1.1 Institute Profile

Institute of Management and Career Courses (IMCC) is a premier Management Institute, established in 1983 by Maharashtra Education Society (MES) for providing quality education and technical expertise at the Post Graduation Level in the Fields of Computers and Management. The Institute is recognized by SPPU under Section 46 of Pune University Act, 1974 and Section 85 of Maharashtra University Act, 1994 and Approved by AICTE New Delhi to conduct MCA and MBA programs. The Institute is located at 131, Mayur Colony, Kothrud, Pune-411038 having 30,000sq. ft. built area & totally independent campus.

IMCC is recognized as a Ph.D. Research Center under the Faculty of Management, SPPU. IMCC has 38 years standing & it is well-known for its conducive educational atmosphere. IMCC focuses on the all-round development of its students. Thus, apart from excellence in academics, students develop their inner potential by way of active participation in cocurricular & extra-curricular activities.

Guest Lectures, Seminars, Workshops, Industrial Visits& Placements. The main motto of the Institute is to instill the concepts of total personality development in the students. The emphasis is laid on

‘Teacher Disciple Relationship’ in place of ‘Boss Subordinate’ relationship at their assignment.

The preamble of IMCC “FACTA-NON-VERBA" lucidly means that the Institute produces the new breed of professionals, who’s deeds will speak and there could be no requirement of pomposity. The conducive milieu of the Institute molds the budding managers to reveal in managing flexibility, integration, change and transformation. These ‘would be’ professionals are channelized in such a way to ‘orchestrate’ and deploy business and technological management skills in a synergistic manner to grab the tangible success. The faculty members put their relentless efforts in educating the students to synthesize business management acumen and technology insights in a creative manner.

1.2 Abstract

Pune Estate: Revolutionizing Real Estate Experience in Pune

"Pune Estate" is an innovative web-based platform aimed at transforming the real estate landscape in Pune, India. With the rapid growth and development in Pune, there arises a crucial need for a centralized digital solution that simplifies and enhances user experience, and fosters transparency in the real estate sector.

The platform serves as a comprehensive hub for property buyers, and brokers, offering a lot of features and functionalities to streamline the entire real estate process. Users can effortlessly search for properties based on their preferences, including location, budget, amenities, and property type, through an intuitive and user-friendly interface.

"Pune Estate" incorporates advanced search algorithms and filtering options to ensure that users find properties that perfectly match their requirements. Moreover, it provides detailed property listings with high-quality images, accurate descriptions, floor plans, and virtual tours, enabling users to make informed decisions without the need for physical visits.

Furthermore, the platform offers seamless communication channels, enabling direct interactions between buyers, and real estate agents.

Overall, "Pune Estate" revolutionizes the real estate experience in Pune by leveraging technology to streamline processes, enhance transparency, and empower users with valuable insights. Whether buying, selling, or renting properties, "Pune Estate" stands as the premier destination for all real estate endeavors in Pune.

1.3 Existing System and Need for System

The real estate market in Pune operated through traditional methods, which often presented various challenges and limitations:

1. Fragmented Information: Property listings were scattered across multiple platforms, including newspapers, classified ads, and individual real estate agency websites. This fragmentation made it difficult for users to access comprehensive and up-to-date information about available properties.
2. Lack of Transparency: The lack of standardized practices and transparency in the real estate sector led to mistrust and uncertainty among buyers and sellers. Hidden costs, inaccurate property descriptions, and unreliable agents were common issues faced by stakeholders.
3. Limited Accessibility: Physical visits to properties were often necessary for prospective buyers to assess their suitability, leading to time and resource-intensive processes. Moreover, individuals located outside Pune faced challenges in accessing relevant property information and conducting transactions remotely.

Need for a System:

The introduction of "Pune Estate" addresses the following needs and shortcomings of the existing real estate system:

1. Centralized Platform: "Pune Estate" provides a centralized platform where all property listings are aggregated, organized, and presented in a user-friendly manner. This consolidation of information enhances accessibility and convenience for users, allowing them to browse through a comprehensive database of properties from various developers and agents.

2. Transparency and Trust: By standardizing property listings, providing accurate descriptions, and implementing stringent quality controls, "Pune Estate" fosters transparency and trust within the real estate ecosystem. Users can rely on the platform to access reliable information, thereby minimizing the risk of fraud and misrepresentation.

3. Enhanced Accessibility: Through its online platform, "Pune Estate" ensures that property information is accessible to users across Pune and beyond, irrespective of their geographical location. This accessibility enables remote property search and facilitates transactions for individuals who may not be physically present in Pune.

Overall, "Pune Estate" addresses the limitations of the existing real estate system by leveraging technology to centralize information, enhance transparency, improve accessibility, and streamline communication. By fulfilling these needs, the platform revolutionizes the real estate experience in Pune, offering a more efficient, reliable, and user-centric approach to property transactions.

1.4 Scope of System

The scope of the "Pune Estate" system encompasses various aspects of the real estate ecosystem in Pune, India, with the aim of providing a comprehensive and user-centric platform for property transactions. The key components of the system include:

1. Property Listings:
    - Aggregation of residential and commercial property listings from various developers, agents, and property owners.
    - Detailed property information including photographs, descriptions, amenities, floor plans, and virtual tours.
2. Search and Filtering:
    - Advanced search functionality allowing users to filter properties based on location, amenities, and other preferences.
    - Customized search options for residential, commercial, and rental properties.

3. User Registration and Profiles:
    - User registration and account creation for buyers, and agents.
    - User profiles with personalized preferences, and favourite listings.
4. Mobile Compatibility:
    - Mobile-responsive design for seamless access and usability across various devices, including smartphones and tablets.
    - Mobile applications for iOS and Android platforms to enhance user convenience and accessibility.
5. Security and Privacy:
    - Implementation of robust security measures to protect user data, transactions, and sensitive information.
    - Compliance with data protection regulations and standards to ensure user privacy and confidentiality.

The scope of "Pune Estate" extends to provide a holistic solution for all stakeholders involved in the real estate market, offering a seamless and transparent experience throughout the property transaction journey.

1.5 Operating Environment - Hardware and Software

The operating system, hardware, and software requirements for accessing the "Pune Estate" system are as follows:

Operating Systems:

1. Windows 7 and above
2. Linux
3. Android OS

Hardware Requirements:

1. RAM: Above 2GB
2. Processor: Any modern processor capable of running the chosen operating system smoothly
3. Storage: Sufficient disk space to store browser cache and downloaded files

Software Requirements:

1. Internet Browser: Compatible with popular internet browsers such as Google Chrome, Microsoft Edge, Safari, etc.
2. Internet Connectivity: Stable internet connection with adequate bandwidth to support browsing, property searches, and data transmission.

1.6 Brief Description of Technology Used:

1.6.1 Operating systems used: The web-site project is developed and deployed on the Windows operating system.

1.6.2 Technology Used

The "Pune Estate" platform leverages a combination of frontend and backend technologies to deliver its functionality and user experience. Here's a brief description of the technologies used:

1. HTML (HyperText Markup Language):

   HTML forms the backbone of web pages, defining the structure and content of the platform's interface.

2. CSS (Cascading Style Sheets):

   CSS is used to style and design the HTML elements, ensuring consistency and aesthetic appeal across the platform. It controls aspects like layout, typography, colors, and responsive design for various screen sizes.

3. JavaScript (JS):

   JavaScript is employed for client-side scripting, enabling interactive features and dynamic content on the platform.

4. PHP (Hypertext Preprocessor):

   PHP serves as the backend scripting language, handling server-side logic and interactions with the database. PHP is responsible for processing user requests, executing business logic, and generating dynamic content based on user inputs and database queries.

MySQL is utilized as the relational database management system (RDBMS) to store and manage structured data for the platform. It handles tasks such as user authentication, property listings, user profiles, transaction records, and more.

2.1 Study of Similar Systems

Studying similar systems in the real estate sector can provide valuable insights into industry trends, user preferences, and technological advancements. Here's an overview of some relevant systems:

1. Zillow: Zillow is a popular real estate marketplace and database that provides users with information about homes, including value estimates, listings, and neighbourhood data.
2. Trulia: Trulia is another leading online real estate platform that offers listings, neighbourhood insights, and tools for homebuyers, sellers, and renters.
3. Realtor.com: Realtor.com is a comprehensive real estate website that provides listings, property details, market trends, and resources for homebuyers and sellers.
4. Redfin: Redfin is a real estate brokerage that offers online tools for searching homes, scheduling tours, and connecting with agents.

By studying these similar systems and related research papers, developers of "Pune Estate" can gain insights into best practices, user behaviours, and technological innovations in the real estate sector, thereby informing the design and implementation of their platform.

2.2 Feasibility Study:

A feasibility study for the "Pune Estate" real estate web project would assess the viability and potential success of the endeavor. Here's a breakdown of key aspects to consider in the feasibility study:

1. Market Analysis:

- Evaluate the current real estate market in Pune, including trends, demand-supply dynamics, and competition.

- Identify target demographics such as homebuyers, sellers, renters, agents, and developers.

- Analyze the market potential for a digital real estate platform in Pune, considering factors like population growth, urbanization, and economic indicators.

2. Technical Feasibility:

- Assess the availability and suitability of technology infrastructure required to develop and maintain the platform.

- Evaluate the feasibility of implementing the chosen technologies (HTML, CSS, PHP, JavaScript, MySQL) in terms of compatibility, scalability, and security.

- Consider factors like hosting options, server requirements, and integration with third-party services.

3. Financial Feasibility:

- Estimate the initial investment required for developing the "Pune Estate" platform, including software development, infrastructure setup, and marketing expenses.

4. Operational Feasibility:

- Evaluate the operational processes involved in running the "Pune Estate" platform, including property listings, user management and communication channels.

- Assess the feasibility of implementing efficient workflows, automation, and quality control measures to ensure smooth operations.

- Consider potential challenges such as user adoption, user support, and managing relationships with real estate agents and developers.

5. Risk Analysis:

- Identify potential risks and challenges that could impact the success of the project, such as technological risks, market volatility, competitive threats, and regulatory changes.

- Develop risk mitigation strategies to address identified risks and minimize their impact on the project's outcomes.

6. Conclusion:

- Summarize the findings of the feasibility study, including the overall feasibility of the project and any recommendations for proceeding.

- Provide a clear recommendation on whether to proceed with the development of the "Pune Estate" platform based on the analysis of market, technical, financial, legal, regulatory, and operational factors.

By conducting a thorough feasibility study, the project stakeholders can make informed decisions about the viability and potential success of the "Pune Estate" real estate web project.

2.3 Objectives of Proposed System:

The objectives of the proposed "Pune Estate" real estate web system are as follows:

1. Centralized Property Marketplace:

- Create a centralized online platform where buyers, sellers, agents, and developers can access comprehensive property listings in Pune.

2. Enhanced User Experience:

- Provide a user-friendly interface with intuitive navigation, advanced search options, and interactive features to enhance the overall user experience.

3. Transparency and Trust:

- Foster transparency and trust within the real estate ecosystem by providing accurate property information, standardized practices, and reliable user reviews.

4. Accessibility and Convenience:

- Improve accessibility and convenience for users by enabling remote property search, virtual property tours, and online transaction capabilities.

5. Data-driven Insights:

- Utilize data analytics and machine learning techniques to generate insights into market trends, property values, and investment opportunities.

6. Mobile Compatibility:

- Ensure compatibility with mobile devices, including smartphones and tablets, through responsive design and dedicated mobile applications for iOS and Android platforms.

7. Security and Privacy:

- Implement robust security measures to protect user data, transactions, and sensitive information, ensuring compliance with data protection regulations and standards.

By achieving these objectives, the "Pune Estate" platform aims to revolutionize the real estate experience in Pune, offering a more transparent, efficient, and user-centric approach to property transactions.

2.4 Users of System:

The users of the "Pune Estate" system primarily include two main categories:

1. Brokers:

- Real estate brokers or agents who are registered on the platform to list properties on behalf of property owners or developers.
- These brokers have access to features such as property listing management, communication tools, and transaction tracking.
- They utilize the platform to showcase properties to potential buyers or renters, facilitate property viewings, negotiate deals, and manage transactions.

2. Buyers Looking for Rent or Buying:

- Individuals or organizations seeking residential or commercial properties for rent or purchase in Pune.
- These buyers use the platform to search for properties based on their preferences, such as location, budget, size, amenities, etc.
- They interact with property listings, communicate with brokers or property owners, schedule property viewings and make inquiries.

These two user groups are central to the functioning of the "Pune Estate" system, representing the supply and demand sides of the real estate market in Pune. The platform serves as a bridge between brokers and buyers, facilitating property transactions and fostering transparency and efficiency in the real estate ecosystem.

3.1 System Requirements

(Functional and Non-Functional requirements)

Functional Requirements:

1. User Registration and Authentication:

- Users should be able to register for an account on the platform using email or social media accounts.

- Authentication mechanisms should be in place to verify user identity and ensure security.

2. Property Listings:

- Brokers should be able to create, edit, and manage property listings, including detailed descriptions, photographs, amenities, and pricing information.

- Buyers should be able to browse, search, and filter property listings based on various criteria such as location, budget, size, and amenities.

3. Property Viewing.

- Buyers should be able to view property directly through the platform.

- Brokers should be able to confirm, reschedule, or cancel property viewing appointments.

4. User Profiles and Preferences:

- User profiles with personalized settings, saved searches, favorite listings, and transaction history.

- Customized recommendations based on user preferences and browsing history.

5. Search Engine Optimization (SEO):

- Implementation of SEO best practices to ensure visibility and ranking in search engine results pages (SERPs).

Non-Functional Requirements:

1. Performance:

- Fast response times and minimal latency for loading property listings and executing user actions.

- Scalability to handle increasing user traffic and data volume over time.

2. Security:

- Robust security measures to protect user data, transactions, and sensitive information.

- Encryption of data transmissions and storage to prevent unauthorized access.

3. Reliability:

- High availability and uptime of the platform to ensure uninterrupted access for users.

- Backup and recovery mechanisms to prevent data loss in case of system failures or disasters.

4. Usability:

- Intuitive and user-friendly interface design to enhance user experience and usability.

- Accessibility features to ensure inclusivity for users with disabilities.

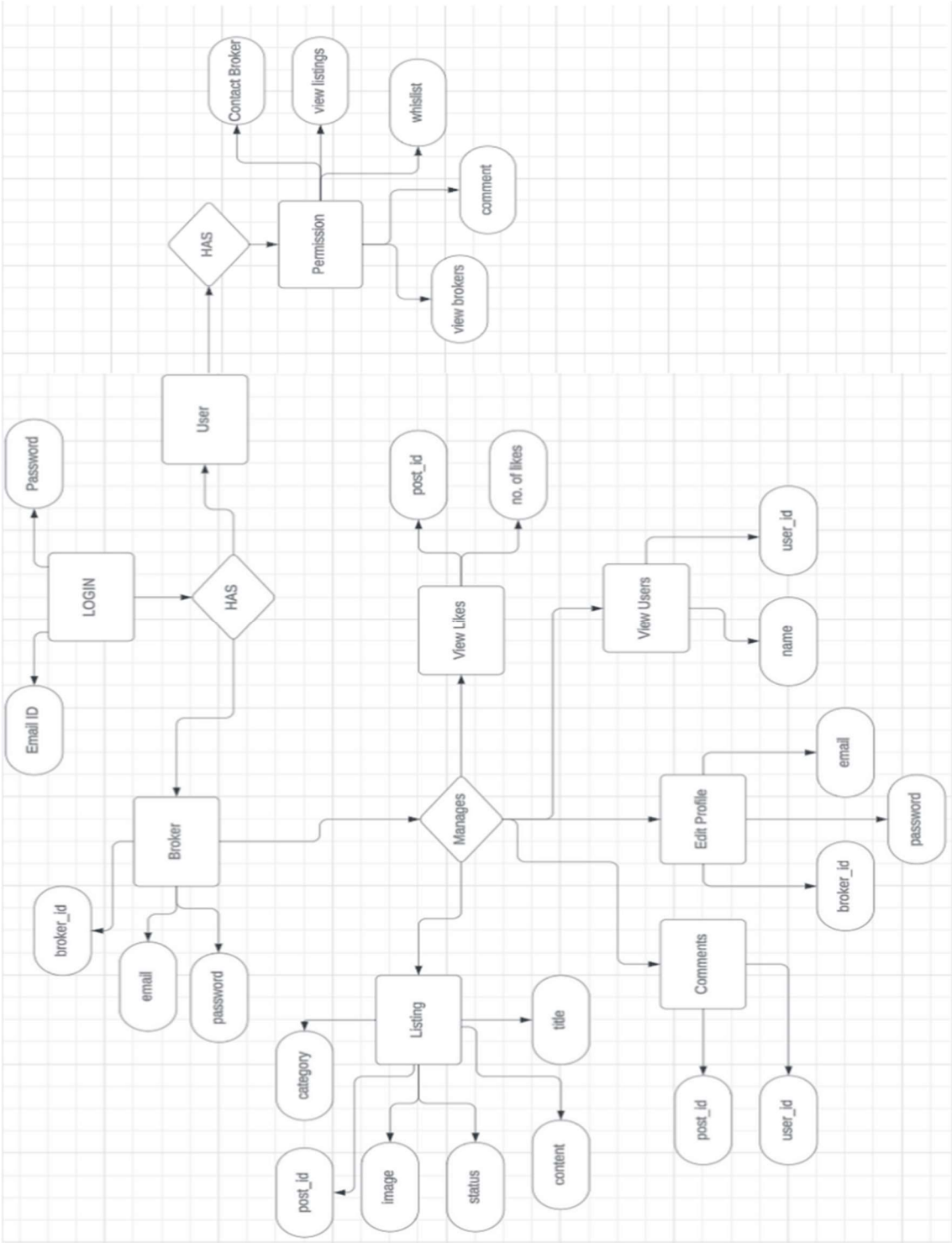
5. Compatibility:

- Compatibility with various web browsers (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, etc.) and operating systems (Windows, Linux, macOS, Android, iOS).

- Responsive design for seamless access and usability across different devices (desktops, laptops, smartphones, tablets).

By addressing these functional and non-functional requirements, the "Pune Estate" platform can deliver a reliable, secure, and user-friendly experience for brokers and buyers in the Pune real estate market.

3.2 Entity Relationship Diagram (ERD)



3.3 Table Structure

1. Broker:

Sr. No	Column	Data Type	Constraints	Description
1	brokerid	int (100)	primary key +incremental	every broker is given a id
2	email	varchar (20)		email used for contact
3	password	varchar (50)		password is encrypted

2. User:

Sr. No	Column	Data Type	Constraints	Description
1	userid	int (100)	primary key +incremental	every user is given id
2	name	varchar (20)		name of the user
3	email	varchar (50)		email used for contact
4	password	varchar (50)		password is encrypted

3. Post:

Sr. No	Column	Data Type	Constraints	Description
1	post id	int (100)	primary key +incremental	every post has an id
2	broker id	int (100)	foreign key	broker id
3	name	varchar (100)		name of the post
4	title	varchar (100)		title of the post
5	content	varchar (10000)		description of the post
6	category	varchar (50)		rent/sale
7	image	varchar (100)		image name is stored
8	date	date	current_ timestamp ()	date time is stored
9	status	varchar (10)		active/disable

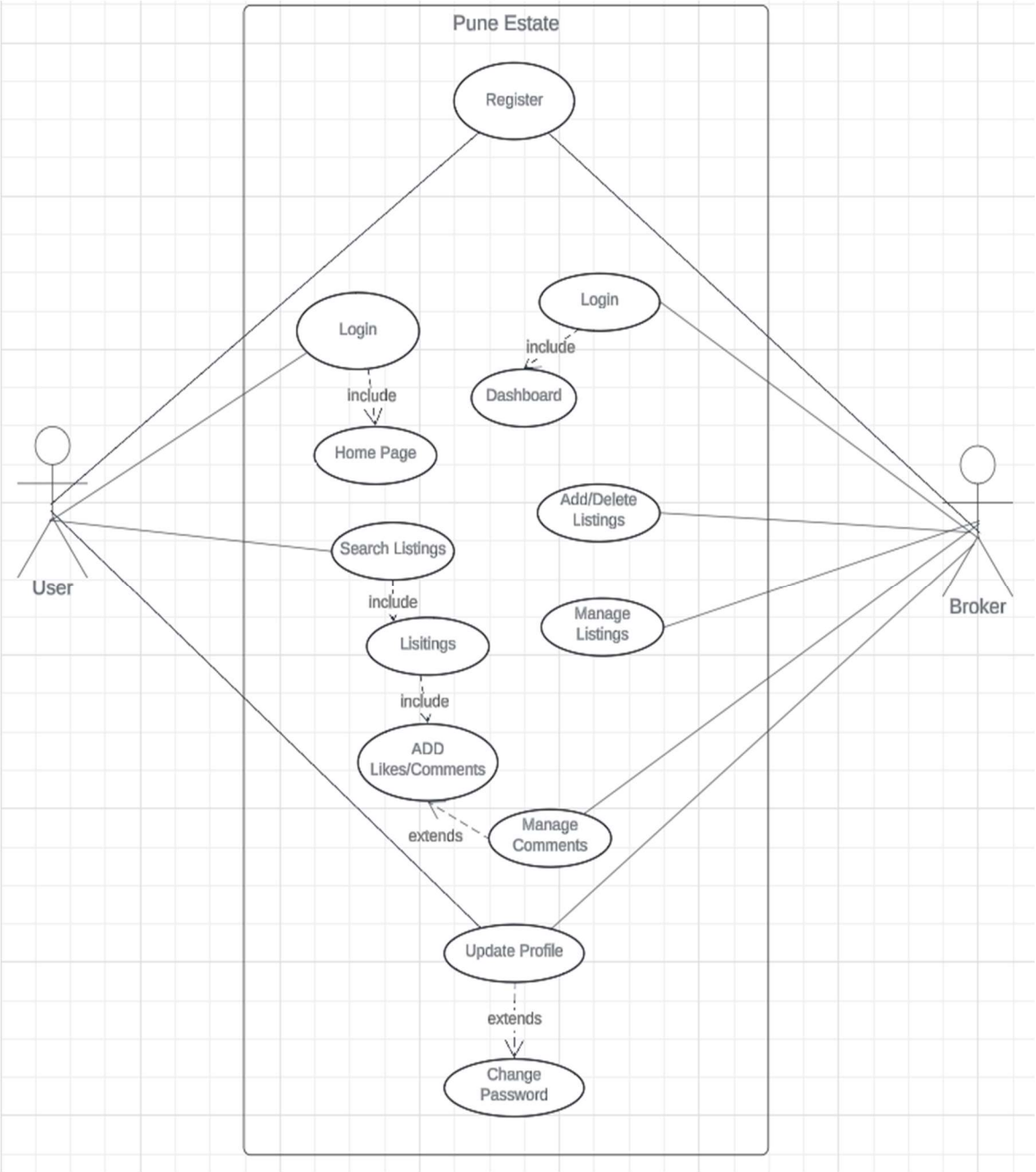
4. Likes:

Sr. No	Column	Data Type	Constraints	Description
1	id	int (100)	primary key + incremental	id to a like
2	user_id	int (100)	foreign key	id of user
3	broker_id	int (100)	foreign key	id of broker
4	post_id	int (100)	foreign key	id of liked post

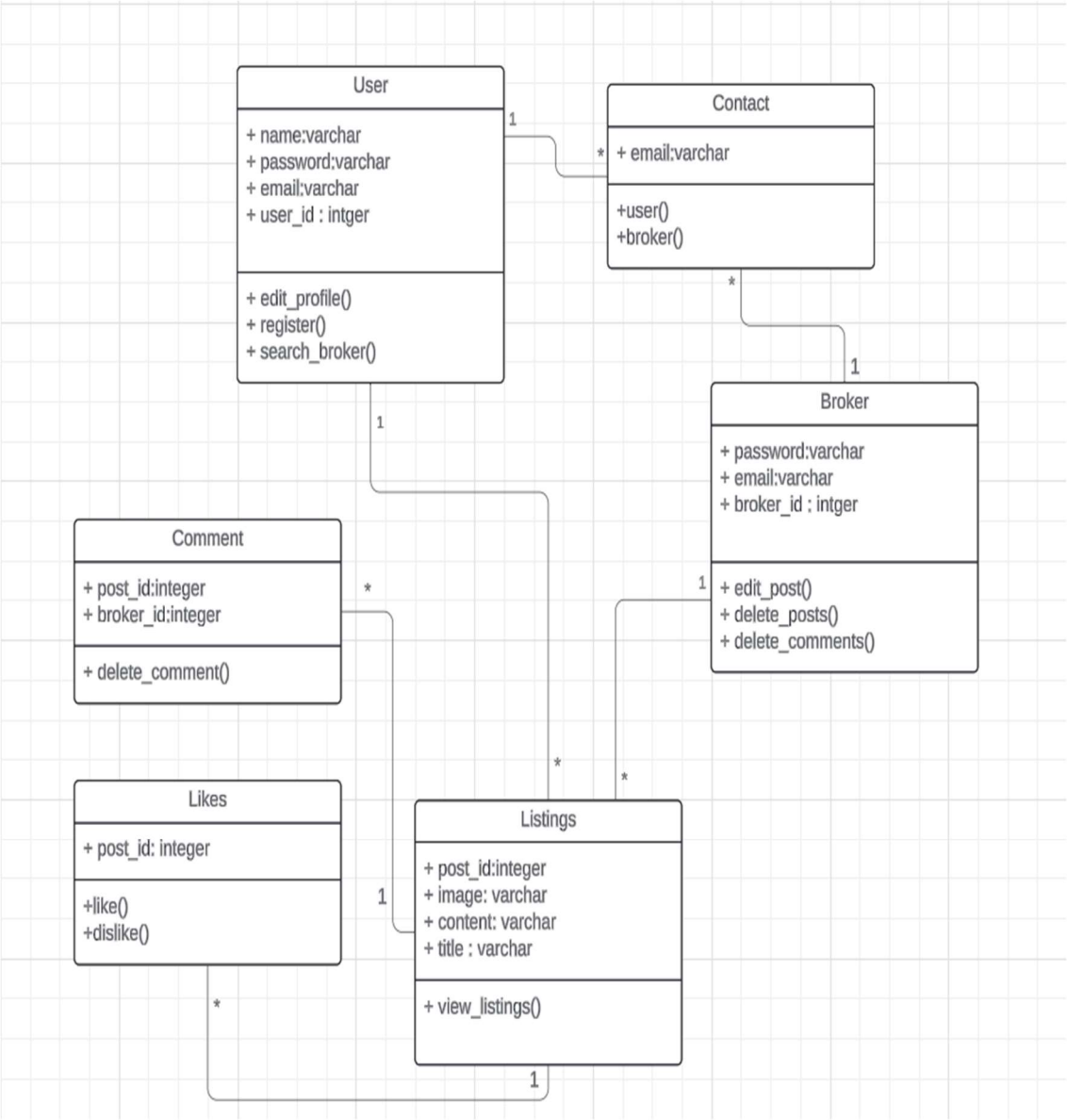
5. Comments:

Sr. No	Column	Data Type	Constraints	Description
1	id	int (100)	primary key +incremental	id of comments
2	post_id	int (100)	foreign key	commented post post_id
3	broker_id	int (100)	foreign key	broker's post
4	user_id	int (100)	foreign key	id of commenter
5	user_name	varchar (50)		name of commenter
6	comment	varchar (1000)		content
7	date	date	current_ timestamp ()	date and time of posting

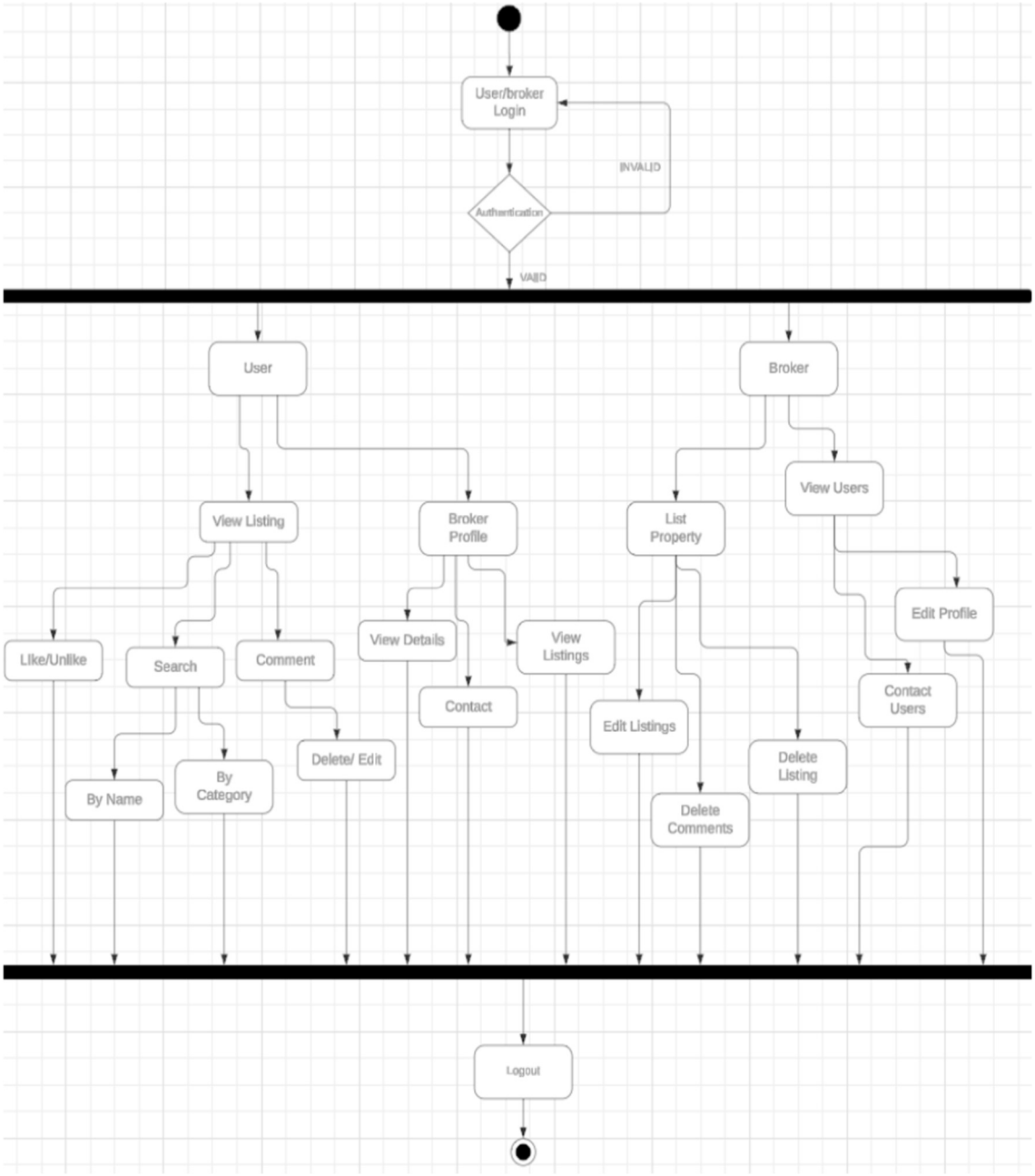
3.4 Use Case Diagram



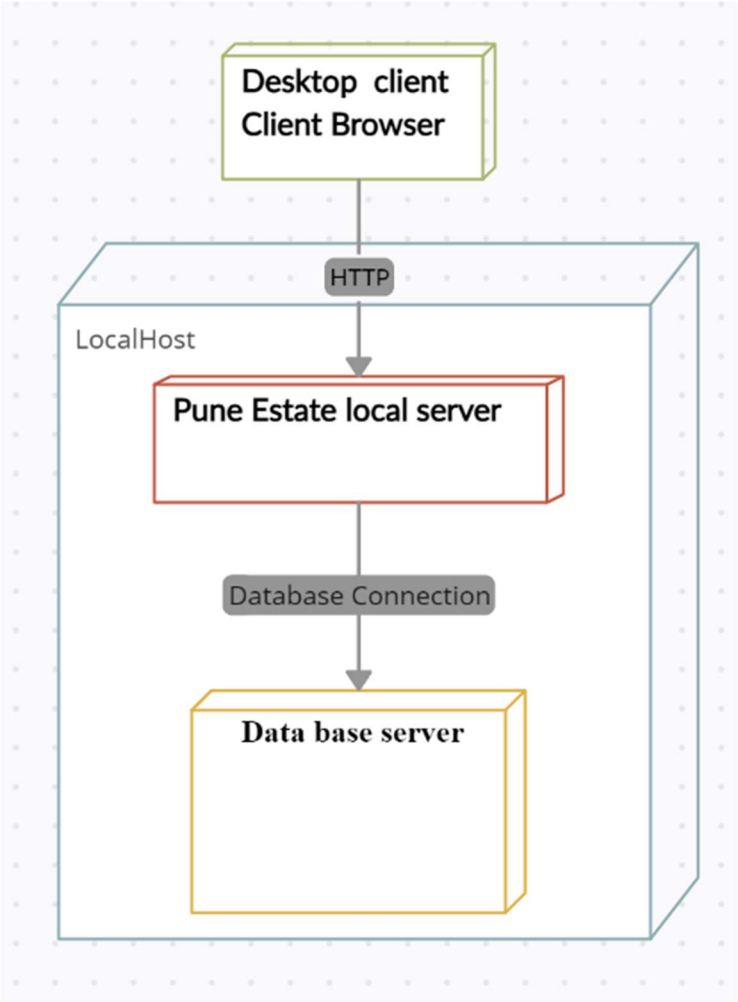
3.5 Class Diagram



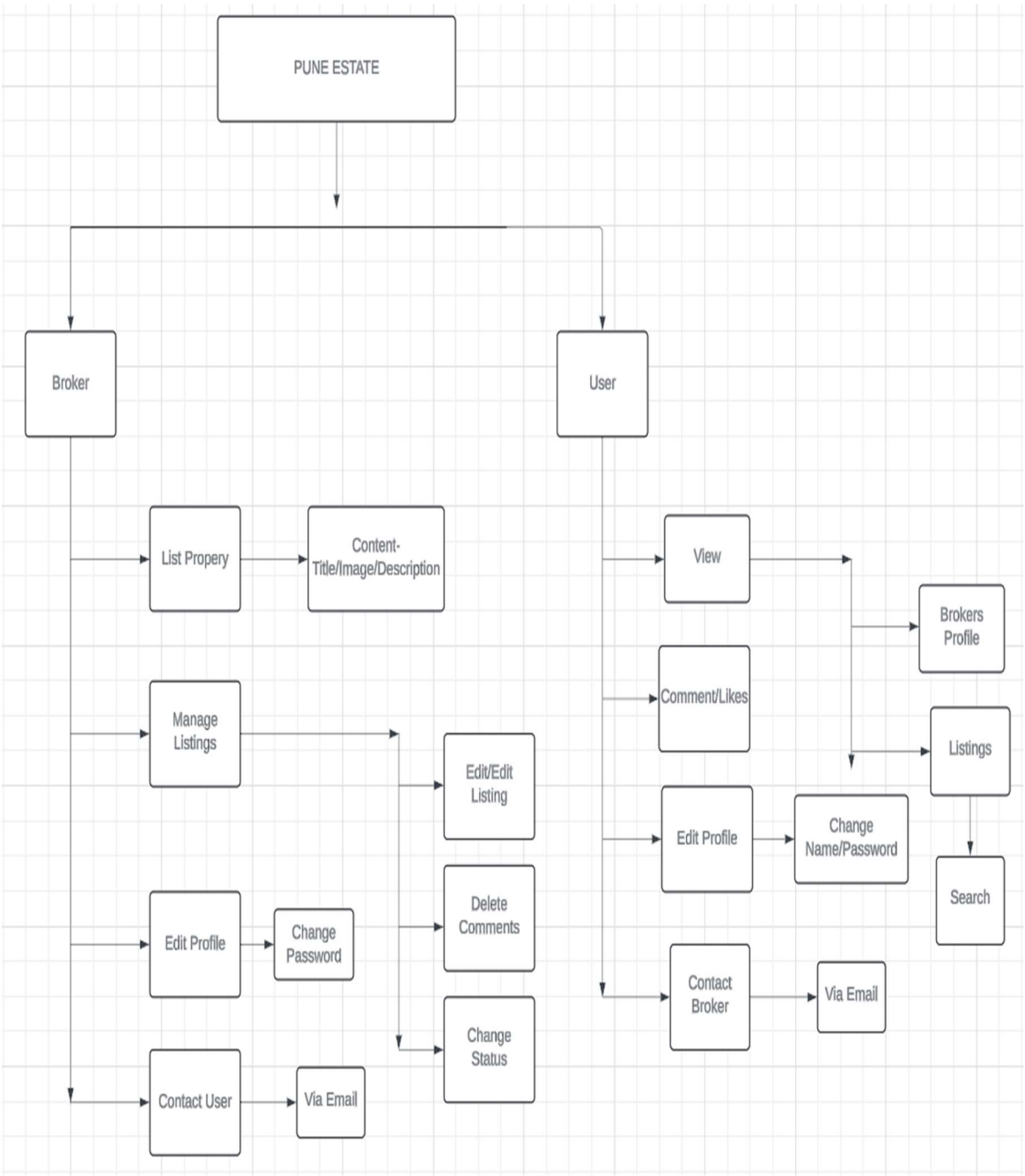
3.6 Activity Diagram



3.7 Deployment Diagram

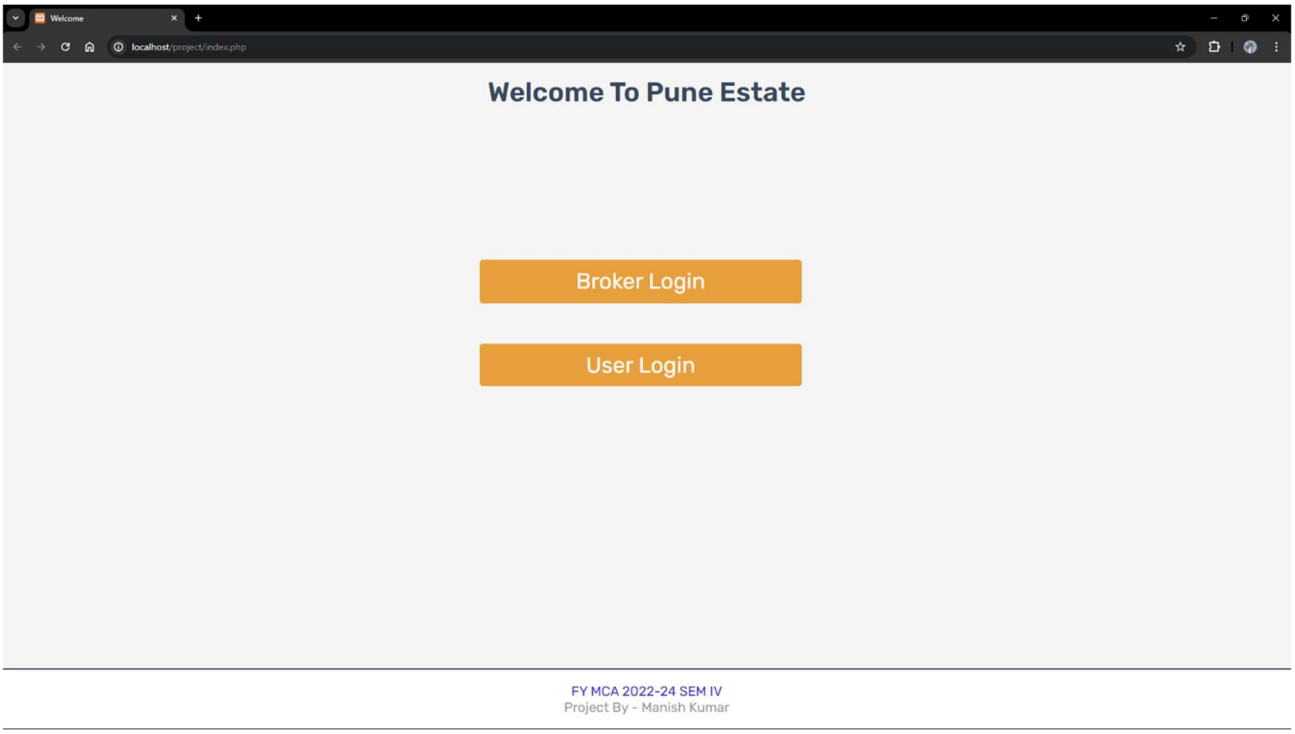


3.8 Module Hierarchy Diagram

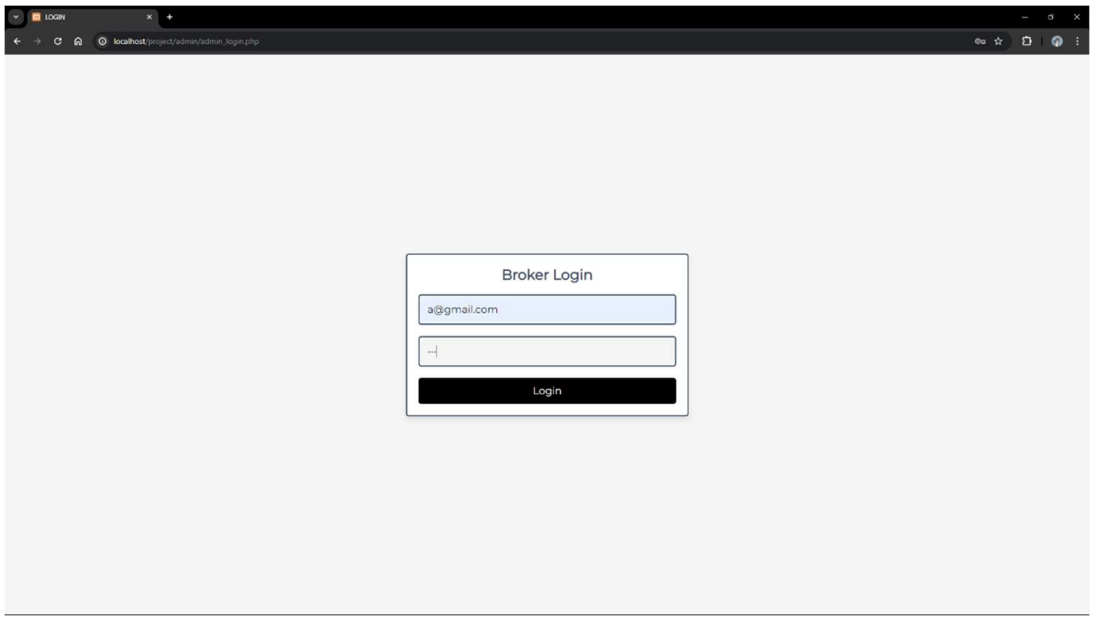


3.9 Sample Input and Output Screens

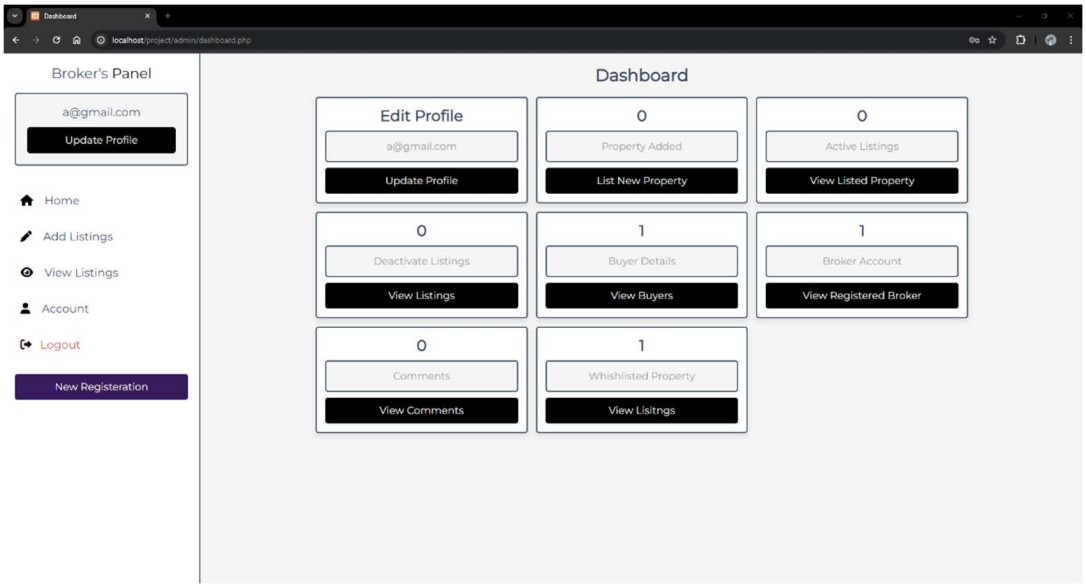
1. INDEX



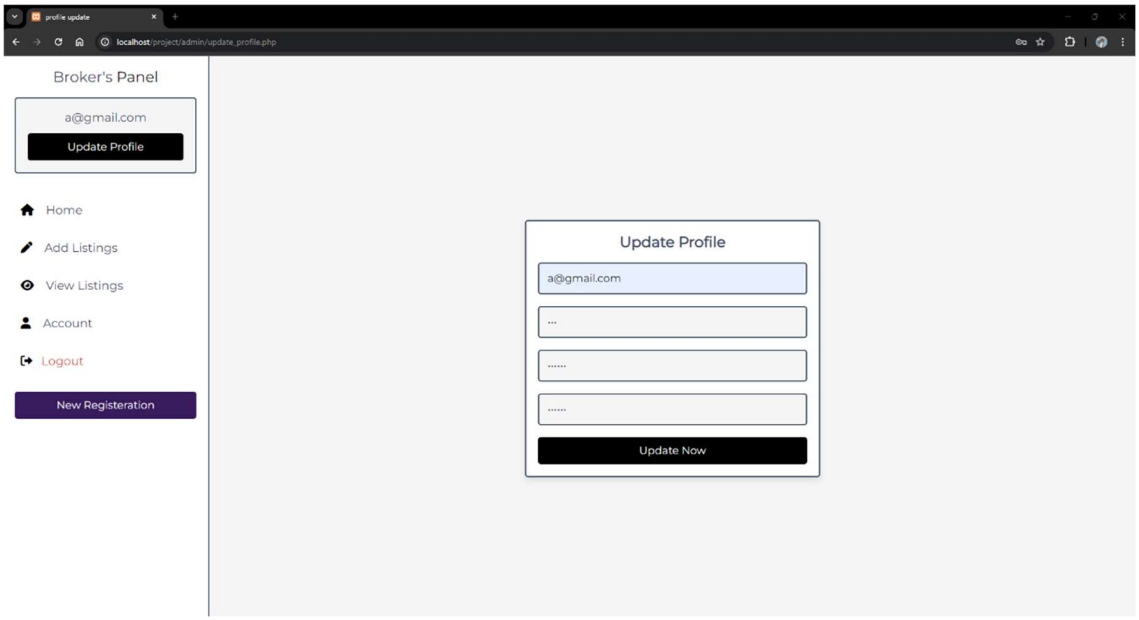
2. BROKER LOGIN



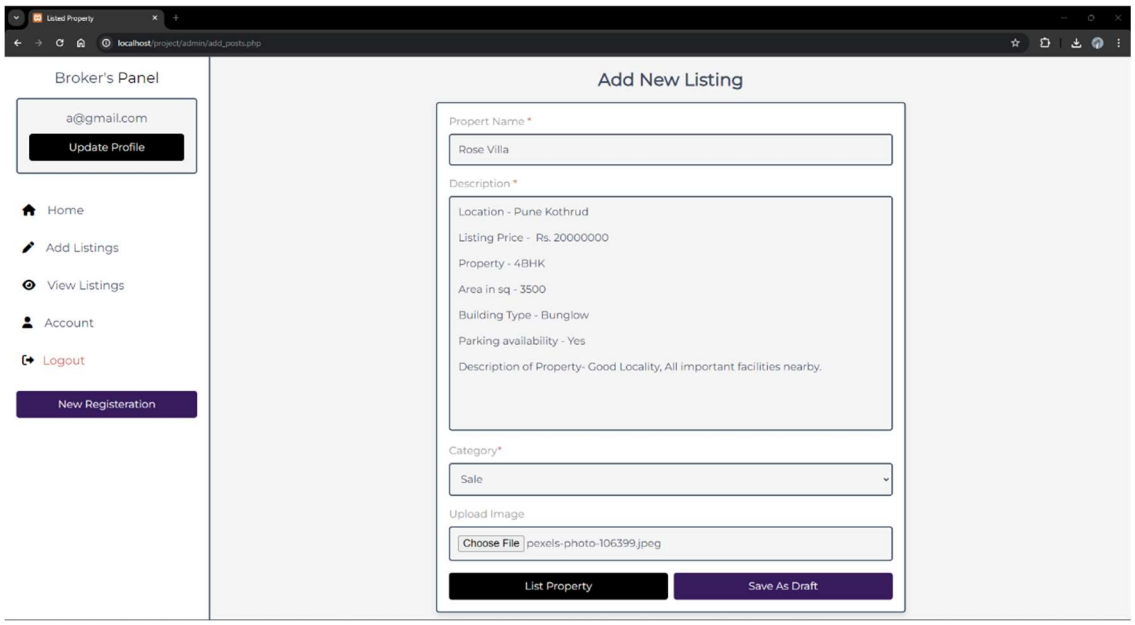
2.1 DASHBOARD



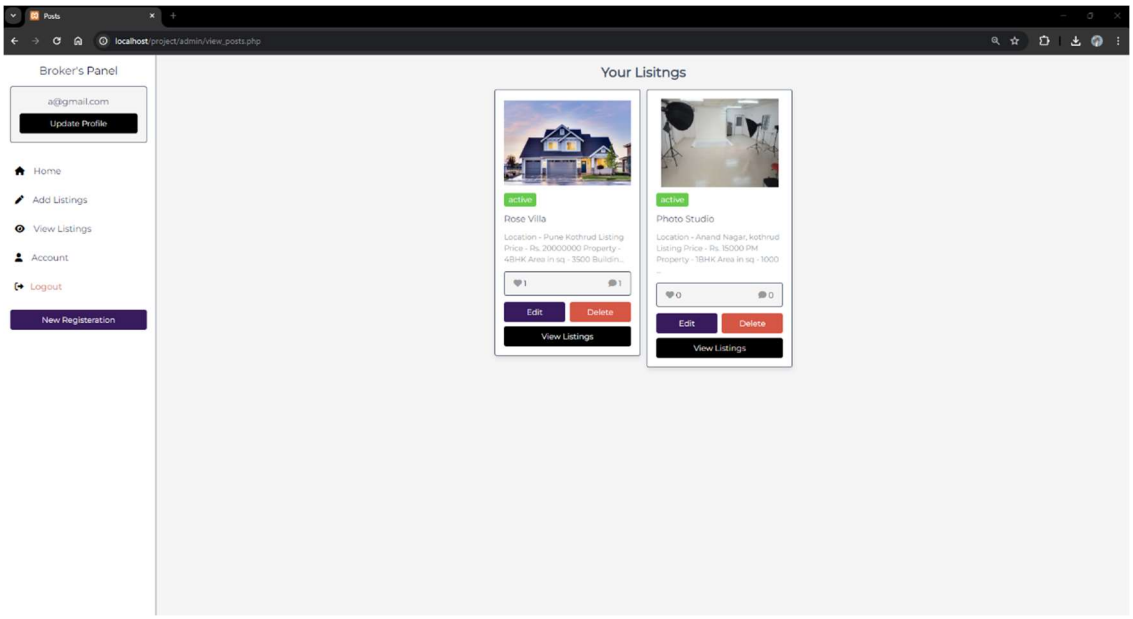
2.2 UPDATE PROFILE



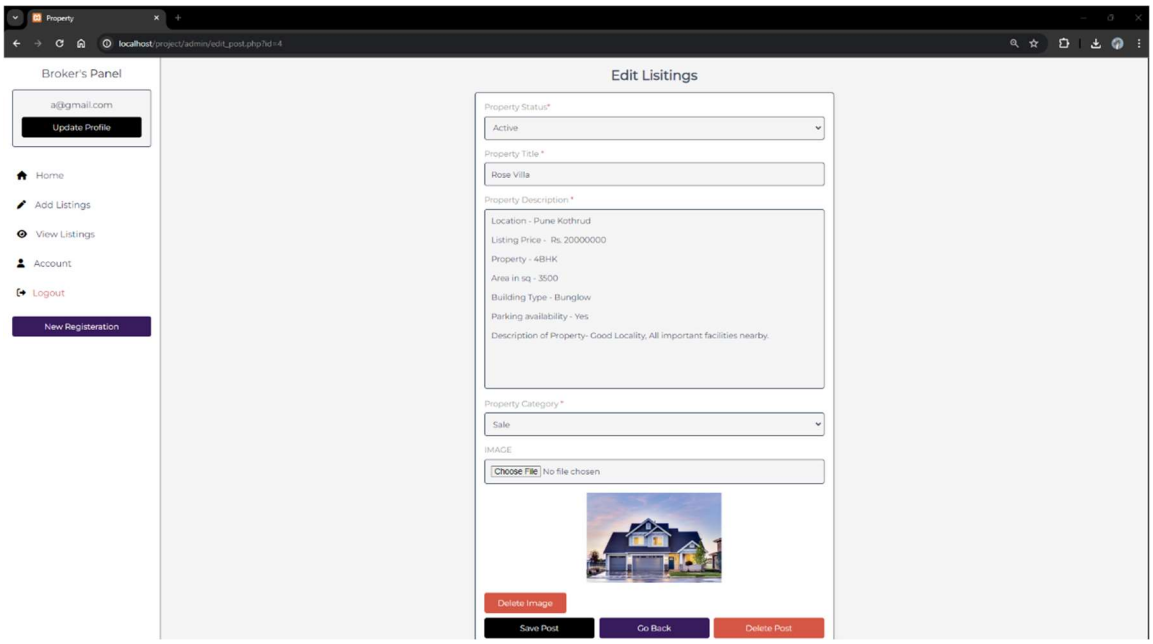
2.3 ADD NEW LISTINGS



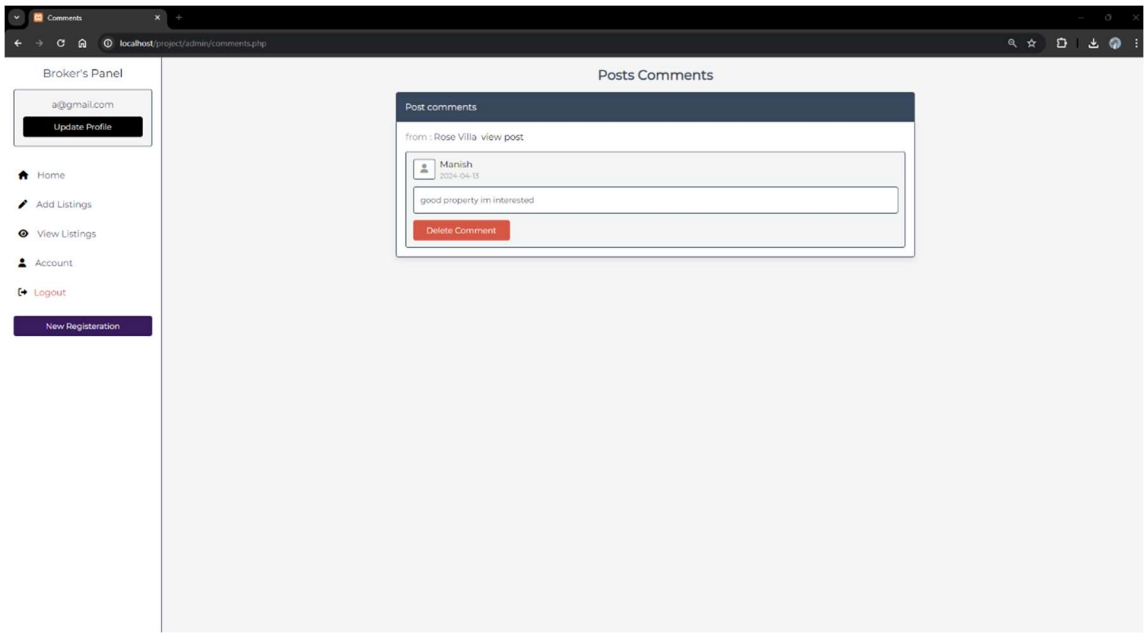
2.4 MANAGE LISTINGS



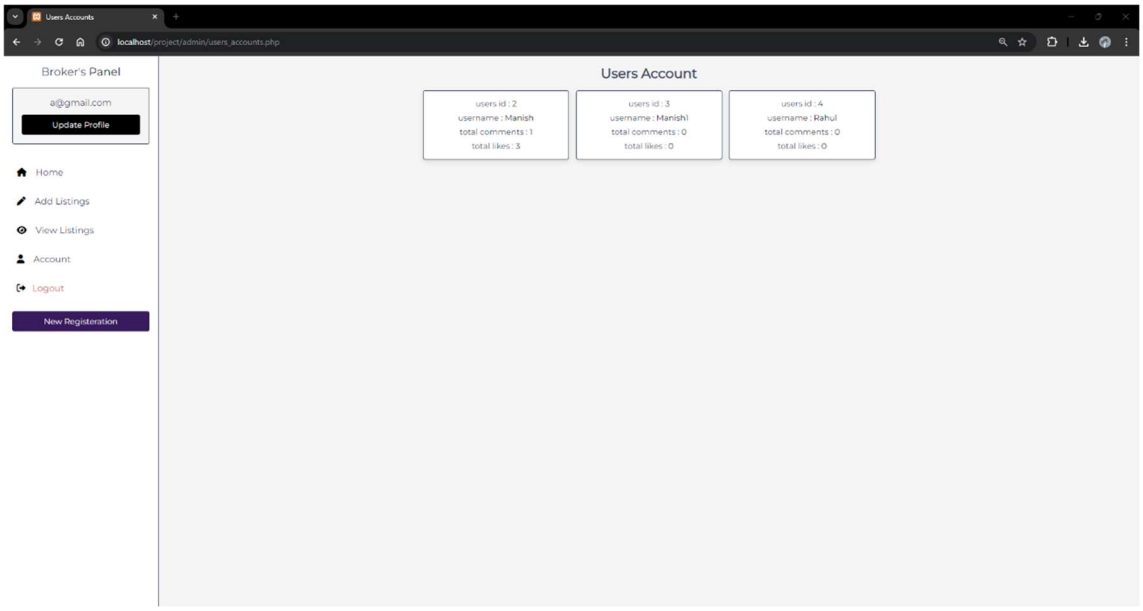
2.5 EDIT LISTINGS



2.6 VIEW USER ACCOUNT

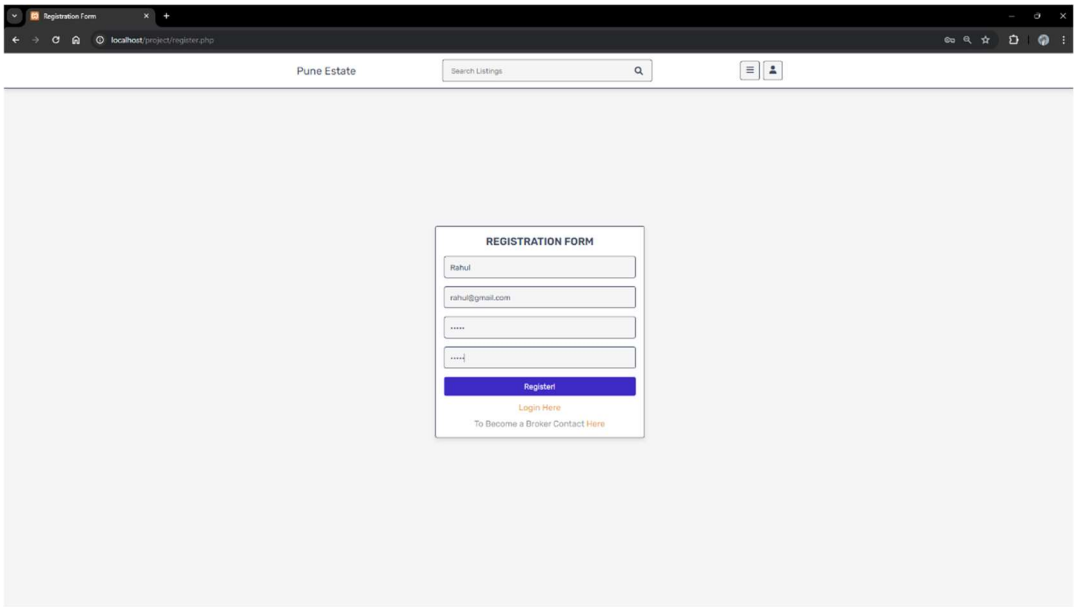


2.7 MANAGE COMMENTS

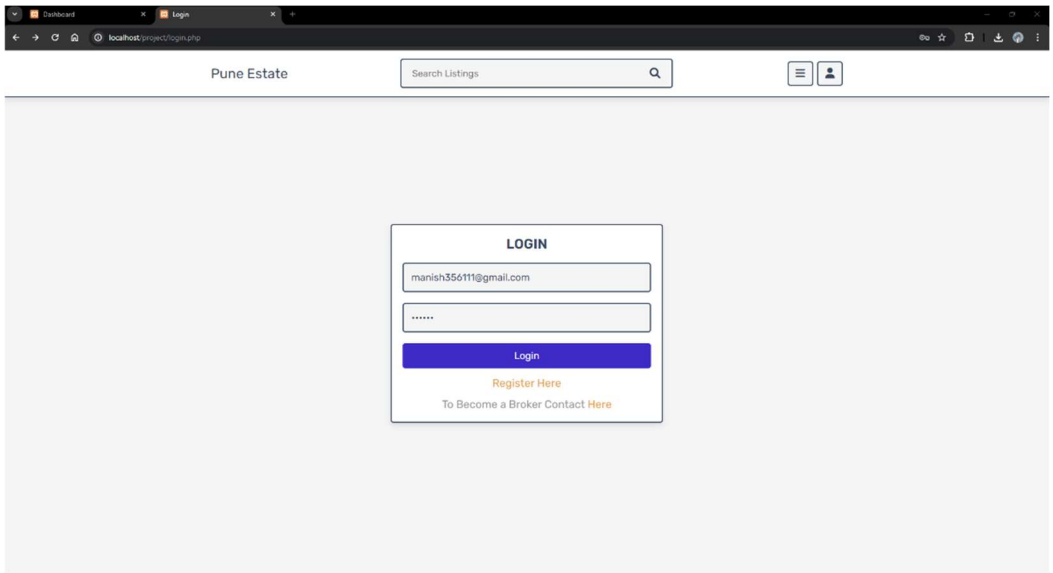


3. USER

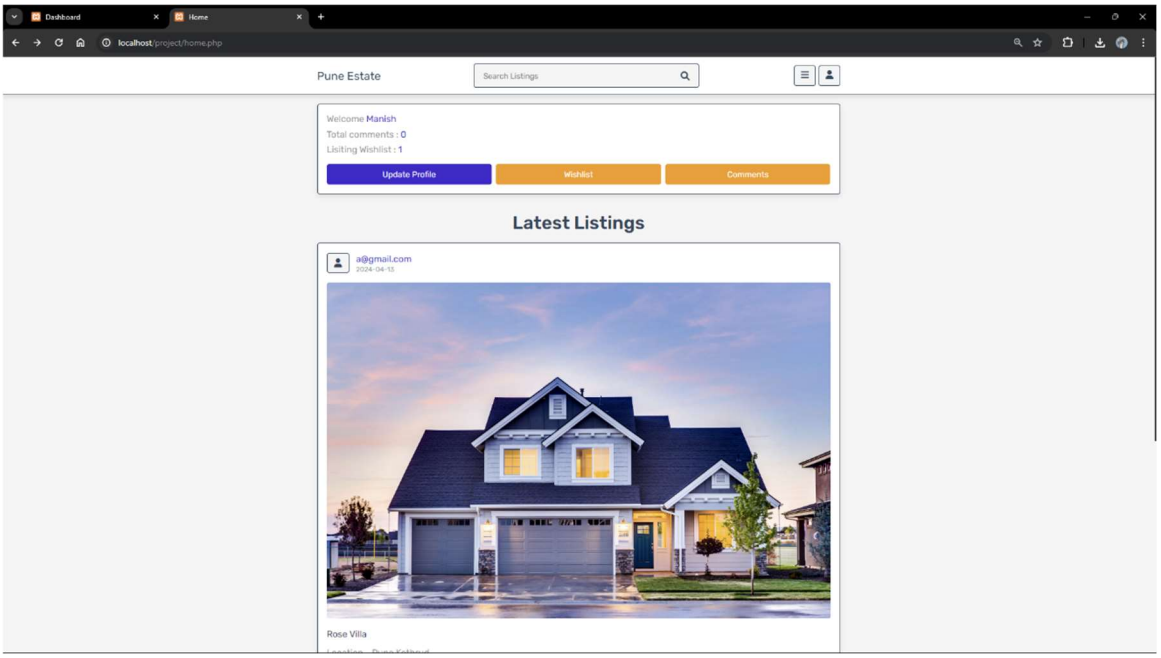
3.1 USER REGISTRATION



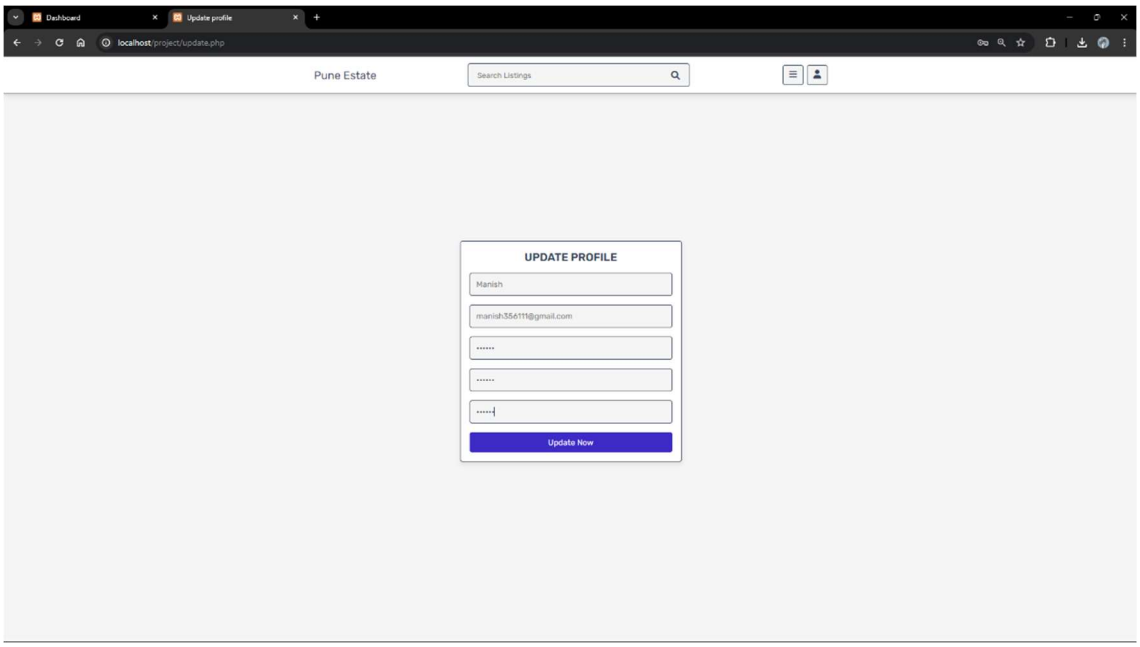
3.2 LOGIN



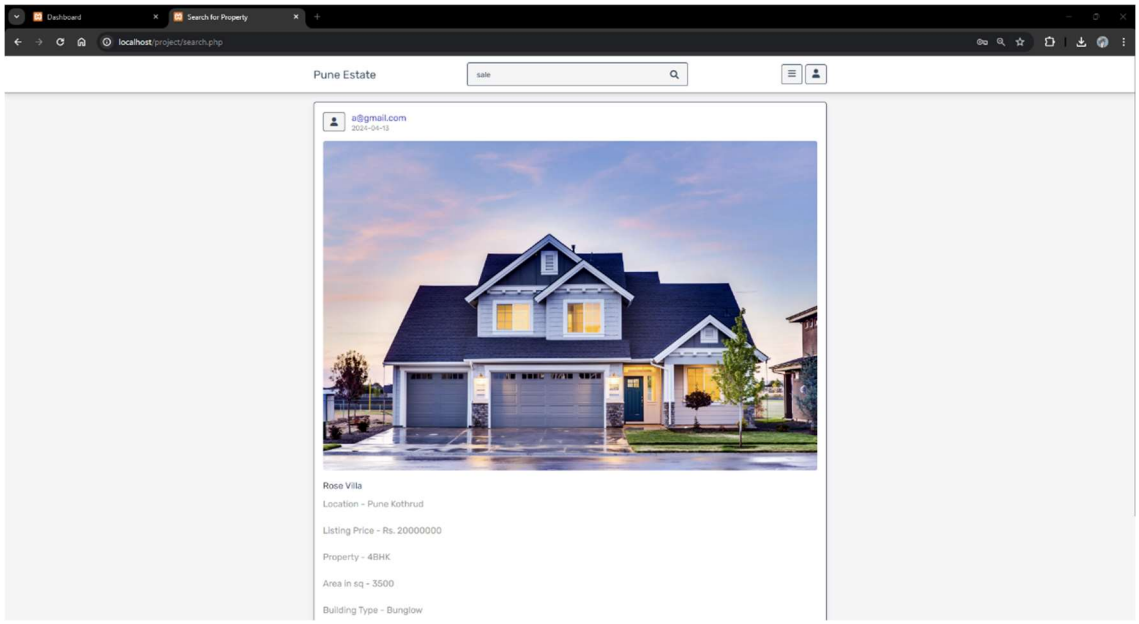
3.3 HOME PAGE/VIEWING LISTINGS



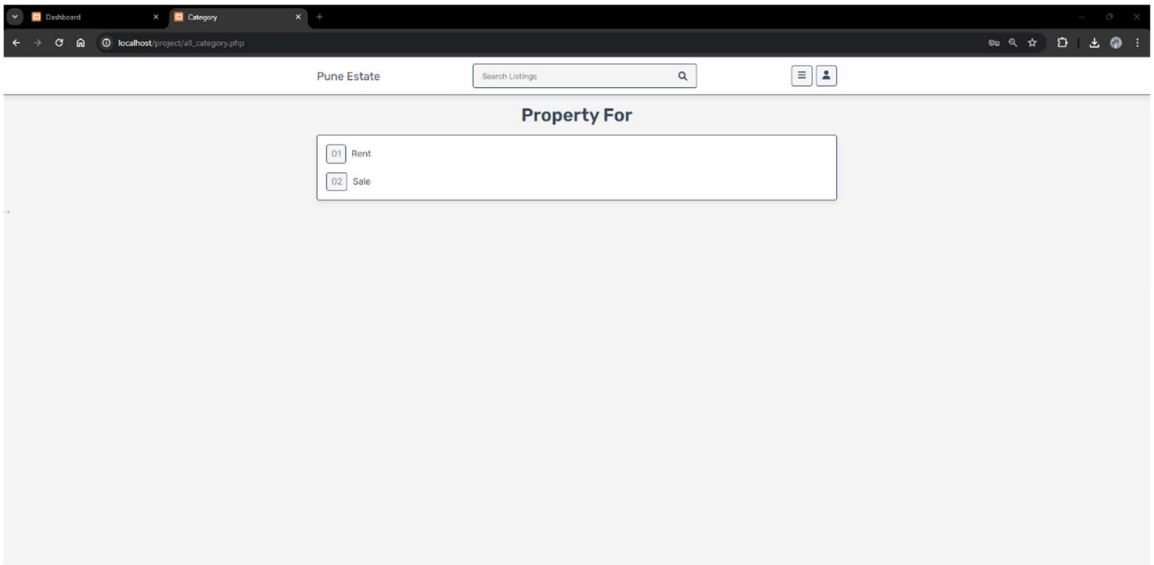
3.4 UPDATE PROFILE



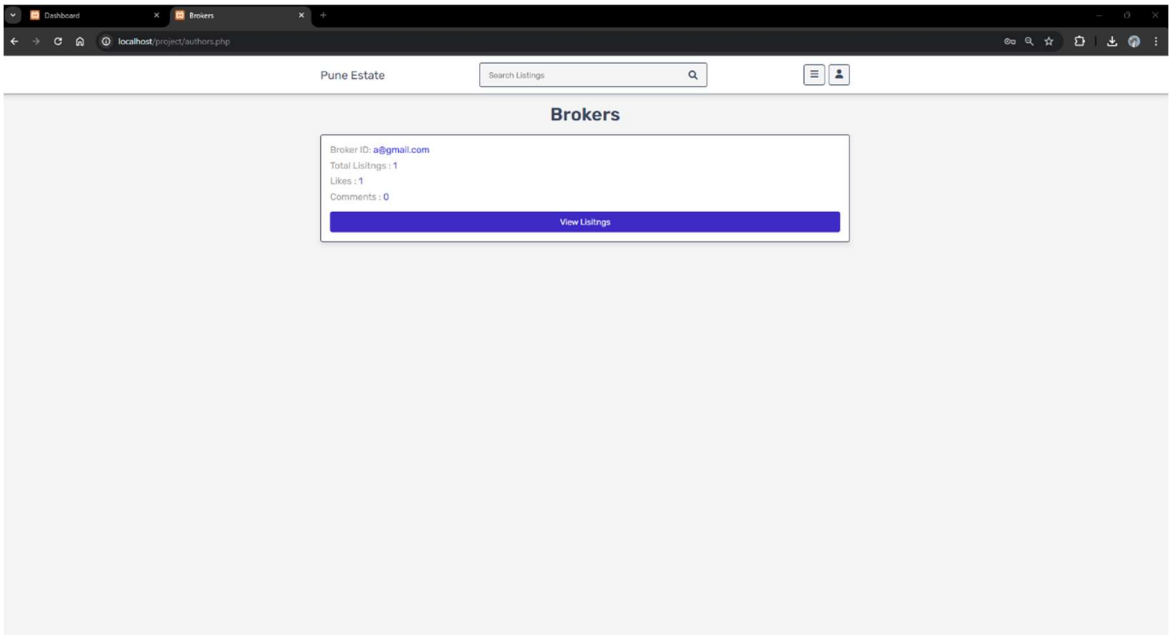
3.5 SEARCH LISTINGS



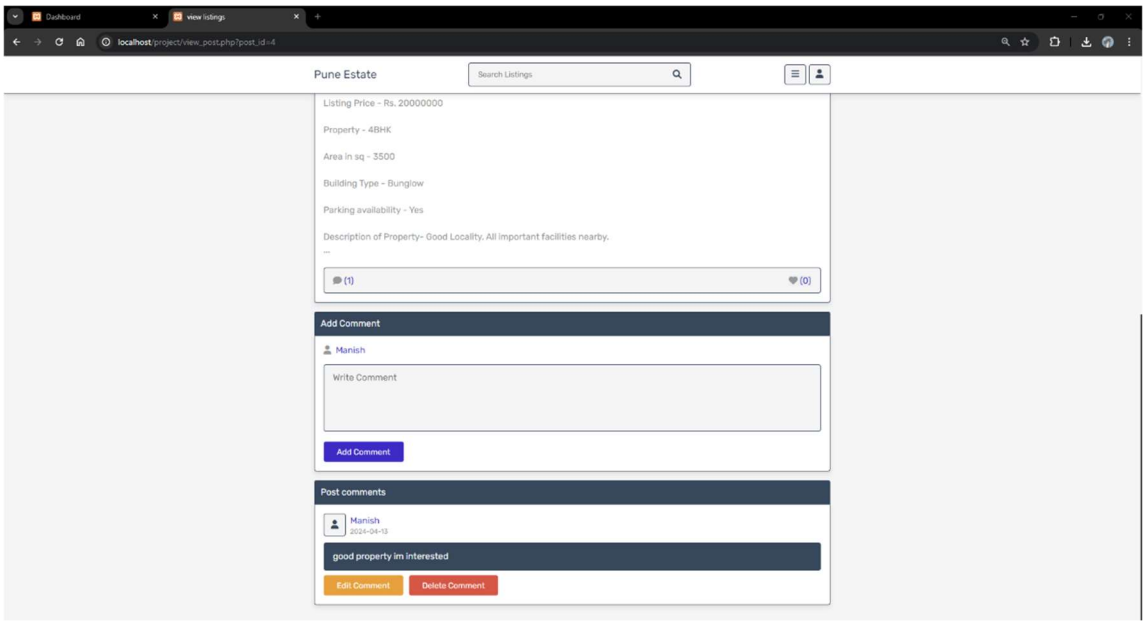
3.6 VIEW BY CATEGORIES



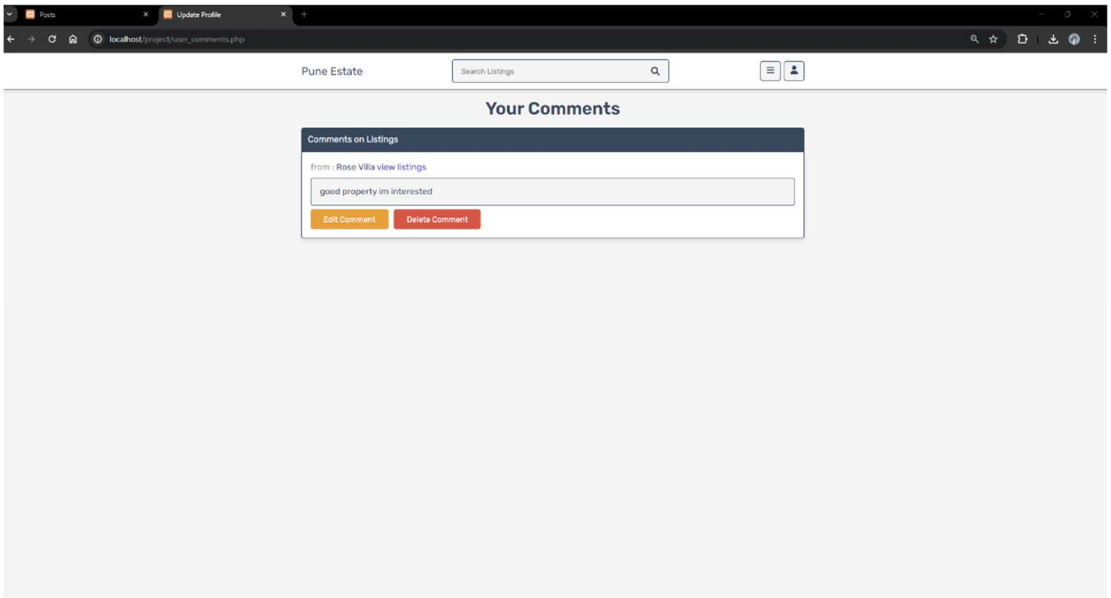
3.7 VIEW/CONTACT BROKER



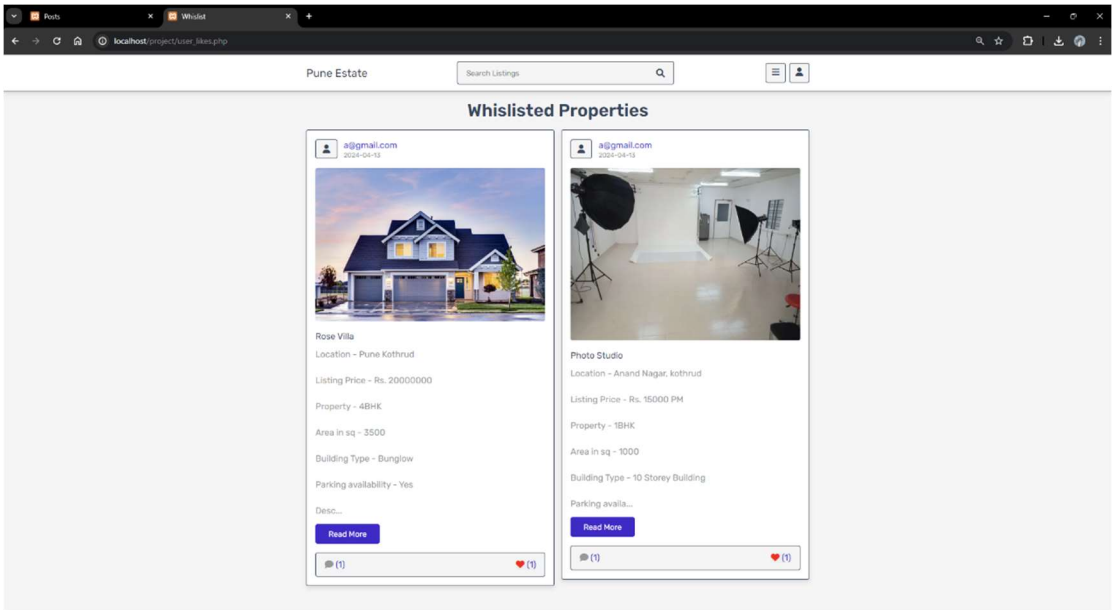
3.8 WHISHLIST/COMMENTS



3.9 EDIT/DELETE COMMENTS



3.10 WHISHLISTED LISTINGS



4.1 Algorithms

User Authentication and Authorization

1. When a user attempts to register:

1.1. Validate user input data:

- Ensure email format is valid.

- Check if password meets strength requirements.

1.2. Check if the email is not already registered:

- Query the database to see if the email exists.

1.3. Hash the password:

- Use a secure hashing algorithm to encrypt the password.

1.4. Store user data in the database:

- Insert user details (email, hashed password) into the user's table.

2. When a user attempts to log in:

2.1. Verify user credentials:

- Retrieve user data from the database based on the provided email.

- Compare the hashed password with the stored hash.

2.2. Generate and store a session token:

- If credentials are valid, create a unique session token.

- Store the session token in a session table or as a cookie.

3. Access control:

3.1. Restrict access based on user roles:

- Define user roles (e.g., seller, buyer).

- Associate permissions with each role.

3.2. Verify permissions:

- Check user role and permissions before allowing access to certain features.

Algorithm: Property Listings

1. Adding a new property listing:

1.1. Collect property details from the broker:

- Get input for location, price, amenities, etc.

1.2. Validate input data:

- Ensure required fields are filled.
- Validate numeric fields (e.g., price).

1.3. Save the property listing data in the database:

- Insert property details into the listings table.

2. Editing/deleting a property listing:

2.1. Allow authorized brokers to edit/delete their own listings:

- Check broker permissions before allowing modifications.

2.2. Validate broker permissions:

- Ensure broker is authorized to edit/delete the listing.

Algorithm: Broker

1. Broker login and access control:

1.1. Authenticate broker users:

- Verify broker credentials against stored data.

1.2. Restrict access to broker's features:

- Only allow access to brokers users.

2. Posting new property listings:

2.1. Provide interface to add/edit/delete listings:

- Display listings.

4.2 Code Snippets

index.php

```
<?php

include 'components/connect.php';

session_start();

if(isset($_SESSION['user_id'])){
  $user_id = $_SESSION['user_id'];
}else{
  $user_id = '';
};
include 'components/like_post.php';
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <title>Welcome</title>
```

```
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">

  <link rel="stylesheet" href="css/style.css">

</head>
<body>
<section class="posts-container">

  <h1 class="heading">Welcome To Pune Estate</h1>

  <div class="flex-btn">
        <a href="admin/admin_login.php" class="option-btn"
style="height:65px;width: 480px;margin-left: 331px;margin-top:
206px;""">
        <span style="font-size:33px;">Broker Login</span></a>
   </div>
   <div class="flex-btn">
        <a href="login.php" class="option-btn" style="margin-top:
60px;width: 480px;margin-left: 331px;""">
        <span style="font-size:33px;">User Login</span></a>
   </div>
</section>
```

```
<footer class="footer">
    <span>FY MCA 2022-24 SEM IV</span>
    <p>Project By - Manish Kumar</p>
</footer>
<?php include 'components/footer.php'; ?>
<script src="js/script.js"></script>


</body>
</html>
```

admin.php

```
<?php
include '../components/connect.php';
session_start();
if(isset($_POST['submit'])){
   $name = $_POST['name'];
   $name = filter_var($name, FILTER_SANITIZE_STRING);
   $pass = sha1($_POST['pass']);
   $pass = filter_var($pass, FILTER_SANITIZE_STRING);

   $select_admin = $conn->prepare("SELECT * FROM `admin`
WHERE name = ? AND password = ?");
   $select_admin->execute([$name, $pass]);
```

```
  if($select_admin->rowCount() > 0){
    $fetch_admin_id = $select_admin-
>fetch(PDO::FETCH_ASSOC);
    $_SESSION['admin_id'] = $fetch_admin_id['id'];
    header('location:dashboard.php');
  }else{
    $message[] = 'Enter Correct Details!';
  }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>LOGIN</title>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">
  <link rel="stylesheet" href="../css/admin_style.css">
```

```
</head>
<body style="padding-left: 0 !important;">

<?php
if(isset($message)){
  foreach($message as $message){
    echo '
    <div class="message">
      <span>'.$message.'</span>
      <i class="fas fa-times"
onclick="this.parentElement.remove();"></i>
    </div>
    ';
  }
}
?>
<section class="form-container">

  <form action="" method="POST">
    <h3>Broker Login</h3>

    <input type="text" name="name" maxlength="30" required
placeholder="User ID" class="box" oninput="this.value =
this.value.replace(/\s/g, '')">
```

```
    <input type="password" name="pass" maxlength="20" required
placeholder="Password" class="box" oninput="this.value =
this.value.replace(/\s/g, '')">
    <input type="submit" value="Login" name="submit"
class="btn">
  </form>
</section>

</body>
</html>
```

Admin_dashboard.php

```
<?php

include '../components/connect.php';

session_start();

$admin_id = $_SESSION['admin_id'];

if(!isset($admin_id)){
  header('location:admin_login.php');
```

```
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Dashboard</title>

  <!-- font awesome cdn link  -->
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">

  <!-- custom css file link  -->
  <link rel="stylesheet" href="../css/admin_style.css">

</head>
<body>
```

```
<?php include '../components/admin_header.php' ?>

<!-- admin dashboard section starts  -->

<section class="dashboard">

  <h1 class="heading">Dashboard</h1>

  <div class="box-container">

  <div class="box">
    <h3>Edit Profile</h3>
    <p><?= $fetch_profile['name']; ?></p>
    <a href="update_profile.php" class="btn">Update profile</a>
  </div>

  <div class="box">
    <?php
      $select_posts = $conn->prepare("SELECT * FROM `posts`
WHERE admin_id = ?");
      $select_posts->execute([$admin_id]);
      $numbers_of_posts = $select_posts->rowCount();
    ?>
    <h3><?= $numbers_of_posts; ?></h3>
```

```
    <p>Property Added</p>
    <a href="add_posts.php" class="btn">List New Property</a>
  </div>

  <div class="box">
    <?php
      $select_active_posts = $conn->prepare("SELECT * FROM
`posts` WHERE admin_id = ? AND status = ?");
      $select_active_posts->execute([$admin_id, 'active']);
      $numbers_of_active_posts = $select_active_posts-
>rowCount();
    ?>
    <h3><?= $numbers_of_active_posts; ?></h3>
    <p>Active Listings</p>
    <a href="view_posts.php" class="btn">View Listed
Property</a>
  </div>

  <div class="box">
    <?php
      $select_deactive_posts = $conn->prepare("SELECT * FROM
`posts` WHERE admin_id = ? AND status = ?");
      $select_deactive_posts->execute([$admin_id, 'deactive']);
```

```
        $numbers_of_deactive_posts = $select_deactive_posts-
>rowCount();
    ?>
    <h3><?= $numbers_of_deactive_posts; ?></h3>
    <p>Deactivate Listings</p>
    <a href="view_posts.php" class="btn">View Listings</a>
  </div>
  <div class="box">
    <?php
        $select_users = $conn->prepare("SELECT * FROM `users`");
        $select_users->execute();
        $numbers_of_users = $select_users->rowCount();
    ?>
    <h3><?= $numbers_of_users; ?></h3>
    <p>Buyer Details</p>
    <a href="users_accounts.php" class="btn">View Buyers</a>
  </div>
  <div class="box">
    <?php
        $select_admins = $conn->prepare("SELECT * FROM
`admin`");
        $select_admins->execute();
        $numbers_of_admins = $select_admins->rowCount();
    ?>
```

```
    <h3><?= $numbers_of_admins; ?></h3>
    <p>Broker Account</p>
    <a href="admin_accounts.php" class="btn">View Registered
Broker</a>
  </div>
  <div class="box">
    <?php
      $select_comments = $conn->prepare("SELECT * FROM
`comments` WHERE admin_id = ?");
      $select_comments->execute([$admin_id]);
      $select_comments->execute();
      $numbers_of_comments = $select_comments->rowCount();
    ?>
    <h3><?= $numbers_of_comments; ?></h3>
    <p>Comments</p>
    <a href="comments.php" class="btn">View comments</a>
  </div>
  <div class="box">
    <?php
      $select_likes = $conn->prepare("SELECT * FROM `likes`
WHERE admin_id = ?");
      $select_likes->execute([$admin_id]);
      $select_likes->execute();
      $numbers_of_likes = $select_likes->rowCount();
```

```
    ?>
    <h3><?= $numbers_of_likes; ?></h3>
    <p>Whishlisted Property</p>
    <a href="view_posts.php" class="btn">View Lisitngs</a>
  </div>

  </div>

</section>
<script src="../js/admin_script.js"></script>

</body>
</html>
```

Add_post.php

```
<?php

include '../components/connect.php';

session_start();

$admin_id = $_SESSION['admin_id'];
```

```
if(!isset($admin_id)){
  header('location:admin_login.php');
}

if(isset($_POST['publish'])){

  $name = $_POST['name'];
  $name = filter_var($name, FILTER_SANITIZE_STRING);
  $title = $_POST['title'];
  $title = filter_var($title, FILTER_SANITIZE_STRING);
  $content = $_POST['content'];
  $content = filter_var($content, FILTER_SANITIZE_STRING);
  $category = $_POST['category'];
  $category = filter_var($category, FILTER_SANITIZE_STRING);
  $status = 'active';

  $image = $_FILES['image']['name'];
  $image = filter_var($image, FILTER_SANITIZE_STRING);
  $image_size = $_FILES['image']['size'];
  $image_tmp_name = $_FILES['image']['tmp_name'];
  $image_folder = '../uploaded_img/'.$image;
```

```
  $select_image = $conn->prepare("SELECT * FROM `posts`
WHERE image = ? AND admin_id = ?");
  $select_image->execute([$image, $admin_id]);

  if(isset($image)){
    if($select_image->rowCount() > 0 AND $image != ''){
      $message[] = 'Image Name Repeated';
    }elseif($image_size > 200000000){
      $message[] = 'Image Size is not Supported';
    }else{
      move_uploaded_file($image_tmp_name, $image_folder);
    }
  }else{
    $image = '';
  }

  if($select_image->rowCount() > 0 AND $image != ''){
    $message[] = 'Rename your Image!';
  }else{
    $insert_post = $conn->prepare("INSERT INTO
`posts`(admin_id, name, title, content, category, image, status)
VALUES(?,?,?,?,?,?,?)");
    $insert_post->execute([$admin_id, $name, $title, $content,
$category, $image, $status]);
```

```
    $message[] = 'Property Listed!';
  }

}

if(isset($_POST['draft'])){

  $name = $_POST['name'];
  $name = filter_var($name, FILTER_SANITIZE_STRING);
  $title = $_POST['title'];
  $title = filter_var($title, FILTER_SANITIZE_STRING);
  $content = $_POST['content'];
  $content = filter_var($content, FILTER_SANITIZE_STRING);
  $category = $_POST['category'];
  $category = filter_var($category, FILTER_SANITIZE_STRING);
  $status = 'deactive';

  $image = $_FILES['image']['name'];
  $image = filter_var($image, FILTER_SANITIZE_STRING);
  $image_size = $_FILES['image']['size'];
  $image_tmp_name = $_FILES['image']['tmp_name'];
  $image_folder = '../uploaded_img/'.$image;
```

```
  $select_image = $conn->prepare("SELECT * FROM `posts`
WHERE image = ? AND admin_id = ?");
  $select_image->execute([$image, $admin_id]);

  if(isset($image)){
    if($select_image->rowCount() > 0 AND $image != ''){
      $message[] = 'Image name Repeated!';
    }elseif($image_size > 2000000){
      $message[] = 'Image Size is not Supported';
    }else{
      move_uploaded_file($image_tmp_name, $image_folder);
    }
  }else{
    $image = '';
  }

  if($select_image->rowCount() > 0 AND $image != ''){
    $message[] = 'Please Rename your Image!';
  }else{
    $insert_post = $conn->prepare("INSERT INTO
`posts`(admin_id, name, title, content, category, image, status)
VALUES(?,?,?,?,?,?,?)");
    $insert_post->execute([$admin_id, $name, $title, $content,
$category, $image, $status]);
```

```
    $message[] = 'Draft saved!';
  }

}

?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Listed Property</title>

  <!-- font awesome cdn link  -->
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">

  <!-- custom css file link  -->
  <link rel="stylesheet" href="../css/admin_style.css">
```

```
</head>
<body>
<?php include '../components/admin_header.php' ?>

<section class="post-editor">

  <h1 class="heading">Add New Listing</h1>

  <form action="" method="post" enctype="multipart/form-data">
    <input type="hidden" name="name" value="<?=
$fetch_profile['name']; ?>">
    <p>Propert Name <span>*</span></p>
    <input type="text" name="title" maxlength="100" required
placeholder="Enter Title" class="box">
    <p>Description <span>*</span></p>
    <textarea name="content" class="box" required
maxlength="1000000" placeholder="" cols="30" rows="10">
Location -

Listing Price -  Rs.

Property -

Area in sq -
```

```
Building Type -

Parking availability -

Description of Property-
        </textarea>
      <p>Category<span>*</span></p>
      <select name="category" class="box" required>
        <option value="" selected disabled>Select Below</option>
        <option value="Rent">Rent</option>
        <option value="Sale">Sale</option>
      </select>
      <p>Upload Image</p>
      <input type="file" name="image" class="box"
accept="image/jpg, image/jpeg, image/png, image/webp,
video/mp4">
      <div class="flex-btn">
        <input type="submit" value="List Property" name="publish"
class="btn">
        <input type="submit" value="save as draft" name="draft"
class="option-btn">
      </div>
    </form>
```

```
</section>
<script src="../js/admin_script.js"></script>
</body>
</html>
```

Edit_post.php

```
<?php

include '../components/connect.php';

session_start();

$admin_id = $_SESSION['admin_id'];

if(!isset($admin_id)){
   header('location:admin_login.php');
}

if(isset($_POST['save'])){

   $post_id = $_GET['id'];
```

```
$title = $_POST['title'];
$title = filter_var($title, FILTER_SANITIZE_STRING);
$content = $_POST['content'];
$content = filter_var($content, FILTER_SANITIZE_STRING);
$category = $_POST['category'];
$category = filter_var($category, FILTER_SANITIZE_STRING);
$status = $_POST['status'];
$status = filter_var($status, FILTER_SANITIZE_STRING);

$update_post = $conn->prepare("UPDATE `posts` SET title = ?,
content = ?, category = ?, status = ? WHERE id = ?");
$update_post->execute([$title, $content, $category, $status,
$post_id]);

$message[] = 'Lisitng Updated!';

$old_image = $_POST['old_image'];
$image = $_FILES['image']['name'];
$image = filter_var($image, FILTER_SANITIZE_STRING);
$image_size = $_FILES['image']['size'];
$image_tmp_name = $_FILES['image']['tmp_name'];
$image_folder = '../uploaded_img/'.$image;
```

```
  $select_image = $conn->prepare("SELECT * FROM `posts`
WHERE image = ? AND admin_id = ?");
  $select_image->execute([$image, $admin_id]);

  if(!empty($image)){
    if($image_size > 20000000000){
      $message[] = 'images size is too large!';
    }elseif($select_image->rowCount() > 0 AND $image != ''){
      $message[] = 'please rename your image!';
    }else{
      $update_image = $conn->prepare("UPDATE `posts` SET
image = ? WHERE id = ?");
      move_uploaded_file($image_tmp_name, $image_folder);
      $update_image->execute([$image, $post_id]);
      if($old_image != $image AND $old_image != ''){
        unlink('../uploaded_img/'.$old_image);
      }
      $message[] = 'image updated!';
    }
  }


}
```

```
if(isset($_POST['delete_post'])){

  $post_id = $_POST['post_id'];
  $post_id = filter_var($post_id, FILTER_SANITIZE_STRING);
  $delete_image = $conn->prepare("SELECT * FROM `posts`
WHERE id = ?");
  $delete_image->execute([$post_id]);
  $fetch_delete_image = $delete_image-
>fetch(PDO::FETCH_ASSOC);
  if($fetch_delete_image['image'] != ''){
    unlink('../uploaded_img/'.$fetch_delete_image['image']);
  }
  $delete_post = $conn->prepare("DELETE FROM `posts` WHERE
id = ?");
  $delete_post->execute([$post_id]);
  $delete_comments = $conn->prepare("DELETE FROM
`comments` WHERE post_id = ?");
  $delete_comments->execute([$post_id]);
  $message[] = 'Lisitng Deleted Successfully!';

}

if(isset($_POST['delete_image'])){
```

```
  $empty_image = '';
  $post_id = $_POST['post_id'];
  $post_id = filter_var($post_id, FILTER_SANITIZE_STRING);
  $delete_image = $conn->prepare("SELECT * FROM `posts`
WHERE id = ?");
  $delete_image->execute([$post_id]);
  $fetch_delete_image = $delete_image-
>fetch(PDO::FETCH_ASSOC);
  if($fetch_delete_image['image'] != ''){
    unlink('../uploaded_img/'.$fetch_delete_image['image']);
  }
  $unset_image = $conn->prepare("UPDATE `posts` SET image = ?
WHERE id = ?");
  $unset_image->execute([$empty_image, $post_id]);
  $message[] = 'Image Deleted Successfully!';

}


?>

<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Property</title>

  <!-- font awesome cdn link  -->
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">

  <!-- custom css file link  -->
  <link rel="stylesheet" href="../css/admin_style.css">

</head>
<body>

<?php include '../components/admin_header.php' ?>

<section class="post-editor">

  <h1 class="heading">Edit Lisitings</h1>

  <?php
```

```
    $post_id = $_GET['id'];
    $select_posts = $conn->prepare("SELECT * FROM `posts`
WHERE id = ?");
    $select_posts->execute([$post_id]);
    if($select_posts->rowCount() > 0){
      while($fetch_posts = $select_posts-
>fetch(PDO::FETCH_ASSOC)){
  ?>
  <form action="" method="post" enctype="multipart/form-data">
    <input type="hidden" name="old_image" value="<?=
$fetch_posts['image']; ?>">
    <input type="hidden" name="post_id" value="<?=
$fetch_posts['id']; ?>">
    <p>Property Status<span>*</span></p>
    <select name="status" class="box" required>
      <!-- <option value="<?= $fetch_posts['status']; ?>"
selected><?= $fetch_posts['status']; ?></option> -->
      <option value="active">Active</option>
      <option value="deactive">Deactive</option>
    </select>
    <p>Property Title <span>*</span></p>
    <input type="text" name="title" maxlength="100" required
placeholder="Enter Title" class="box" value="<?=
$fetch_posts['title']; ?>">
```

```
    <p>Property Description <span>*</span></p>
    <textarea name="content" class="box" required
maxlength="10000" placeholder="" cols="30" rows="10"><?=
$fetch_posts['content']; ?></textarea>
    <p>Property Category <span>*</span></p>
    <select name="category" class="box" required>
      <option value="<?= $fetch_posts['category']; ?>" selected><?=
$fetch_posts['category']; ?></option>
      <option value="Rent">Rent</option>
      <option value="Sale">Sale</option>

    </select>
    <p>IMAGE </p>
    <input type="file" name="image" class="box"
accept="image/jpg, image/jpeg, image/png, image/webp,
video/mp4">
    <?php if($fetch_posts['image'] != ''){ ?>
      <img src="../uploaded_img/<?= $fetch_posts['image']; ?>"
class="image" alt="">
      <input type="submit" value="delete image" class="inline-
delete-btn" name="delete_image">
    <?php } ?>
    <div class="flex-btn">
```

```
        <input type="submit" value="save post" name="save"
class="btn">
        <a href="view_posts.php" class="option-btn">go back</a>
        <input type="submit" value="delete post" class="delete-btn"
name="delete_post">
    </div>
  </form>

  <?php
        }
    }else{
        echo '<p class="empty">no posts found!</p>';
  ?>
  <div class="flex-btn">
    <a href="view_posts.php" class="option-btn">view listings</a>
    <a href="add_posts.php" class="option-btn">add listings</a>
  </div>
  <?php
    }
  ?>

</section>

<!-- custom js file link  -->
```

```
<script src="../js/admin_script.js"></script>


</body>
</html>
```

User_registration.php

```
<?php

include 'components/connect.php';

session_start();

if(isset($_SESSION['user_id'])){
  $user_id = $_SESSION['user_id'];
}else{
  $user_id = '';
};

if(isset($_POST['submit'])){

  $name = $_POST['name'];
  $name = filter_var($name, FILTER_SANITIZE_STRING);
```

```
  $email = $_POST['email'];
  $email = filter_var($email, FILTER_SANITIZE_STRING);
  $pass = sha1($_POST['pass']);
  $pass = filter_var($pass, FILTER_SANITIZE_STRING);
  $cpass = sha1($_POST['cpass']);
  $cpass = filter_var($cpass, FILTER_SANITIZE_STRING);

  $select_user = $conn->prepare("SELECT * FROM `users`
WHERE email = ?");
  $select_user->execute([$email]);
  $row = $select_user->fetch(PDO::FETCH_ASSOC);

  if($select_user->rowCount() > 0){
    $message[] = 'Email Already Exists!';
  }else{
    if($pass != $cpass){
      $message[] = 'Confirm Password not Matched!';
    }else{
      $insert_user = $conn->prepare("INSERT INTO `users`(name,
email, password) VALUES(?,?,?)");
      $insert_user->execute([$name, $email, $cpass]);
      $select_user = $conn->prepare("SELECT * FROM `users`
WHERE email = ? AND password = ?");
      $select_user->execute([$email, $pass]);
```

```
        $row = $select_user->fetch(PDO::FETCH_ASSOC);
        if($select_user->rowCount() > 0){
          $_SESSION['user_id'] = $row['id'];
          header('location:home.php');
        }
      }
   }

}

?>

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Registration Form</title>
   <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">
```

```
  <!-- custom css file link  -->
  <link rel="stylesheet" href="css/style.css">

</head>
<body>

<!-- header section starts  -->
<?php include 'components/user_header.php'; ?>
<!-- header section ends -->

<section class="form-container">

  <form action="" method="post">
    <h3>Registration Form</h3>
    <input type="text" name="name" required placeholder="Name"
class="box" maxlength="50">
    <input type="email" name="email" required
placeholder="Email" class="box" maxlength="50"
oninput="this.value = this.value.replace(/\s/g, '')">
    <input type="password" name="pass" required
placeholder="Password" class="box" maxlength="50"
oninput="this.value = this.value.replace(/\s/g, '')">
```

```
    <input type="password" name="cpass" required
placeholder="Confirm Password" class="box" maxlength="50"
oninput="this.value = this.value.replace(/\s/g, '')">
    <input type="submit" value="register!" name="submit"
class="btn">
    <p><a href="login.php">Login Here</a></p>
    <p>To Become a Broker Contact <a
href="mailto:manish356111@gmail.com?subject=Registration%20f
or%20Broker&body=Broker%20Name--%20Contact%20Details--
%20Broker%20Address--">Here</a></p>
  </form>

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

<!-- custom js file link  -->
<script src="js/script.js"></script>

</body>
</html>
```

User_login.php

```
<?php

include 'components/connect.php';

session_start();

if(isset($_SESSION['user_id'])){
  $user_id = $_SESSION['user_id'];
}else{
  $user_id = '';
};

if(isset($_POST['submit'])){

  $email = $_POST['email'];
  $email = filter_var($email, FILTER_SANITIZE_STRING);
  $pass = sha1($_POST['pass']);
  $pass = filter_var($pass, FILTER_SANITIZE_STRING);

  $select_user = $conn->prepare("SELECT * FROM `users`
WHERE email = ? AND password = ?");
  $select_user->execute([$email, $pass]);
  $row = $select_user->fetch(PDO::FETCH_ASSOC);
```

```
  if($select_user->rowCount() > 0){
    $_SESSION['user_id'] = $row['id'];
    header('location:home.php');
  }else{
    $message[] = 'incorrect username or password!';
  }

}

?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Login</title>

  <!-- font awesome cdn link  -->
```

```
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">

  <!-- custom css file link  -->
  <link rel="stylesheet" href="css/style.css">

</head>
<body>

<!-- header section starts  -->
<?php include 'components/user_header.php'; ?>
<!-- header section ends -->

<section class="form-container">

  <form action="" method="post">
    <h3>Login</h3>
    <input type="email" name="email" required
placeholder="Email" class="box" maxlength="50"
oninput="this.value = this.value.replace(/\s/g, '')">
    <input type="password" name="pass" required
placeholder="Password" class="box" maxlength="50"
oninput="this.value = this.value.replace(/\s/g, '')">
```

```
    <input type="submit" value="login" name="submit"
class="btn">
    <p><a href="register.php">Register Here</a></p>
    <p>To Become a Broker Contact <a
href="mailto:manish356111@gmail.com?subject=Registration%20f
or%20Broker&body=Broker%20Name--%20Contact%20Details--
%20Broker%20Address--">Here</a></p>
  </form>

</section>

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

<!-- custom js file link  -->
<script src="js/script.js"></script>

</body>
</html>
```

Posts.php

```
<?php

include 'components/connect.php';

session_start();

if(isset($_SESSION['user_id'])){
  $user_id = $_SESSION['user_id'];
}else{
  $user_id = '';
};

include 'components/like_post.php';

?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
  <title>Listings</title>

  <!-- font awesome cdn link  -->
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">

  <link rel="stylesheet" href="css/style.css">

</head>
<body>

<?php include 'components/user_header.php'; ?>

<section class="posts-container">

  <h1 class="heading">Latest Listings</h1>

  <div class="box-container">

    <?php
      $select_posts = $conn->prepare("SELECT * FROM `posts`
WHERE status = ?");
      $select_posts->execute(['active']);
```

```
        if($select_posts->rowCount() > 0){
          while($fetch_posts = $select_posts-
>fetch(PDO::FETCH_ASSOC)){

            $post_id = $fetch_posts['id'];

            $count_post_comments = $conn->prepare("SELECT *
FROM `comments` WHERE post_id = ?");
            $count_post_comments->execute([$post_id]);
            $total_post_comments = $count_post_comments-
>rowCount();

            $count_post_likes = $conn->prepare("SELECT * FROM
`likes` WHERE post_id = ?");
            $count_post_likes->execute([$post_id]);
            $total_post_likes = $count_post_likes->rowCount();

            $confirm_likes = $conn->prepare("SELECT * FROM
`likes` WHERE user_id = ? AND post_id = ?");
            $confirm_likes->execute([$user_id, $post_id]);
    ?>
    <form class="box" method="post">
      <input type="hidden" name="post_id" value="<?= $post_id;
?>">
```

```
        <input type="hidden" name="admin_id" value="<?=
$fetch_posts['admin_id']; ?>">
        <div class="post-admin">
          <i class="fas fa-user"></i>
          <div>
            <a href="author_posts.php?author=<?=
$fetch_posts['name']; ?>"><?= $fetch_posts['name']; ?></a>
            <div><?= $fetch_posts['date']; ?></div>
          </div>
        </div>

        <?php
          if($fetch_posts['image'] != ''){
        ?>
        <img src="uploaded_img/<?= $fetch_posts['image']; ?>"
class="post-image" alt="">



        <?php
        }
        ?>
        <div class="post-title"><?= $fetch_posts['title']; ?></div>
```

```
        <div class="post-content content-150"><?=
$fetch_posts['content']; ?></div>
        <a href="view_post.php?post_id=<?= $post_id; ?>"
class="inline-btn">read more</a>
        <a href="category.php?category=<?= $fetch_posts['category'];
?>" class="post-cat"> <i class="fas fa-tag"></i> <span><?=
$fetch_posts['category']; ?></span></a>
        <div class="icons">
          <a href="view_post.php?post_id=<?= $post_id; ?>"><i
class="fas fa-comment"></i><span>(<?= $total_post_comments;
?>)</span></a>
          <button type="submit" name="like_post"><i class="fas fa-
heart" style="<?php if($confirm_likes->rowCount() > 0){ echo
'color:var(--red);'; } ?>  "></i><span>(<?= $total_post_likes;
?>)</span></button>
        </div>

    </form>
    <?php
      }
    }else{
      echo '<p class="empty">No Property Added</p>';
    }
    ?>
```

```
  </div>

</section>

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

<script src="js/script.js"></script>

</body>
</html>
```

5.1 Test Strategy

A comprehensive test strategy for the "Pune Estate" real estate web platform would ensure that the system meets its functional and non-functional requirements, is reliable, secure, and delivers a seamless user experience.

Here's an outline of the test strategy:

1. Test Planning:

- Define test objectives, scope, and timelines.
- Identify key features and functionalities to be tested.
- Allocate resources, including testing team members and testing environments.
- Develop a test plan outlining test scenario, test cases, and testing techniques.

2. Test Environment Setup:

- Set up testing environments that replicate production conditions, including hardware, software, and network configurations.
- Ensure compatibility with various web browsers, operating systems, and devices.

- Implement tools for test management, defect tracking, and performance monitoring.

3. Functional Testing:

- Verify that all functional requirements are implemented correctly and meet user expectations.

- Conduct test scenarios covering user registration, property listings, property viewing, and user profiles.

- Validate data input validation, error handling, and edge cases.

4. Usability Testing:

- Evaluate the user interface design, navigation, and overall usability of the platform.

- Gather feedback from representative users to identify usability issues and areas for improvement.

- Ensure accessibility for users with disabilities, adhering to accessibility standards (WCAG).

5. Performance Testing:

- Measure system performance under normal and peak load conditions.

- Conduct load testing, stress testing, and scalability testing to identify performance bottlenecks and optimize system resources.

- Monitor response times, throughput, and resource utilization to ensure acceptable performance levels.

6. Security Testing:

- Identify and address potential security vulnerabilities, such as SQL injection, cross-site scripting (XSS), and authentication flaws.

- Perform penetration testing to assess the resilience of the platform against external attacks.

- Implement security measures such as encryption, secure authentication mechanisms, and data protection controls.

7. Compatibility Testing:

- Validate platform compatibility across different web browsers (Chrome, Firefox, Edge, Safari), operating systems (Windows, Linux, macOS, Android, iOS), and devices (desktops, laptops, smartphones, tablets).

- Ensure responsiveness and consistent user experience across various screen sizes and resolutions.

8. Regression Testing:

- Conduct regression testing to verify that new updates or fixes do not introduce regression defects.

- Re-run previously executed test cases to ensure that existing functionalities remain intact after changes are made to the system.

9. Documentation and Reporting:

- Document test plans, test cases, test results, and any defects found during testing.

- Generate test reports summarizing testing activities, findings, and recommendations for improvement.

- Communicate test results to stakeholders and collaborate on addressing identified issues.

By following this test strategy, the "Pune Estate" platform can undergo thorough testing to ensure quality, reliability, security, and usability before being deployed to production and made available to brokers and buyers in the Pune real estate market.

5.2 Unit Test Plan

This is the lowest level of testing that is conducted to remove syntax & logic errors from a single unit. Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

Here are some Unit Test Plans:

1. User Authentication and Authorization Module

   - Verify user registration process.

   - Test user login functionality.

   - Validate user roles and permissions.

2. Property Listings Management Module

   - Verify adding new property listings.

   - Test editing and deleting property listings.

   - Validate data integrity and consistency in the database.

3. Property Search

- Verify property search functionality.
- Test filtering properties by name, category, etc.
- Validate search results accuracy.

4. View Property Details

- Verify viewing detailed property information.
- Test displaying property photos and amenities.
- Validate contact seller functionality.

7. Broker/User Profile

- Verify updating user profile information.
- Test changing passwords.
- Validate user profile data persistence.

5.3 Acceptance Test Plan

This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system customer rather than simulated test data. Acceptance testing may reveal errors & omissions in the system requirements definition because the real data exercise the system in different ways from the test data. It may also reveal requirements problems where the system's facilities do not really meet the user's needs or the system performance is unacceptable.

1. User Registration and Authentication

   - Validate user registration process.

   - Test user login functionality and authentication mechanisms.

2. Property Listings Management

   - Verify adding, editing, and deleting property listings.

   - Validate data accuracy and consistency in property listings.

3. Property Search

   - Test property search functionality with various criteria.

   - Validate search results accuracy and relevance.

4. View Property Details

- Verify viewing detailed property information.
- Test displaying property photos and amenities.

5. Broker/User Profile Management

- Validate updating user profile information.
- Test changing passwords and communication preferences.

5.4 Test Case / Test Script

Users:

TEST CASE ID	SCENARIO TO TEST	STEPS TO PERFORMANCE	EXPECTED RESULTS	ACTUAL RESULTS	PASS/ FAIL
TC1	User Registration	1. Enter name, email address, password, confirm password. 2. Click on the sign-up button.	Website should expect valid details enter by user and should redirect user to their home page.	Registration into Application is successful.	Pass
TC2.1	Login	1. Clicks on the login menu. 2. Enter the valid email and password. 3. Click on the login in button.	Website should expect valid details entered by the user and should redirect user to home page.	Login to Application is successful.	Pass
TC2.2	Login	1. Open the Login page. 2. Enter invalid username	Website should not accept invalid email	Login denied with	Pass

		3. Click on the sign in button.	address. website should throw message. "Invalid Credentials".	appropriate message.	
TC2.3	Login	1. Open the Login page. 2.Enter the valid username. 3. Enter invalid password. 4. Click on the login button.	Website should not accept invalid password. Website should throw message "Invalid Credentials"	Login denied with appropriate message.	Pass
TC3	Search Post	Search listing with broker name or title if known.	Application shows the listings under that search	Can easily access view listings.	Pass
TC4	Post	User should be able view posts on their profile.	User should be able view posts.	Posts displayed on the homepage	Pass

TC5	Comment/ Likes	User should be able to Wishlist or Comment on a Listing.	User should be able to see or do or edit their comments or listing	Display Comments and users Wishlist.	Pass

Broker:

TEST CASE ID	SCENARIO TO TEST	STEPS TO PERFORMANCE	EXPECTED RESULTS	ACTUAL RESULTS	PASS/ FAIL
TC1	Login	1. Enter the valid email and password. 2. Click on the login in button.	Website should expect valid details entered by the broker and should redirect broker to dashboard	Login to Application is successful.	Pass
TC2	Add Property	1. Clicks add Listings	Website should expect valid	Listing Published	Pass

		2. Add all Required Details 3. Click on Publish	details entered by the broker and publish the post	successful message should appear	
TC3	Edit Listings	1. Edit all Details as required 2. Click on Update	Accept all Edit inputs and replace the input in database	Listing successfully updated message should appear	Pass
TC4	Delete Listings	1 Manage Listings 2 Press on Delete Listing	It should delete that particular listing	Listing Deleted Successfully message should appear	Pass
TC5	Delete Comments	1.View Listings Comments 2 If found improper delete button should be pressed	it should delete the selected comment	Comments deleted successfully message should appear	Pass

5.5 Defect report/ Test Log

Defect Report:

Defect ID	Description	Severity	Status	Assigned To	Date Reported
DEF-001	Login button not functioning	High	Open	Developer	Date
DEF-002	Can't Edit Profile	Medium	Open	Developer	Date
DEF-003	Can't View Posts	High	Fixed	Developer	Date
DEF-004	Incorrect number of likes displayed	Low	Fixed	Developer	Date

Test Log:

ID	Scenario	Steps to perform	Expected Result	Result	Pass/ Fail	Tested
TC1	Login	Enter valid username and password.	User should be logged in.	Pass	Pass	Date
TC2	Search Posts	Search for a post by title/ category	Listing should be found.	Pass	Pass	Date
TC3	Post	Create a new post.	Post should be created and displayed.	Pass	Pass	Date
TC4	Comments/ Likes	View the Likes/ Comment.	All details should be displayed	Pass	Pass	Date

The limitations of the proposed system include:

- Limited Scalability:

The system may face challenges in scaling up to accommodate a growing user base or increasing volume of property listings. This could result in performance issues or the need for significant system upgrades as the platform expands.

- Complexity of Property Data:

Managing various types of property data, including images, descriptions, and specifications, could pose challenges in terms of storage, retrieval, and display, especially as the database grows larger.

- Security Concerns:

As the system deals with sensitive information such as property details, user profiles, ensuring robust security measures against data breaches, unauthorized access, and cyber-attacks is crucial. Any security vulnerabilities could lead to compromised user data and loss of trust.

- User Experience:

While the system aims to provide functionality for property owners, agents, and buyers, ensuring a seamless and intuitive user experience for all types of users can be challenging. User interfaces may need to be continuously improved based on user feedback to enhance usability.

- Integration with External Systems:

Integrating with external systems such as payment gateways, mapping services, or real estate APIs may pose integration challenges, requiring thorough testing and coordination with third-party providers.

Here are some proposed enhancements for the Pune Estate system:

- Mobile Application:

Develop a mobile application for the REMS system to provide users with more flexibility and convenience in accessing property listings, managing their accounts, and conducting transactions on-the-go

- Advanced Search Filters:

Enhance the search functionality with advanced filters such as price range, property size, amenities, and proximity to landmarks. This will help users refine their search criteria and find properties that meet their specific requirements more efficiently.

- Multilingual Support:

Offer multilingual support to cater to a diverse user base, especially in regions with multiple languages spoken. This can enhance accessibility and user experience for non-native speakers.

- Chatbot Assistance:

Implement a chatbot feature to provide real-time assistance to users, answer common questions, and guide them through the property search and transaction process. This can improve user engagement and customer service efficiency.

In conclusion, the Pune Estate project presents a robust platform for buying, selling, and renting properties, with distinct modules catering to the needs of property brokers, and buyers. The system offers various functionalities such as property listing management, user profile customization, search capabilities to facilitate seamless property transactions.

While the project demonstrates significant potential and functionality, there are areas for improvement and further development. The proposed enhancements, including the introduction of a mobile application, advanced search filters, virtual tours, predictive analytics, and integration with social media, can elevate the platform's capabilities and user experience.

Moreover, addressing limitations such as scalability, security concerns, user experience optimization, and compliance with legal regulations is essential for the long-term success and sustainability of the system.

By continuously refining and enhancing the platform based on user feedback, technological advancements, and market trends, the project can position itself as a leading solution in the real estate industry, offering unparalleled convenience, transparency, and value to its users.