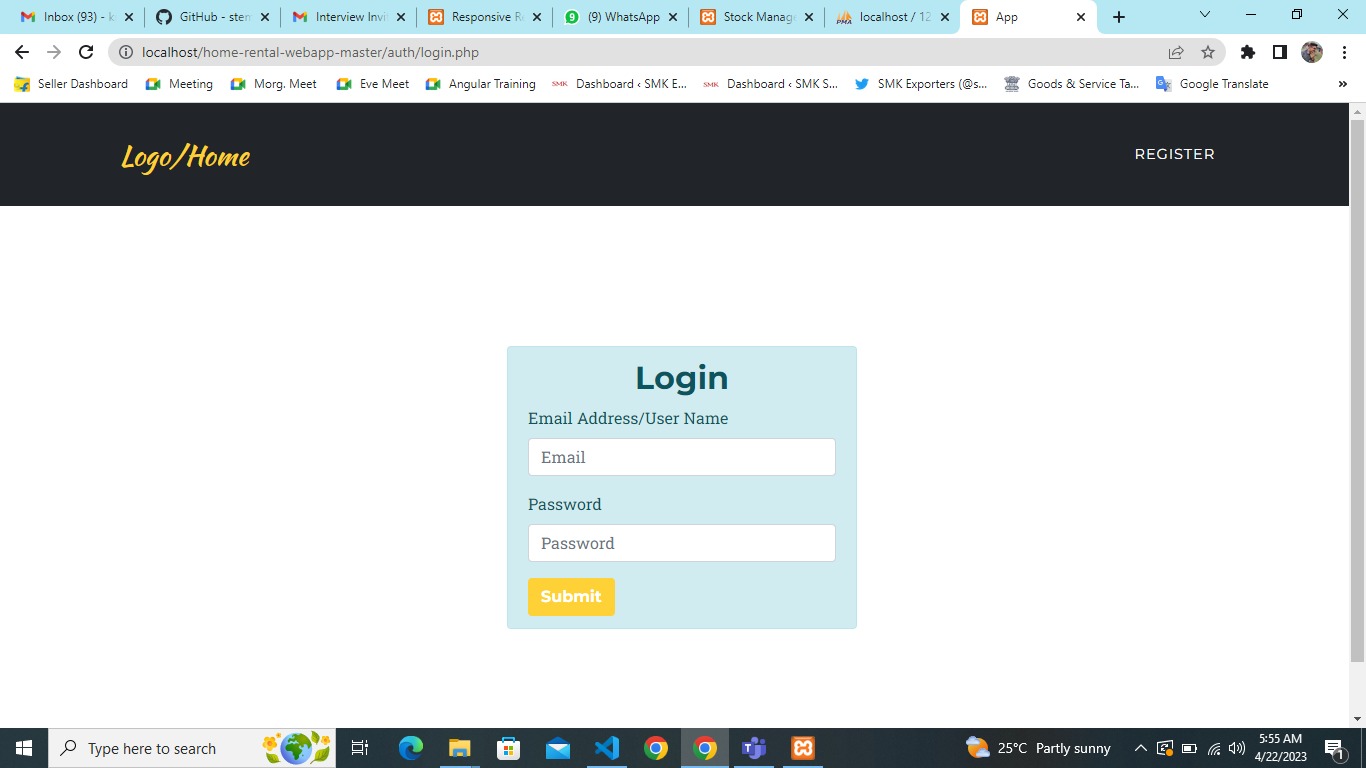
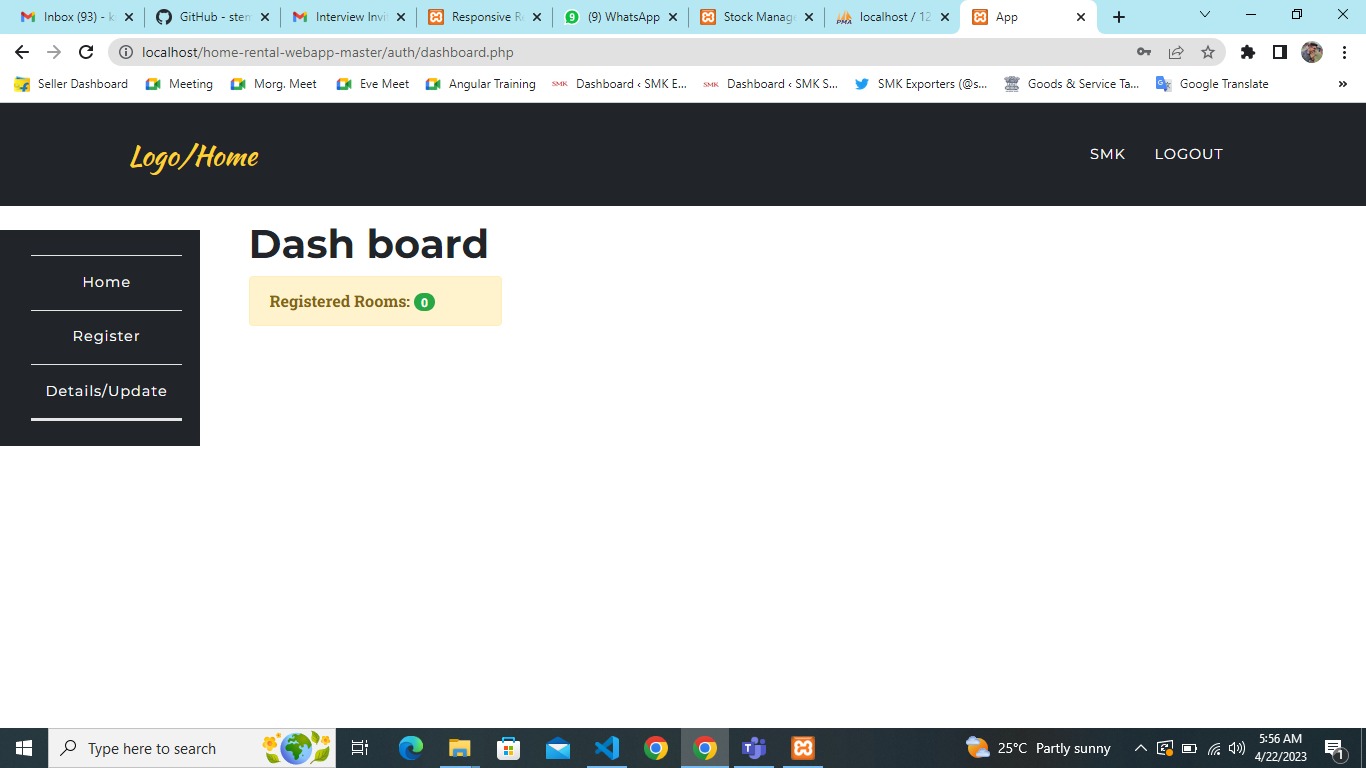
**Database Server**

**10.3 SCREEN LAYOUT**

The program has two operating modes, an embedding mode and an extraction mode.

**HOME PAGE**





**10.4 SOURCE CODE**

**Register.php**

<?php

require '../config/config.php';

if(empty($\_SESSION['username']))

header('Location: login.php');

if(isset($\_POST['register\_individuals'])) {

$errMsg = '';

// Get data from FROM

$fullname = $\_POST['fullname'];

$email = $\_POST['email'];

$mobile = $\_POST['mobile'];

$alternat\_mobile = $\_POST['alternat\_mobile'];

$plot\_number = $\_POST['plot\_number'];

$country = $\_POST['country'];

$state = $\_POST['state'];

$city = $\_POST['city'];

$address = $\_POST['address'];

$landmark = $\_POST['landmark'];

$rent = $\_POST['rent'];

$deposit = $\_POST['deposit'];

$description = $\_POST['description'];

//$open\_for\_sharing = $\_POST['open\_for\_sharing'];

$user\_id = $\_SESSION['id'];

$accommodation = $\_POST['accommodation'];

//$image = $\_POST['image']?$\_POST['image']:NULL;

//$other = $\_POST['other'];

$rooms = $\_POST['rooms'];

$vacant = $\_POST['vacant'];

$sale = $\_POST['sale'];

//upload an images

$target\_file = "";

if (isset($\_FILES["image"]["name"])) {

$target\_file = "uploads/".basename($\_FILES["image"]["name"]);

$uploadOk = 1;

$imageFileType = strtolower(pathinfo($target\_file,PATHINFO\_EXTENSION));

// Check if image file is a actual image or fake image

$check = getimagesize($\_FILES["image"]["tmp\_name"]);

if($check !== false) {

move\_uploaded\_file($\_FILES["image"]["tmp\_name"], "uploads/" . $\_FILES["image"]["name"]);

$uploadOk = 1;

} else {

echo "File is not an image.";

$uploadOk = 0;

}

}

//end of image upload

try {

$stmt = $connect->prepare('INSERT INTO room\_rental\_registrations (fullname, email, mobile, alternat\_mobile, plot\_number, rooms, country, state, city, address, landmark, rent, sale, deposit, description, image, accommodation, vacant, user\_id) VALUES (:fullname, :email, :mobile, :alternat\_mobile, :plot\_number, :rooms, :country, :state, :city, :address, :landmark, :rent, :sale, :deposit, :description, :image, :accommodation, :vacant, :user\_id)');

$stmt->execute(array(

':fullname' => $fullname,

':email' => $email,

':mobile' => $mobile,

':alternat\_mobile' => $alternat\_mobile,

':plot\_number' => $plot\_number,

//':ap\_number\_of\_plats' => $ap\_number\_of\_plats,

':rooms' => $rooms,

':country' => $country,

':state' => $state,

':city' => $city,

':address' => $address,

':landmark' => $landmark,

':rent' => $rent,

':sale' => $sale,

':deposit' => $deposit,

':description' => $description,

':accommodation' => $accommodation,

':image' => $target\_file,

//':other' => $other,

':vacant' => $vacant,

':user\_id' => $user\_id

));

header('Location: register.php?action=reg');

exit;

}

catch(PDOException $e) {

echo $e->getMessage();

}

}

if(isset($\_POST['register\_apartment'])) {

$errMsg = '';

// Get data from FROM

$fullname = $\_POST['fullname'];

$email = $\_POST['email'];

$mobile = $\_POST['mobile'];

$alternat\_mobile = $\_POST['alternat\_mobile'];

$plot\_number = $\_POST['plot\_number'];

$country = $\_POST['country'];

$state = $\_POST['state'];

$city = $\_POST['city'];

$address = $\_POST['address'];

$landmark = $\_POST['landmark'];

$rent = $\_POST['rent'];

$deposit = $\_POST['deposit'];

$description = $\_POST['description'];

//$open\_for\_sharing = $\_POST['open\_for\_sharing'];

$user\_id = $\_SESSION['id'];

$accommodation = $\_POST['accommodation'];

$apartment\_name = $\_POST['apartment\_name'];

$image = $\_FILES['image']['name'];

//$other = $\_POST['other'];

//upload an images

$target\_file = "";

if (isset($image)) {

# code...

$target\_file = "uploads/".basename($\_FILES["image"]["name"]);

$uploadOk = 1;

$imageFileType = strtolower(pathinfo($target\_file,PATHINFO\_EXTENSION));

// Check if image file is a actual image or fake image

//$check = getimagesize($\_FILES["image"]["tmp\_name"]);

//if($check !== false) {

move\_uploaded\_file($\_FILES["image"]["tmp\_name"], "uploads/" . $\_FILES["image"]["name"]);

$uploadOk = 1;

// } else {

// echo "File is not an image.";

//$uploadOk = 0;

// }

}

//end of image upload

try {

$stmt = $connect->prepare('INSERT INTO room\_rental\_registrations\_apartment (fullname, email, mobile, alternat\_mobile, plot\_number, apartment\_name, ap\_number\_of\_plats, rooms, floor, purpose, own, area, country, state, city, address, landmark, rent, deposit, description, image, accommodation, vacant, user\_id) VALUES (:fullname, :email, :mobile, :alternat\_mobile, :plot\_number, :apartment\_name, :ap\_number\_of\_plats, :rooms, :floor, :purpose, :own, :area, :country, :state, :city, :address, :landmark, :rent, :deposit, :description, :image, :accommodation, :vacant, :user\_id)');

foreach ($\_POST['ap\_number\_of\_plats'] as $key => $value) {

# code...

$stmt->execute(array(

':fullname' => $\_POST['fullname'][$key],

':email' => $email,

':mobile' => $mobile,

':alternat\_mobile' => $alternat\_mobile,

':plot\_number' => $plot\_number,

':apartment\_name' => $apartment\_name,

':ap\_number\_of\_plats' => $value,

':rooms' => $\_POST['rooms'][$key],

':floor' => $\_POST['floor'][$key],

':purpose' => $\_POST['purpose'][$key],

':own' => $\_POST['own'][$key],

':area' => $\_POST['area'][$key],

':country' => $country,

':state' => $state,

':city' => $city,

':address' => $\_POST['address'],

':landmark' => $\_POST['landmark'],

':rent' => $\_POST['rent'][$key],

':deposit' => $\_POST['deposit'][$key],

':description' => $\_POST['description'][$key],

':image' => $target\_file,

':accommodation' => $\_POST['accommodation'][$key],

//':other' => $\_POST['other'][$key],

':vacant' => $\_POST['vacant'][$key],

':user\_id' => $user\_id

));

}

header('Location: register.php?action=reg');

exit;

}catch(PDOException $e) {

echo $e->getMessage();

}

}

if(isset($\_GET['action']) && $\_GET['action'] == 'reg') {

$errMsg = 'Registration successfull. Thank you';

}

?>

**Compliant.php**

<?php include '../include/header.php';?>

<!-- Header nav -->

<nav class="navbar navbar-expand-lg navbar-dark" style="background-color:#212529;" id="mainNav">

<div class="container">

<a class="navbar-brand js-scroll-trigger" href="../index.php">Logo/Home</a>

<button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

Menu

<i class="fa fa-bars"></i>

</button>

<div class="collapse navbar-collapse" id="navbarResponsive">

<ul class="navbar-nav text-uppercase ml-auto">

<li class="nav-item">

<a class="nav-link" href="#"><?php echo $\_SESSION['fullname']; ?> <?php if($\_SESSION['role'] == 'admin'){ echo "(Admin)"; } ?></a>

</li>

<li class="nav-item">

<a href="../auth/logout.php" class="nav-link">Logout</a>

</li>

</ul>

</div>

</div>

</nav>

<!-- end header nav -->

<?php include '../include/side-nav.php';?>

<section class="wrapper" style="margin-left: 16%;margin-top: -11%;">

<!-- Nav tabs -->

<ul class="nav nav-tabs" role="tablist">

<li class="nav-item">

<a class="nav-link active" data-toggle="tab" href="#home" role="tab">Individual Home Registration</a>

</li>

<li class="nav-item">

<a class="nav-link" data-toggle="tab" href="#profile" role="tab">Apartment Registration</a>

</li>

</ul>

<div class="tab-content">

<!-- Single room -->

<div class="tab-pane active" id="home" role="tabpanel"><br>

<?php include 'partials/individaul.php';?>

</div>

<!-- Apartment -->

<div class="tab-pane" id="profile" role="tabpanel">

<?php include 'partials/apartment.php';?>

</div>

</div>

</section>

<?php include '../include/footer.php';?>

<script type="text/javascript">

var rowCount = 1;

function addMoreRows(frm) {

rowCount ++;

var recRow = '<div id="rowCount'+rowCount+'"><div class="row"><div class="col-md-4"><div class="form-group"><br> <a href="javascript:void(0);" onclick="removeRow('+rowCount+');" class="btn btn-danger btn-sm">Delete</a> </div> </div></div><div class="row"> <div class="col-md-4"><div class="form-group"> <label for="fullname">Full Name</label> <input type="fullname" class="form-control" id="fullname" placeholder="Full Name" name="fullname[]" required> </div> </div> <div class="col-md-4"><div class="form-group"> <label for="ap\_number\_of\_plats">Flat Number</label> <input type="ap\_number\_of\_plats" class="form-control" id="ap\_number\_of\_plats" placeholder="Flat Number" name="ap\_number\_of\_plats[]" required> </div> </div> <div class="col-md-4"> <div class="form-group"> <label for="rooms">Rooms</label> <input type="rooms" class="form-control" id="rooms" placeholder="2BHK/3BHK/1RK" name="rooms[]" required> </div> </div></div><div class="row"> <div class="col-md-4"> <div class="form-group"> <label for="area">Area</label> <input type="area" class="form-control" id="area" placeholder="Area" name="area[]"> </div> </div> <div class="col-md-4"> <div class="form-group"> <label for="purpose">Purpose</label> <select class="form-control" id="purpose" name="purpose[]"> <option value="Residential">Residential</option> <option value="Commercial">Commercial</option> </select> </div> </div> <div class="col-md-4"> <div class="form-group"> <label for="floor">Floor</label> <select class="form-control" id="floor" name="floor[]"> <option value="Ground Floor">Ground Floor</option> <option value="1st">1st</option> <option value="2nd">2nd</option> <option value="3rd">3rd</option> <option value="4th">4th</option> <option value="5th">5th</option> <option value="6th">6th</option> <option value="7th">7th</option> <option value="8th">8th</option> </select> </div> </div> </div> <div class="row"><div class="col-md-4"> <div class="form-group"> <label for="ownership">Owner/Rented</label> <select class="form-control" id="ownership" name="own[]"> <option value="owner">Owner</option> <option value="rented">Rented</option> </select> </div> </div> <div class="col-md-4"> <div class="form-group"> <label for="rent">Rent</label> <input type="rent" class="form-control" id="rent" placeholder="Rent" name="rent[]"> </div> </div> <div class="col-md-4"> <div class="form-group"> <label for="deposit">Deposit</label> <input type="deposit" class="form-control" id="deposit" placeholder="Deposit" name="deposit[]"> </div> </div> </div><div class="row"><div class="col-md-4"> <div class="form-group"> <label for="accommodation">Facilities</label> <input type="accommodation" class="form-control" id="accommodation" placeholder="Facilities" name="accommodation[]"> </div> </div> <div class="col-md-4"> <div class="form-group"> <label for="description">Description</label> <input type="description" class="form-control" id="description" placeholder="Description" name="description[]" required> </div> </div> <div class="col-4"> <div class="form-group"> <label for="vacant">Vacant/Occupied</label> <select class="form-control" id="vacant" name="vacant[]"> <option value="1">Vacant</option> <option value="0">Occupied</option> </select> </div> </div> </div></div>'; $('#addedRows').append(recRow);

}

function removeRow(removeNum) {

$('#rowCount'+removeNum).remove();

}

</script>

<?php

require '../config/config.php';

if(empty($\_SESSION['username']))

header('Location: login.php');

if(isset($\_POST['register'])) {

$name = $\_POST['name'];

$cmp = $\_POST['cmp'];

$username = $\_POST['user\_id'];

$fullname = $\_POST['fullname'];

try {

$stmt = $connect->prepare('INSERT INTO cmps (name,cmp,username,fullname) VALUES (:name, :cmp,:username,:fullname)');

$stmt->execute(array(

':name' => $name,

':cmp' => $cmp,

':username' => $username,

':fullname' => $fullname

));

$errMsg = 'Sent Successfully';

//header('Location: register.php?action=joined');

}catch(PDOException $e) {

$errMsg = $e->getMessage();

}

}

?>

<?php include '../include/header.php';?>

<!-- Header nav -->

<nav class="navbar navbar-expand-lg navbar-dark fixed-top" style="background-color:#212529;" id="mainNav">

<div class="container">

<a class="navbar-brand js-scroll-trigger" href="../index.php">Logo/Home</a>

<button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

Menu

<i class="fa fa-bars"></i>

</button>

<div class="collapse navbar-collapse" id="navbarResponsive">

<ul class="navbar-nav text-uppercase ml-auto">

<li class="nav-item">

<a class="nav-link" href="#"><?php echo $\_SESSION['fullname']; ?> <?php if($\_SESSION['role'] == 'admin'){ echo "(Admin)"; } ?></a>

</li>

<li class="nav-item">

<a href="../auth/logout.php" class="nav-link">Logout</a>

</li>

</ul>

</div>

</div>

</nav>

<!-- end header nav -->

<section style="padding-left:0px;">

<?php include '../include/side-nav.php';?>

</section>

<section class="wrapper" style="margin-left: 16%;margin-top: -23%;">

<div class="container">

<div class="row">

<div class="col-12">

<?php

if(isset($errMsg)){

echo '<div style="color:#FF0000;text-align:center;font-size:17px;">'.$errMsg.'</div>';

}

?>

<h2>Complaints</h2>

<form action="" method="post">

<div class="row">

<div class="col-6">

<div class="form-group">

<label for="name">Apartment No/Name Room No/Name</label>

<input type="text" class="form-control" id="name" placeholder="Full Name" name="name" required>

<input type="hidden" name="user\_id" value="<?php echo $\_SESSION['username']; ?>">

<input type="hidden" name="fullname" value="<?php echo $\_SESSION['fullname']; ?>">

</div>

</div>

<div class="col-6">

<div class="form-group">

<label for="cmp">Complaint</label>

<input type="text" class="form-control" id="cmp" placeholder="Text" name="cmp" required>

</div>

</div>

</div>

<button type="submit" class="btn btn-primary" name='register' value="register">Submit</button>

</form>

</div>

</div>

</div>

</section>

<?php include '../include/footer.php';?>