CSE471 : Database Systems
Project Report
RealHome

Group No : 04 , CSE471 Lab Section : 01 , Spring 2024		
ID	**Name**	**Contribution**
20201075	Atif Ronan	M1(F1,F5),M2(F5),M3(F1,F3),M4(F1)
20201069	Nafiun Al Amin	M1(F4),M2(F3),M3(F5),M4(F4,F5)
20201104	Nusaiba Zaman	M1(F2),M2(F1,F2),M3(F2),M4(F3)
20201152	Ayan Haider	M1(F3),M2(F4),M3(F4),M4(F2,F6)

Table of Contents

Introduction

The real estate industry is one of Bangladesh's leading contributors to its economy. In fact, according to Barua et al. (2010, 239), with the growing population and people deciding to move inward in Dhaka for work, the real estate industry has become an attractive sector for investments. Everyone, starting with the lower middle class to the aspiration class, wants to own pieces of land or properties, and with our website, 'RealHome' we plan to make their investment choice easier. In fact, the website is built for everyone, whether someone wants to own a small portion of a property through our property tokenization system, bid for highly demanding properties through our auction system, or get a price prediction on a property. Apart from these, several other features have been added to build a user friendly environment, with over twenty currently added features.

Project Features

Module 1:

Users or Admins can register for an account within the system, before logging in. The property sellers are able to post pictures and descriptions of their property. These properties are verified through the admin panel. When a property is verified a tick sign is shown on the property post. Lastly, admin can search for properties based on their verification status, to view only verified/unverified properties.

Components: Login, Property posting, Search button, Verification of Properties, Admin Panel

Feature 1: Registration, Log in, Log out for users (Buyers and Sellers) and Admins

Feature 2: Sellers are able to post pictures and descriptions of a property.

Feature 3: Admin can filter through verified and unverified posts.

Feature 4: Property posts are verified by the admin panel through a verification sign.

Feature 5: Admin Panel Creation

Module 2:

The system can host auctions for properties. The sellers can post for an auction, these posts can be viewed by Admin panel. The users can also view the ongoing and previous auctions while participating. The buyers can press a Bid now button to increase their bid more than the last bid. Admins can check the history for properties when it comes to users that are sellers. Lastly, the system has a loan calculator, that calculates loan return amounts and interest.

Component: Auction Panel, Bidding system, Buy now price, Selling history

Feature 1: Seller can posts in Auction Panel and Users can view

Feature 2: Property post in Admin Panel with delete, status button, Auction Posts in Admin Panel

Feature 3: Calculator for Loans

Feature 4: Bidding system for buyers, they can bid with a value greater than the last bid.

Feature 5: Admins can check seller History.

Module 3:

The users can view nearby infrastructures from their selected location. They can also check for nearby shops and malls. Each property has their own QR code, request for advertisement button and comment section.

Components: Nearby Infrastructure viewing, Request for advertisement, Comment section, Nearby shops and malls, QR code

Feature 1: Users can view nearby infrastructures and their information through location search.

Feature 2: Each property will have their own comment section.

Feature 3: Each property has their own QR code for scanning.

Feature 4: Users can view nearby shops and malls through location search.

Feature 5: The sellers can request for advertisement to the admin.

Module 4:

Users can rate a property on a scale of 1 to 5. Each property will be displayed orientation wise based on it’s average rating. Comments can be liked or disliked for each property. Price can be predicted based on a Bangladesh dataset, for each property. After scanning the QR code, a pdf poster is generated. Lastly, each property can be bought in shares with other people through tokenisation.

Components: Tokenization system, LeaderBoard based on the ratings, Liking system, Rating System, QR code pdf, Price Prediction

Feature 1: Property can be tokenised through buying shares of property.

Feature 2: Users can generate a pdf poster from the QR code of a property.

Feature 3: Users can like or dislike a comment for each property.

Feature 4: Users can rate a property, the property's average will be displayed under it’s property post.

Feature 5: Property is displayed based on it’s average rating.

Feature 6: Price prediction for each property using Machine Learning.

Frontend Development

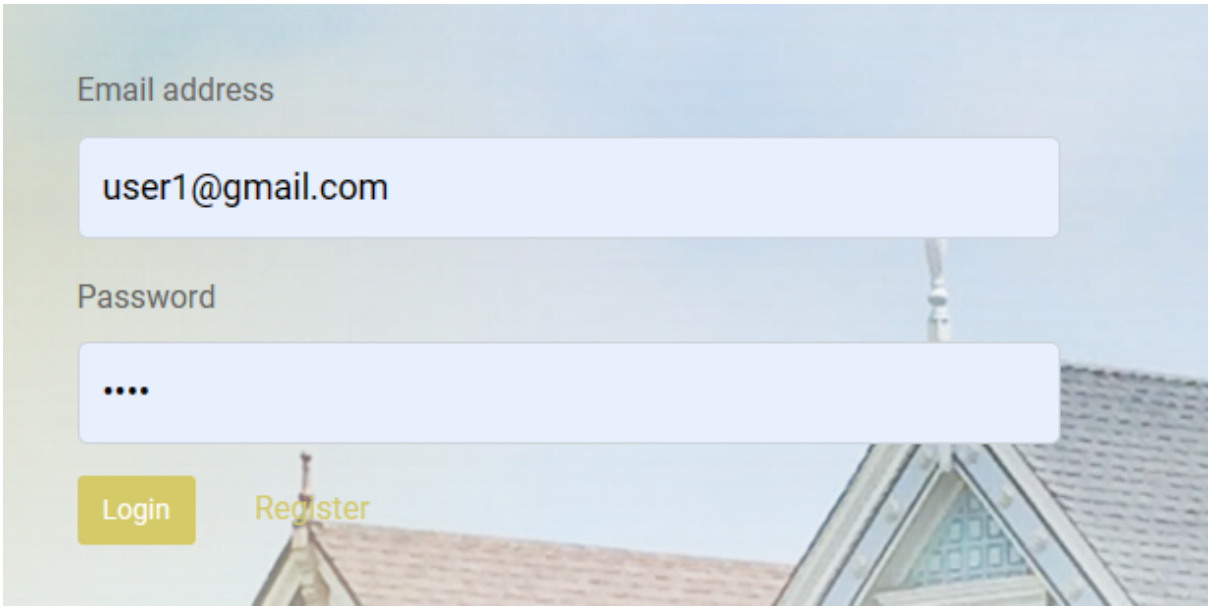
Contribution of ID : 20201075, Name : Atif Ronan

Feature 1:- Custom Authentication

Login:-

```
<div class="center-container">
    <form class='ms-auto me-auto mb-3' style="width: 500px" action="{{route('loginUser')}}" method='POST'>
        @csrf
        <div class="mb-3">
            <label  class="form-label">Email address</label>
            <input type="email" class="form-control" name="email">
        </div>

        <div class="mb-3">
            <label for="exampleInputPassword1" class="form-label">Password</label>
            <input type="password" class="form-control" name="password">
        </div>
        <button type="submit" class="btn btn-primary">Login</button><a style="margin-left: 30px;" href="{{ route('registration') }}">Register</a>
```

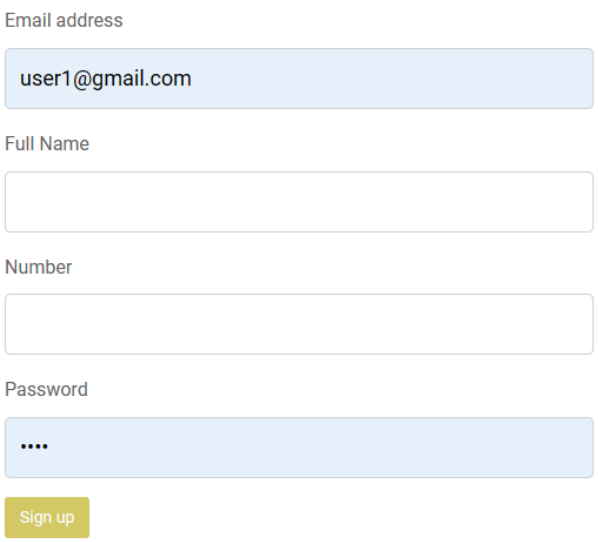
Here, using form, we take the user's email and password. And it also has the route for registration



Registration

```
<form class='ms-auto me-auto mt-3' style="width: 500px" action="{{route('registrationUser')}}" method='POST'>
    @csrf
    <div class="mb-3">
        <label  class="form-label">Email address</label>
        <input type="email" class="form-control" name="email">
    </div>

    <div class="mb-3">
        <label  class="form-label">Full Name</label>
        <input type="text" class="form-control" name="name">
    </div>

    <div class="mb-3">
        <label  class="form-label">Number</label>
        <input type="text" class="form-control" name="number">
    </div>

    <div class="mb-3">
        <label  class="form-label">Password</label>
        <input type="password" class="form-control" name="password">
    </div>
    <button type="submit" class="btn btn-primary">Sign up</button>
</form>
```

Here, through form, we take Email address, Full Name Number and Password



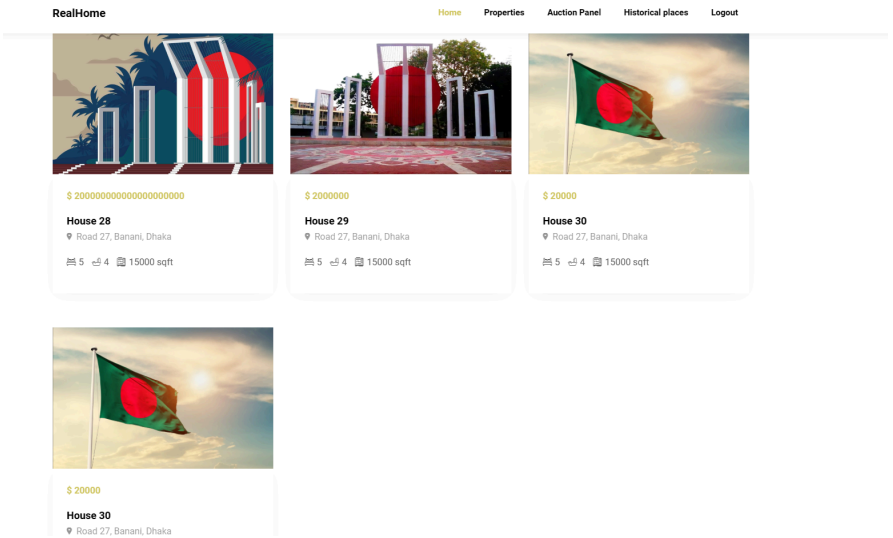
Feature 2:- Check seller history

```
</div>

<span class="subheading">{{$post-> address}}</span>
<span class="subheading">{{$post-> region}}</span>

<a class="" href="{{ url('history',$post->username) }}">Posted by {{$post->username}}</a>
<p>Average Rating: {{$post->rating}} Stars ({{$post->number_of_ratings}} Ratings)</p>
```

Here The url redirects user to the class history which redirects history.blade.php


```
<section class="ftco-section goto-here">
    <div class="container">
        <div class="row">
            @foreach($post as $post)
                <div class="col-md-4">
                    <div class="property-wrap ftco-animate">
                        <div class="img d-flex align-items-center justify-content-center" style="background-image: url(postimage/{{$post->image}});">
                            <a href="{{ url('single',$post-> id) }}" class="icon d-flex align-items-center justify-content-center btn-custom">
                                <span class="ion-ios-link"></span>
                            </a>
                            <div class="list-agent d-flex align-items-center">
                                <a href="#" class="agent-info d-flex align-items-center">
                                    <div class="img-2 rounded-circle" style="background-image: url(home/images/person_1.jpg);"></div>
                                    <h3 class="mb-0 ml-2">Ben Ford</h3>
                                </a>
                                <div class="tooltip-wrap d-flex">
                                    <a href="#" class="icon-2 d-flex align-items-center justify-content-center" data-toggle="tooltip" data-placement="top" title="Bookmark">
                                        <span class="ion-ios-heart"><i class="sr-only">Bookmark</i></span>
                                    </a>
                                    <a href="#" class="icon-2 d-flex align-items-center justify-content-center" data-toggle="tooltip" data-placement="top" title="Compare">
                                        <span class="ion-ios-eye"><i class="sr-only">Compare</i></span>
                                    </a>
                                </div>
                            </div>
                        </div>
                        <div class="text">
                            <p class="price mb-3"><span class="orig-price">$ {{$post-> price}}</span></p>
                            @if ($post->status == 'Inactive')
                                <i class="fa fa-check-circle text-failed" aria-hidden="true">Unavailable</i>
                            @endif
                            <h3 class="mb-0"><a href="{{ url('single',$post-> id) }}">{{$post-> name}}</a></h3>
                            @if ($post->Post_status == 'verified')
                                <i class="fa fa-check-circle text-success" aria-hidden="true">verified</i>
                            @endif
                            <span class="location d-inline-block mb-3"><i class="ion-ios-pin mr-2"></i>{{$post-> address}}</span>
                            <ul class="property_list">
                                <li><span class="flaticon-bed"></span>{{$post-> bedroom}}</li>
                                <li><span class="flaticon-bathtub"></span>{{$post-> bathroom}}</li>
                                <li><span class="flaticon-floor-plan"></span>{{$post-> area}} sqft</li>
                            </ul>
                            @if ($post->status != 'Inactive')
                                <p><a href="{{route('reject_post1',$post-> id)}}" class="btn btn-primary py-3 px-4">Sold</a></p>
                            @endif
                        </div>
                    </div>
                </div>
            @endforeach
        </div>
    </div>
</section>
```

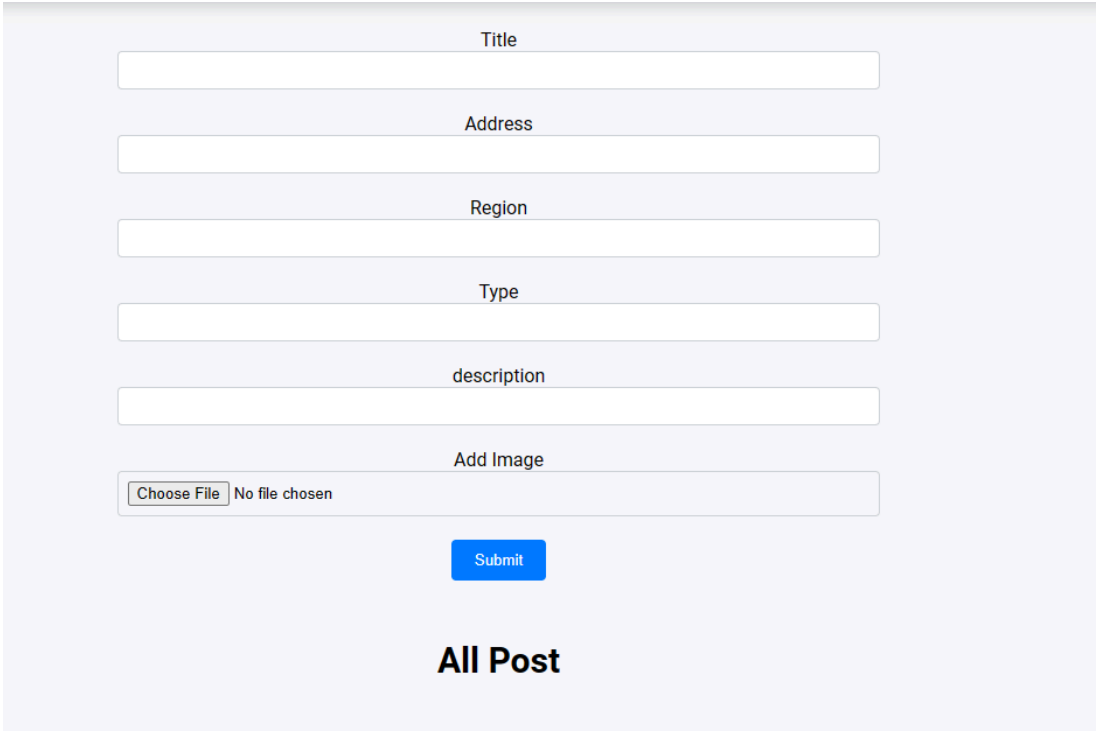
Here, the data of seller retrieved from post table is individually shown using a for loop.



Feature 4:- Users can view nearby Historical infrastructure and its detailed information:-

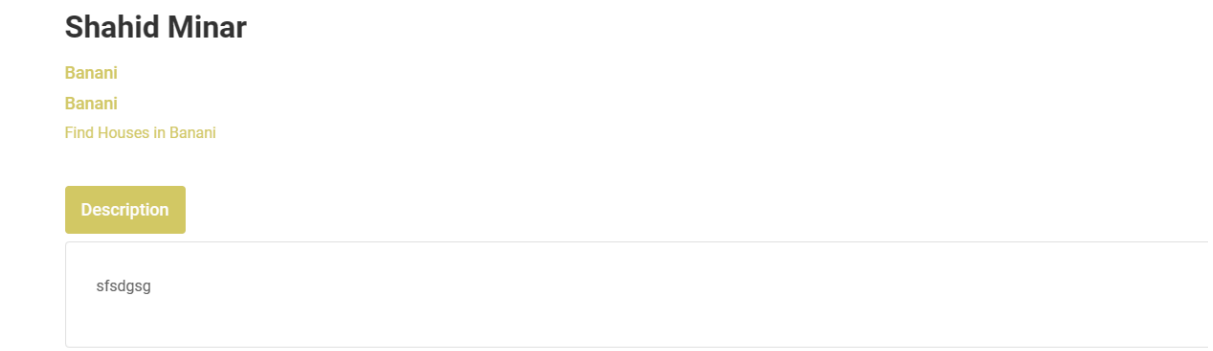
```
</head>
<body>
@include('admin.header')
    <div class="page-wrapper mdc-toolbar-fixed-adjust">
        <main class="content-wrapper1">
            <div class="page-content">
                <form action="{{ route('postnearby') }}" method="POST" enctype="multipart/form-data">
                    @csrf
                    <div class="div_center">
                        <label>Title</label>
                        <input type="text" name="title">
                    </div>
                    <div class="div_center">
                        <label>Address</label>
                        <input type="text" name="address">
                    </div>
                    <div class="div_center">
                        <label>Region</label>
                        <input type="text" name="region" pattern="[a-zA-Z]{1,}" required>
                    </div>
                    <div class="div_center">
                        <label>Type</label>
                        <input type="text" name="type">
                    </div>
                    <div class="div_center">
                        <label>description</label>
                        <input type="text" name="description">
                    </div>
                    <div class="div_center">
                        <label>Add Image</label>
                        <input type="file" name="image">
                    </div>
                    <div class="div_center">
                        <input type="submit" class="btn btn-primary" value="Submit">
                    </div>
                </form>
```

The code above uses a form to take admin inputs for Historical nearby infrastructure.



```
<section class="ftco-section ftco-property-details">
  <div class="container">
    <div class="row justify-content-center">
        <div class="col-md-12">
            <div class="property-details">
                <div class="img rounded" style="background-image: url(home/images/work-1.jpg);"></div>
                <div class="text">
                    <h2>{{$post-> name}}</h2>
                    <span class="subheading">{{$post-> address}}</span>
                    <span class="subheading">{{$post-> region}}</span>
                    <a href="{{url('specificpost',$post-> address)}}">Find Houses in {{$post-> address}}</a>
                </div>
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-md-12 pills">
            <div class="bd-example bd-example-tabs">
                <div class="d-flex">
                    <ul class="nav nav-pills mb-2" id="pills-tab" role="tablist">
                    <li class="nav-item">
                        <a class="nav-link active" id="pills-manufacturer-tab" data-toggle="pill" href="#pills-manufacturer" role="tab" aria-controls="pills-manufacturer" aria-expanded="true">D
                    </li>
                    </ul>
                </div>
                <div class="tab-content" id="pills-tabContent">
                <div class="tab-pane fade show active" id="pills-manufacturer" role="tabpanel" aria-labelledby="pills-manufacturer-tab">
                    <p>{{$post-> description}}</p>
                </div>
            </div>
        </div>
</section>
```

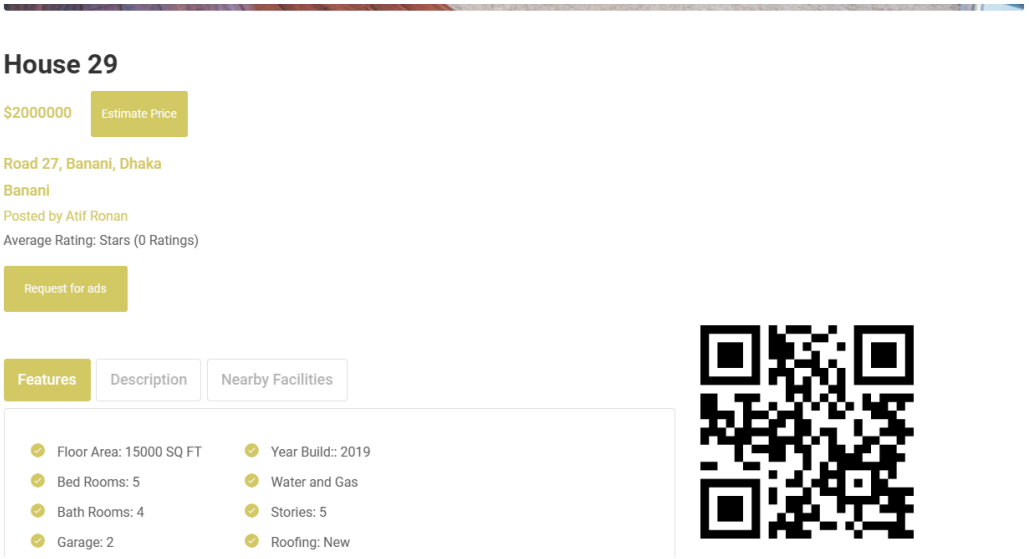
Here individual data from nearby table is shown through html.



Feature 4: Qr Code:-

```
<div class="col-md-12">
    <div class="qr-code">
        <p>{{ QrCode::size(255)->generate("http://127.0.0.1:8000/single/" . $post->id) }}</p>
    </div>
```

Using simple softwareIO, in every sing.blade.php, individual QR codes are generated

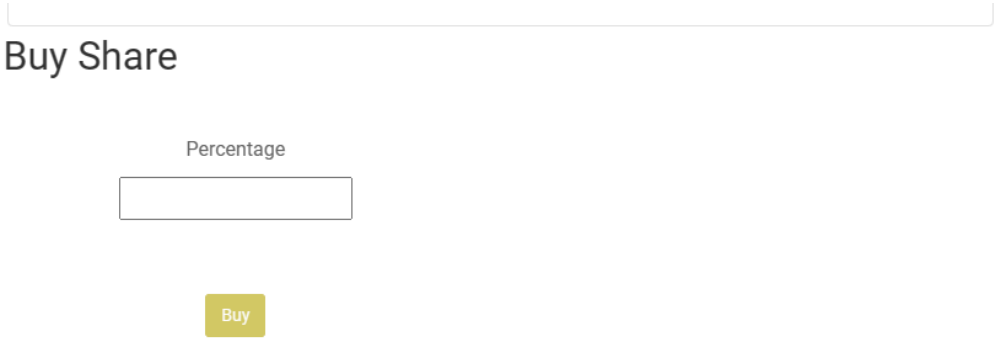


Feature 5:- The Users can buy shares of real estate:-

```
@if ($post->token!=0)
<div class="row">
<div class="col-md-4">
    <div class="row justify-content-center align-items-left">
        <div class="col-md-12">
            <h2>Buy Share</h2>
            <form action="{{ route('share',$post->id) }}" method="POST" enctype="multipart/form-data">
                @csrf
                <div class="div_center">
                    <label>Percentage</label>
                    <input type="text" name="percent">
                </div>
                <div class="div_center">
                    <input type="submit" class="btn btn-primary" value="Buy">
                </div>
            </form>
            @if ((int)$post->token==100)
                <h2>or</h2>
                    <p><a style="margin-top: 30px; margin-left: 30px;" href="{{ url('buy',$post-> id) }}" class="btn btn-primary py-3 px-3.5">Buy Property</a></p>
            @endif
        </div>
    </div>
</div>
@endif
```

```
    </section>
    <section style="background-color: #f8f9fa; padding: 30px;">
        <div class="container">
            <h2 class="text-center mb-4">All Comments</h2>
            <div class="row justify-content-center">
                <div class="col-md-8">
                @foreach($comments as $comment)
                @if ($comment->post_id==$post->id)
    <div class="card mb-3">
        <div class="card-body">
            <h5 class="card-title">{{ $comment->name }}</h5>
            <p class="card-text">{{ $comment->content }}</p>
            <div class="row">
                <div class="col-md-6">
                    <form action="{{ route('like_or_dislike', ['id' => $comment->id]) }}" method="post">
                        @csrf
                        <input type="hidden" name="action" value="like">
                        <button type="submit" class="btn btn-success"><i class="far fa-thumbs-up"></i></button>
                    </form>
                    <span>{{ $comment->likes }} likes</span>
                </div>
                <div class="col-md-6">
                    <form action="{{ route('like_or_dislike', ['id' => $comment->id]) }}" method="post">
                        @csrf
                        <input type="hidden" name="action" value="dislike">
                        <button type="submit" class="btn btn-danger"><i class="far fa-thumbs-down"></i></button>
                    </form>
                    <span>{{ $comment->dislikes }} dislikes</span>
                </div>
            </div>
        </div>
    </div>
    @endif
@endforeach
```

The codes above use html to buy real estate and buy shares of the real estate. If a user buys the real estate, the option for buying the real estate and or buying shares of the real estate wont show again as token is 0. Similarly if user buys a certain share of the real estate, token will not be 100 and the option of buying the entire real estate wont show.



Contribution of ID : 20201104, Name : Nusaiba Zaman

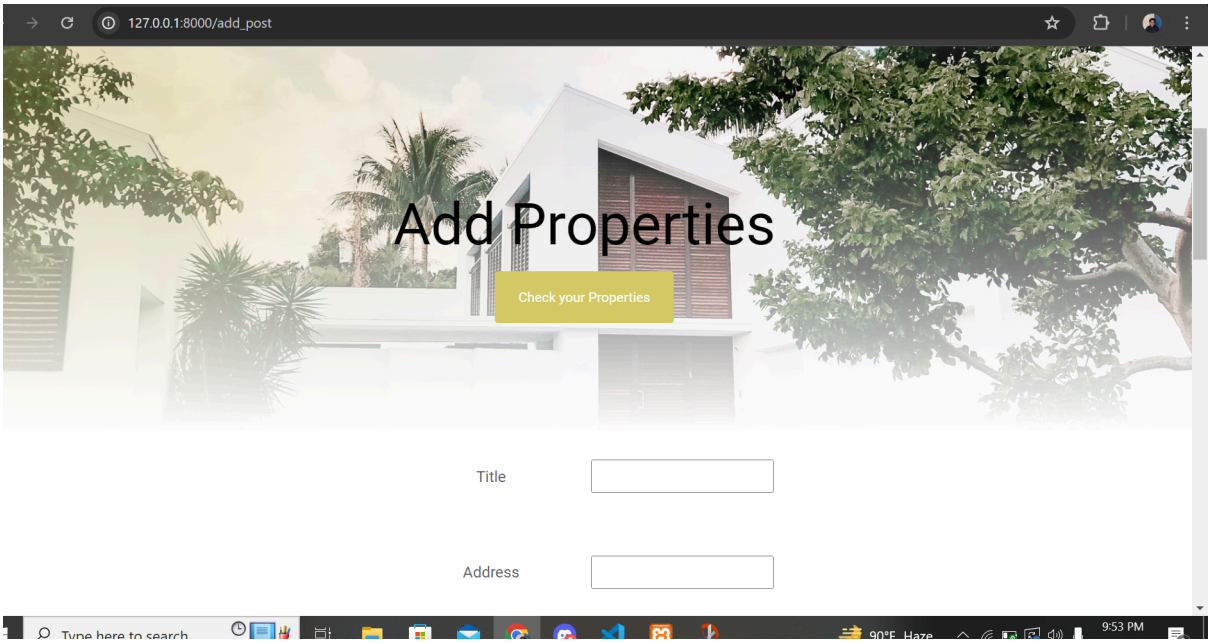
Feature 1: Sellers are able to post pictures and descriptions of a property.

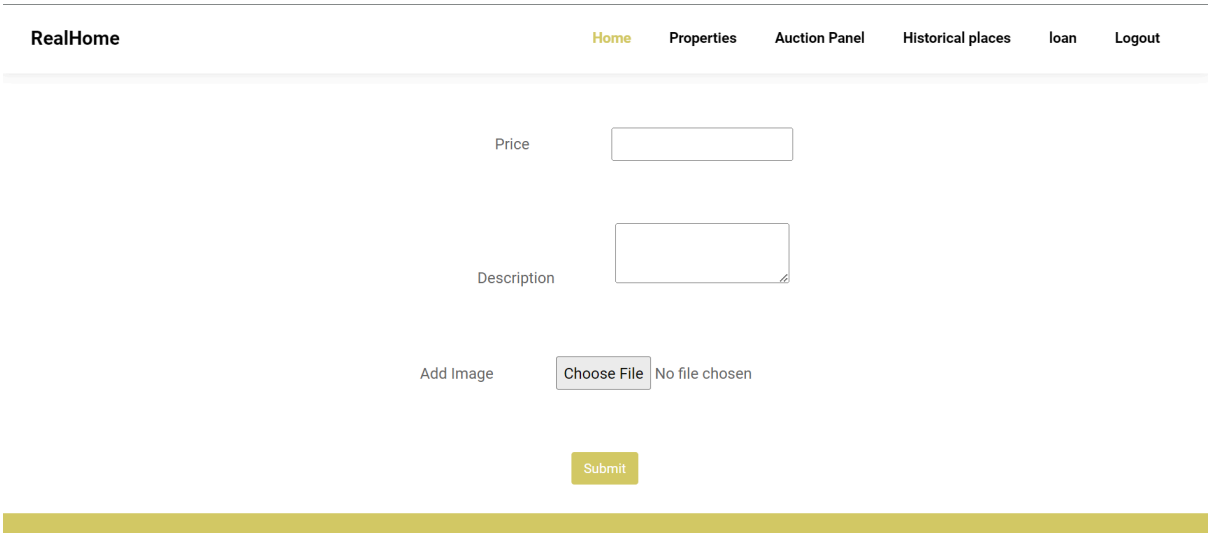
```
@section('body')
    <section class="hero-wrap hero-wrap-2 ftco-degree-bg js-fullheight" style="background-image: url('home/images/b
      <div class="container">
        <div class="row no-gutters slider-text js-fullheight align-items-end justify-content-center">
          </div>
        </div>
      </div>
    </section>
    <section>
        <form action="{{ route('poster') }}" method="POST" enctype="multipart/form-data">
            @csrf
            <div class="div_center">
                <label>Title</label>
                <input type="text" name="title">
            </div>
            <div class="div_center">
                <label>Address</label>
                <input type="text" name="address">
```

```
  <section>
      <form action="{{ route('poster') }}" method="POST" enctype="multipart/form-data">
          <div class="div_center">
              <label>Title</label>
              <input type="text" name="title">
          </div>
          <div class="div_center">
              <label>Address</label>
              <input type="text" name="address">
          </div>
          <div class="div_center">
                <label>Number of Bedroom</label>
                <input type="text" name="bedroom">
          </div>
          <div class="div_center">
              <label>Number of Bathroom</label>
              <input type="text" name="bathroom">
          </div>
```

```
      <form action="{{ route('poster') }}" method="POST" enctype="multipart/form-data">
          <div class="div_center">
          </div>
          <div class="div_center">
              <label>Number of Bathroom</label>
              <input type="text" name="bathroom">
          </div>
          <div class="div_center">
              <label>Number of Garage</label>
              <input type="text" name="garage">
          </div>
          <div class="div_center">
              <label>Floor Area</label>
              <input type="text" name="area">
          </div>
          <div class="div_center">
              <label>Stories</label>
```

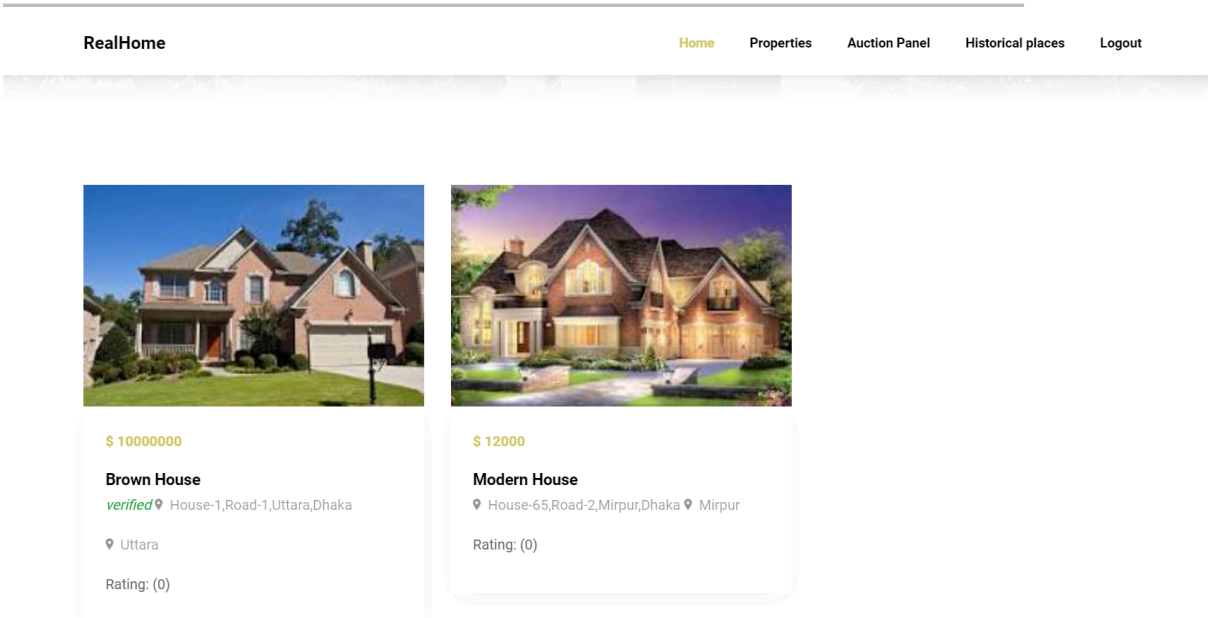
```
<form action="{{ route('poster') }}" method="POST" enctype="multipart/form-data">
    <div class="div_center">
        <input type="text" name="area">
    </div>
    <div class="div_center">
        <label>Stories</label>
        <input type="text" name="stories">
    </div>
    <div class="div_center">
        <label>Price</label>
        <input type="text" name="price">
    </div>

    <div class="div_center">
        <label>Description </label>
        <textarea name="description"></textarea>
    </div>

    <div class="div_center">
        <label>Description </label>
        <textarea name="description"></textarea>
    </div>
    <div class="div_center">
        <label>Add Image</label>
        <input type="file" name="image">
    </div>
    <div class="div_center">
        <input type="submit" class="btn btn-primary" value="Submit">
    </div>
</form>
</section>
@include('include.footer')
```





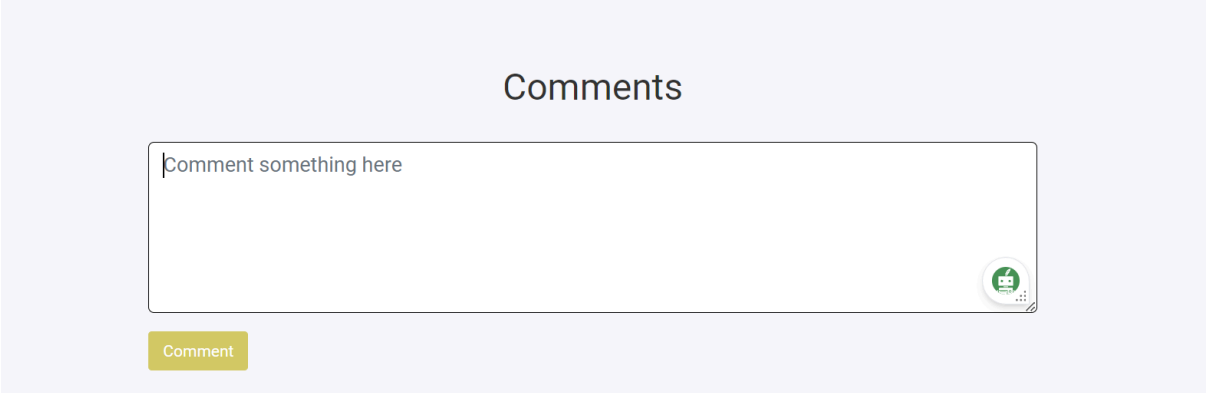
This code snippet takes in user input for property name, address, bedroom, bathroom, garage, stories, area, description and price along with the image in the form.

```
        </div>
        <div class="text">
            <p class="price mb-3"><span class="orig-price">$ {{$post->price}}</span></p>
            <h3 class="mb-0"><a href="{{ url('single',$post->id) }}">{{$post->name}}</a></h3>
            @if ($post->Post_status === 'verified')
                <i class="fa fa-check-circle text-success" aria-hidden="true">verified</i>
            @endif
            <span class="location d-inline-block mb-3"><i class="ion-ios-pin mr-2"></i>{{$post->address}}</span>
            <p class="ratings">Rating: {{$post->rating}}({{$post->number_of_ratings}})</p>
        </div>
    </div>
  </div>
ndif
reach
```



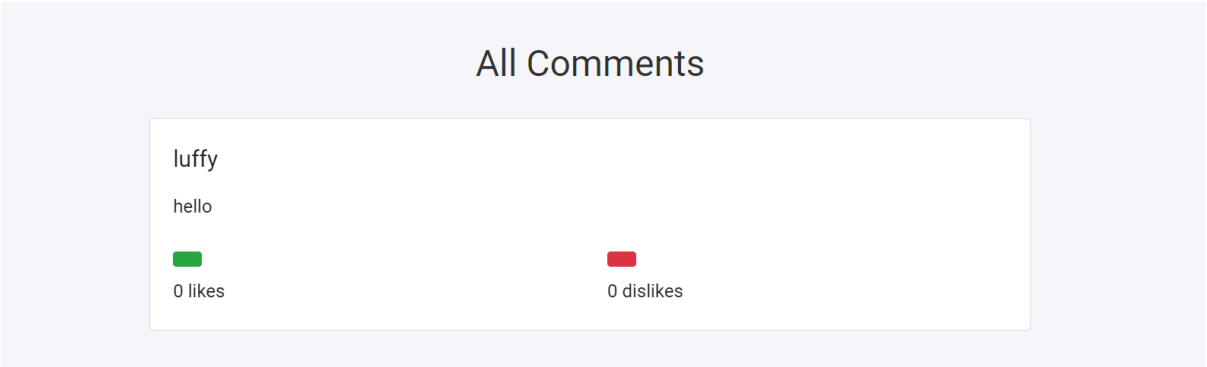
This code snippet shows the property as a post, showing the price, property name, post status and address in view form.

Feature 2: Each property will have their own comment section.




```
<form action="{{ route('comment_add',$post->id) }}" method="post">
    @csrf
    <div class="form-group">
        <textarea name="content" class="form-control" rows="5" placeholder="Comment something here"></textarea>
    </div>
    <input type="hidden" name="post_id" value="#">
    <button type="submit" class="btn btn-primary">Comment</button>
</form>
```

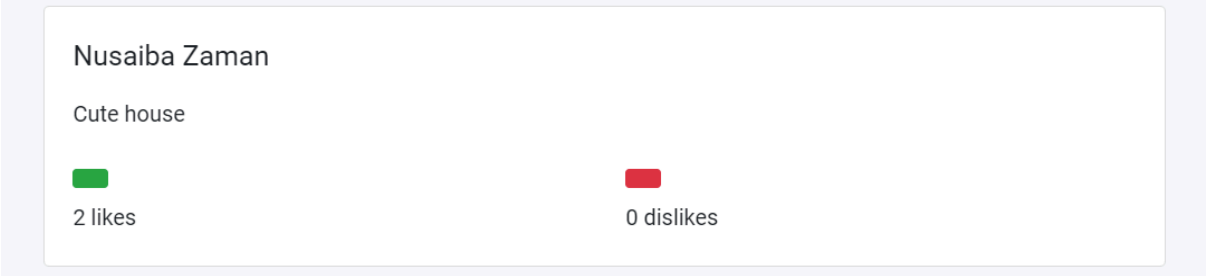
```
                @foreach($comments as $comment)
                @if ($comment->post_id==$post->id)
    <div class="card mb-3">
        <div class="card-body">
            <h5 class="card-title">{{ $comment->name }}</h5>
            <p class="card-text">{{ $comment->content }}</p>
            <div class="row">
                <div class="col-md-6">
```



The comments are written in the comments section. The comments are taken in the form by user input.

Feature 3: Users can like or dislike a comment for each property.

```
<div class="row">
    <div class="col-md-6">
        <form action="{{ route('like_or_dislike', ['id' => $comment->id]) }}" method="post">
            @csrf
            <input type="hidden" name="action" value="like">
            <button type="submit" class="btn btn-success"><i class="far fa-thumbs-up"></i></button>
        </form>
        <span>{{ $comment->likes }} likes</span>
    </div>
    <div class="col-md-6">
        <form action="{{ route('like_or_dislike', ['id' => $comment->id]) }}" method="post">
            @csrf
            <input type="hidden" name="action" value="dislike">
            <button type="submit" class="btn btn-success"><i class="far fa-thumbs-up"></i></button>
        </form>
        <span>{{ $comment->likes }} likes</span>
    </div>
    <div class="col-md-6">
        <form action="{{ route('like_or_dislike', ['id' => $comment->id]) }}" method="post">
            @csrf
            <input type="hidden" name="action" value="dislike">
            <button type="submit" class="btn btn-danger"><i class="far fa-thumbs-down"></i></button>
        </form>
        <span>{{ $comment->dislikes }} dislikes</span>
    </div>
```



The code snippet uses dynamic updating of like and dislike counts. The number of likes or dislikes are incremented or decremented upon action.

Feature 4: Seller can posts in Auction Panel and users can view

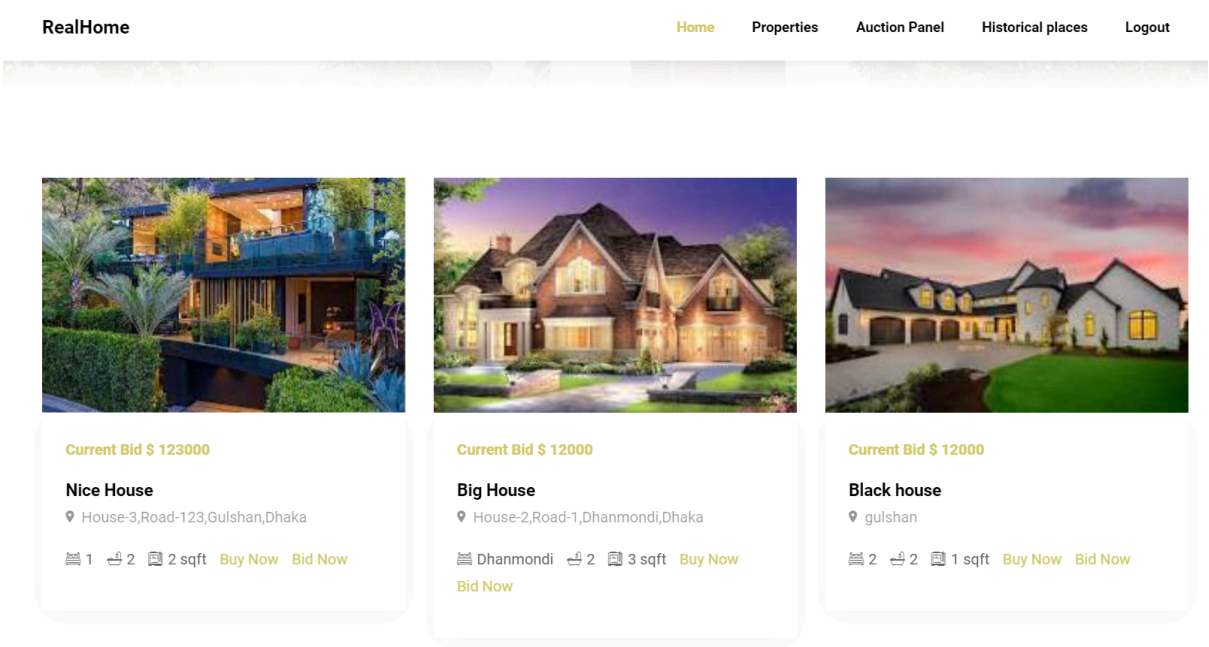
resources > views > home > panel.blade.php

```
@extends('layout')
@section('title', 'Properties')
@section('body')
    @include('include.header')
    <section class="hero-wrap hero-wrap-2 ftco-degree-bg js-fullheight" style="background-image: url('home/images/bg_
        <div class="overlay"></div>
        <div class="overlay-2"></div>
        <div class="container">
            <div class="row no-gutters slider-text js-fullheight align-items-end justify-content-center">
                <div class="col-md-9 ftco-animate pb-5 mb-5 text-center">
                    <h1 class="mb-3 bread">Auction Panel</h1>
                    <p class="breadcrumbs"><span class="mr-2"><a href="{{ route('welcome') }}">Home <i class="ion-ios
                    <p><a href="{{route('post_auction')}}" class="btn btn-primary py-3 px-4">Add Auction</a></p>
                </div>
            </div>
        </div>
    </section>

```

```
@section('body')
    <section class="ftco-section goto-here">
        <div class="container">
            <div class="row">
                @foreach($auction as $auction)
                    <div class="col-md-4 ">
                        <div class="property-wrap ftco-animate">
                            <div class="img d-flex align-items-center justify-content-center" style="background-image
                                <a href="properties-single.html" class="icon d-flex align-items-center justify-conten
                                    <span class="ion-ios-link"></span>
                                </a>
                                <div class="list-agent d-flex align-items-center">
                                    <a href="#" class="agent-info d-flex align-items-center">
                                        <div class="img-2 rounded-circle" style="background-image: url('home/images/p
                                        <h3 class="mb-0 ml-2">Avatar Aang</h3>
                                    </a>
                                    <div class="tooltip-wrap d-flex">
                                        <a href="#" class="icon-2 d-flex align-items-center justify-content-center" d
                                            <span class="ion-ios-heart"><i class="sr-only">Bookmark</i></span>
                                        </a>
```

```
</div>
<div class="text">
    <p class="price mb-3"><span class="orig-price">Current Bid $ {{ $auction->base }}</span></p>
    <p class="price mb-3"><span class="orig-price">Buy Now $ {{ $auction->price }}</span></p>
    <h3 class="mb-0"><a href="">{{ $auction->name }}</a></h3>

    <span class="location d-inline-block mb-3"><i class="ion-ios-pin mr-2"></i>{{ $auction->address }}</span>
    <ul class="property_list">
```

```
    <span class="location d-inline-block mb-3"><i class="ion-ios-pin mr-2"></i>{{ $auction->address }}</span
    <ul class="property_list">


        <li><span class="flaticon-bed"></span>{{ $auction->bedroom }}</li>
        <li><span class="flaticon-bathtub"></span>{{ $auction->bathroom }}</li>
        <li><span class="flaticon-floor-plan"></span>{{ $auction->area }} sqft</li>
        <li><a href="{#}" class="btn-buy-now">Bid Now</a>

    </ul>
```



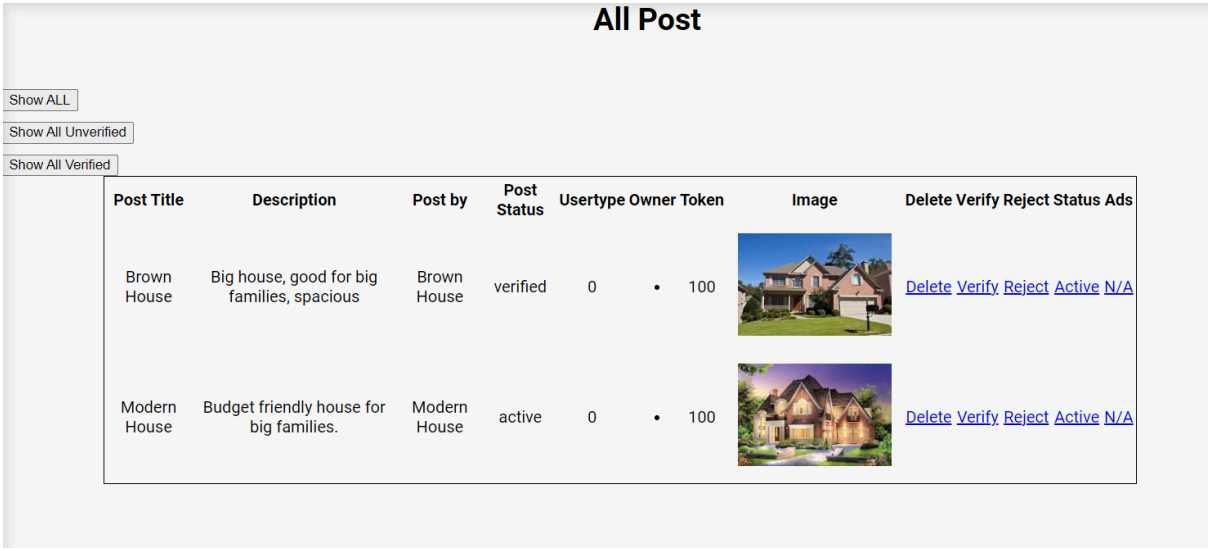
This code snippet shows the auction property as a post, showing the current ongoing bid price, property name and address in view form.

Feature 5: Property post in Admin Panel with delete, status button, Auction Posts in Admin Panel

```
<div class="page-wrapper mdc-toolbar-fixed-adjust">
      <main class="content-wrapper">
        <div class="page-content">
                <div class="row">
                    <div class="col-md-10">
                                <th>Post by</th>
                                <th>Post Status</th>
                                <th>Usertype</th>
                                <th>Owner</th>
                                <th>Token</th>
                                <th>Image</th>
                                <th>Delete</th>
                                <th>Verify</th>
                                <th>Reject</th>
                                <th>Status</th>
                                <th>Ads</th>
                            </tr>
                            @foreach($post as $singlePost)
                            <tr>
```

```
<td>
    <a href="{{ url('delete_post', $singlePost->id) }}" class="btn btn-danger">Delete</a>
</td>
```

```
<td>
    <a href="{{ route('status_post', $singlePost->id) }}" class="btn btn-primary">{{ $singlePost->status}}</
</td>
```

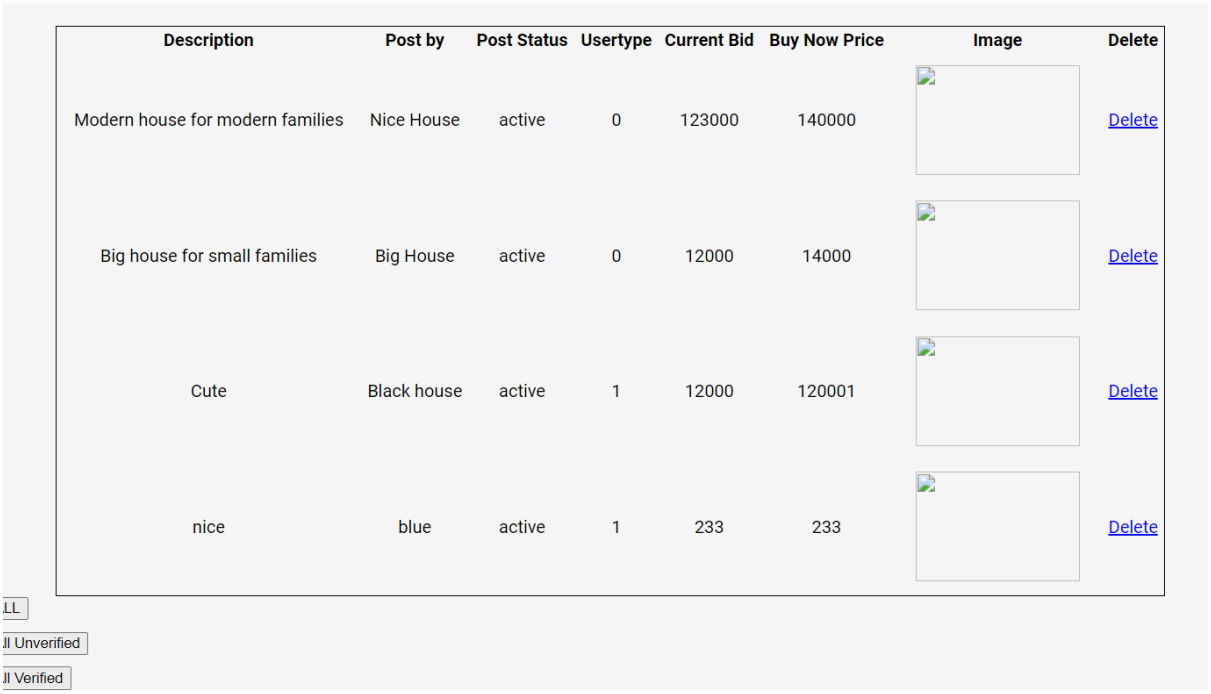


This shows the interface for the Property post in Admin Panel along with the delete and status buttons. It creates a structured view in the admin panel showing all posts with functionalities for deleting and status checking.

```
</head>
<body>
@include('admin.header')
<div class="page-wrapper mdc-toolbar-fixed-adjust">
      <main class="content-wrapper">
        <div class="page-content">
                <h1 class="title_deg">All Post</h1>

                <div class="container">
                <div class="row">
                    <div class="col-md-10">

                        <table class="table_deg">
                            <tr>

                                <th>Description</th>
                                <th>Post by</th>
                                <th>Post Status</th>
                                <th>Usertype</th>
                                <th>Current Bid</th>
                                <th>Buy Now Price</th>
                                <th>Image</th>
                                <th>Delete</th>
```

```
        </tr>
        @foreach($auction as $singlePost)
        <tr>

            <td>{{ $singlePost->description }}</td>
            <td>{{ $singlePost->name }}</td>




            <td>{{ $singlePost->Post_status }}</td>
            <td>{{ $singlePost->user_type }}</td>
            <td>{{ $singlePost->base }}</td>
            <td>{{ $singlePost->price }}</td>


            <td>
                <img class="img_deg" src="postimage/{{ $singlePost->image }}"
            </td>
```

```
            <td>{{ $singlePost->Post_status }}</td>
            <td>{{ $singlePost->user_type }}</td>
            <td>{{ $singlePost->base }}</td>
            <td>{{ $singlePost->price }}</td>


            <td>
                <img class="img_deg" src="postimage/{{ $singlePost->image }}"
            </td>
            <td>
                <a href="{{ url('delete_post_1', $singlePost->id) }}" class="
            </td>

        </tr>
        @endforeach
    </table>
</div>
```



Description	Post by	Post Status	Usertype	Current Bid	Buy Now Price	Image	Delete
Modern house for modern families	Nice House	active	0	123000	140000		Delete
Big house for small families	Big House	active	0	12000	14000		Delete
Cute	Black house	active	1	12000	120001		Delete
nice	blue	active	1	233	233		Delete

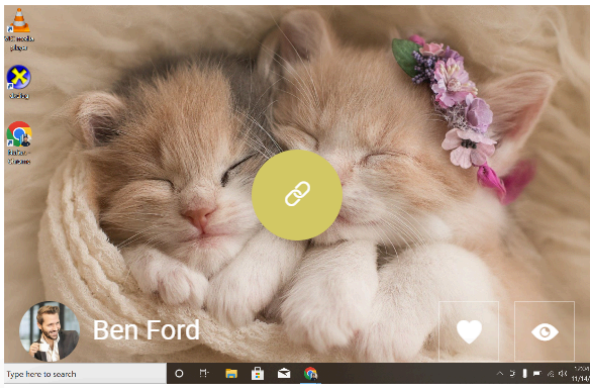
This shows the website interface for Auction in the Admin Panel. It creates a structured view to show all auction posts in the admin panel.

Contribution of ID : 20201069, Name : Nafiun Al Amin

Feature 1: Show verified post

```
<div class="text">
    <p class="price mb-3"><span class="orig-price">$ {{$post->price}}</span></p>
    <h3 class="mb-0"><a href="{{ url('single',$post->id) }}">{{$post->name}}</a></h3>
    @if ($post->Post_status === 'verified')
        <i class="fa fa-check-circle text-success" aria-hidden="true">verified</i>
    @endif
```

- This portion checks for the post_status from the post table, and adds a verified sign if the post is verified.



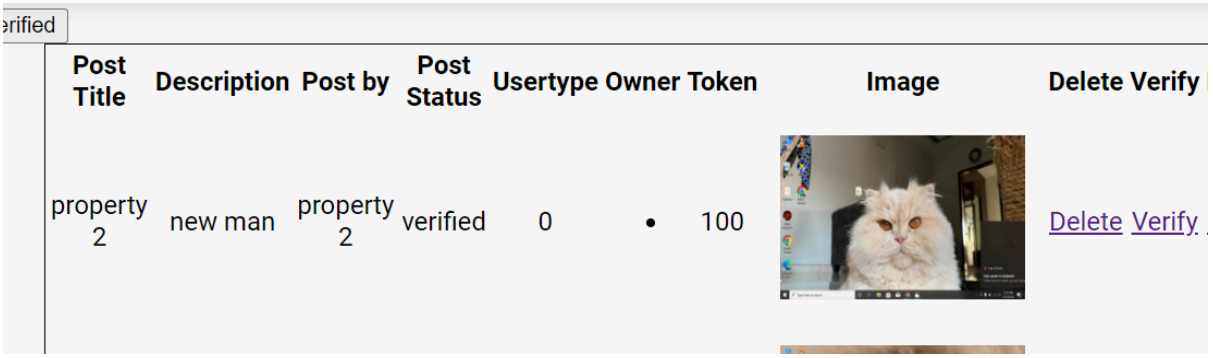
$ asa

as

verified asa

```
<td>
    <a href="{{ route('verify_post', $singlePost->id) }}" class="btn btn-outline-secondary">Verify</a>
</td>
<td>
```

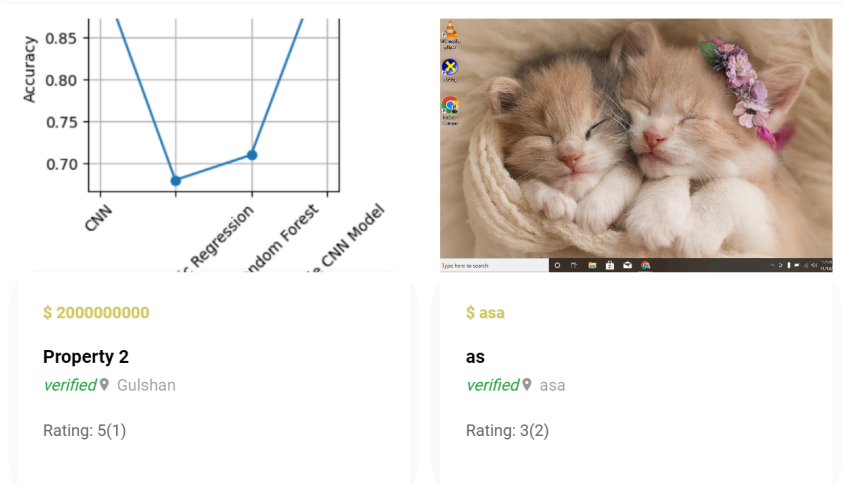
- **This displays the verify button on the admin panel of the properties.**



Post Title	Description	Post by	Post Status	Usertype	Owner Token	Image	Delete Verify
property 2	new man	property 2	verified	0	• 100		Delete Verify

Feature 2: Show Ratings

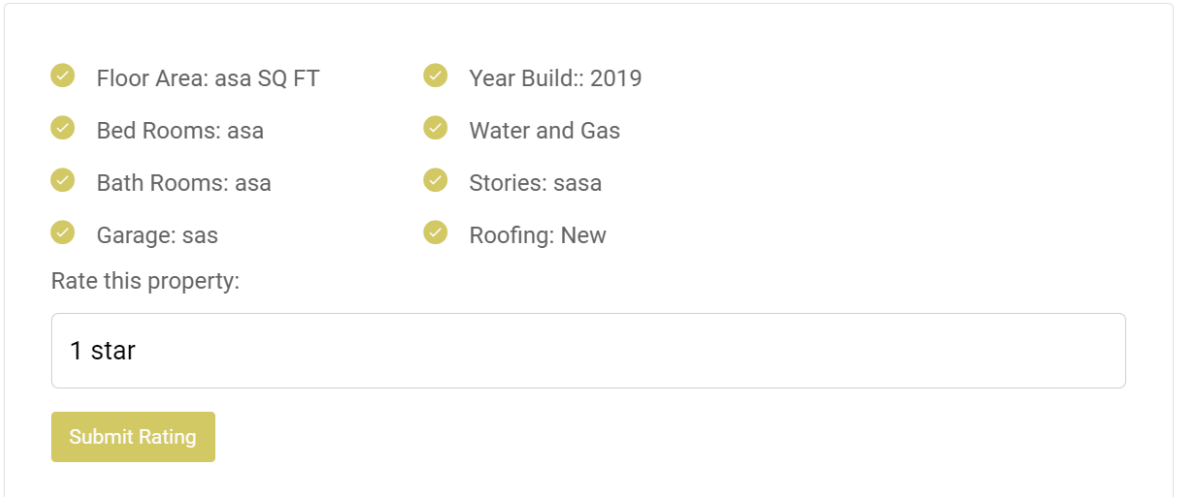
```
        <div class="text">
            <p class="price mb-3"><span class="orig-price">$ {{$post->price}}</span></p>
            <h3 class="mb-0"><a href="{{ url('single',$post->id) }}">{{$post->name}}</a></h3>
            @if ($post->Post_status === 'verified')
                <i class="fa fa-check-circle text-success" aria-hidden="true">verified</i>
            @endif
            <span class="location d-inline-block mb-3"><i class="ion-ios-pin mr-2"></i>{{$post->address}}</span>
            <p class="ratings">Rating: {{$post->rating}}({{$post->number_of_ratings}})</p>
        </div>
    </div>
</div>
```



- Along with the verification sign, the rating from the post table is retrieved and shown under each post.

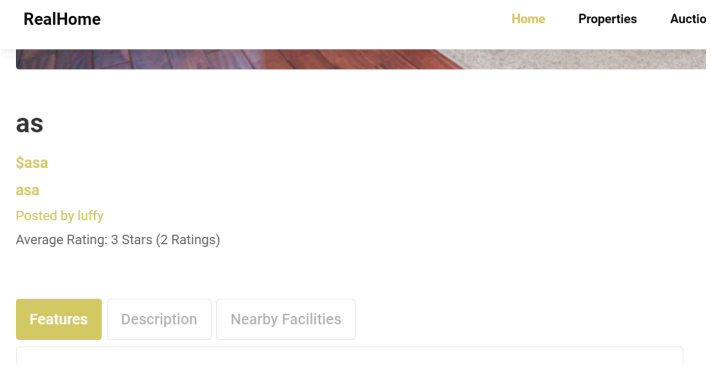
```
        <form method="POST" action="{{ route('store_rating', ['post' => $post->id]) }}">
            @csrf
            <div class="form-group">
                <label for="rating">Rate this property:</label>
                <select class="form-control" id="rating" name="rating">
                    <option value="1">1 star</option>
                    <option value="2">2 stars</option>
                    <option value="3">3 stars</option>
                    <option value="4">4 stars</option>
                    <option value="5">5 stars</option>
                </select>
            </div>
            <button type="submit" class="btn btn-primary">Submit Rating</button>
        </form>
    </div>
    <div class="tab-pane fade" id="pills-manufacturer" role="tabpanel" aria-labelledby="pills-manufac
        <p>{{$post->description}}</p>
    </div>
</div>
```

- This section provides options for the rating system between 1 and 5 for the input.




```
<div class="text">
    <h2>{{$post->name}}</h2>
    <span class="subheading">${{$post->price}}</span>
    <span class="subheading">{{$post->address}}</span>
    <a class="" href="{{ url('history',$post->username) }}">Posted by {{$post->username}}</a>
    <p>Average Rating: {{$post->rating}} Stars ({{$post->number_of_ratings}} Ratings)</p>

    @php
```

- The average rating of a post is also shown when a user clicks for a specific property.

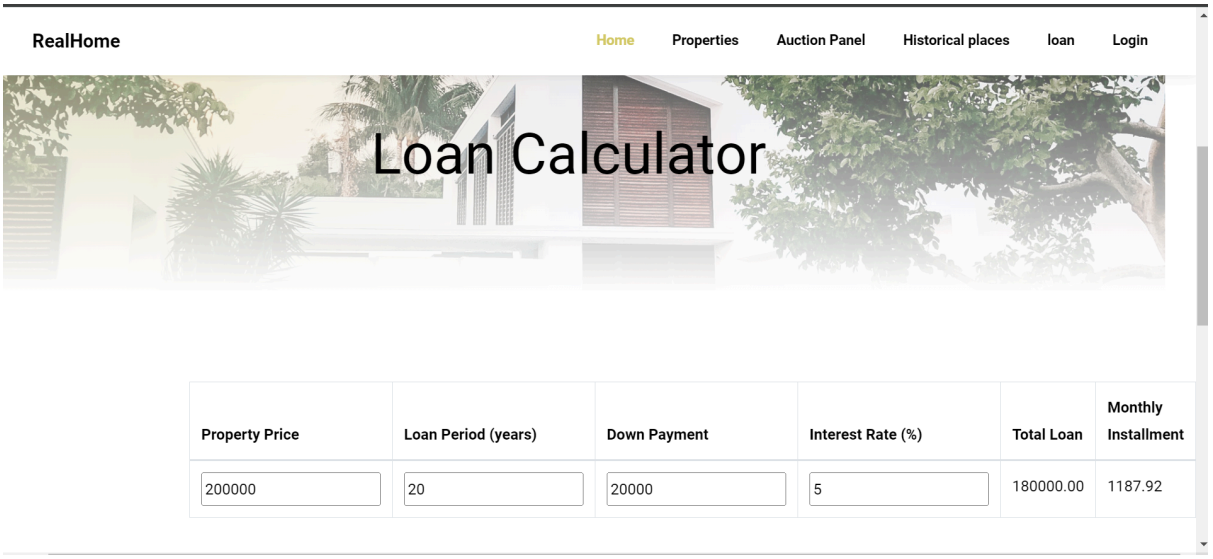


Feature 3: Show the loan calculator

```
@extends('layout')
@section('title','Loan Calculator')
@section('body')
    @include('include.header')
    <section class="hero-wrap hero-wrap-2 ftco-degree-bg js-fullheight" style="background-image: url('home/images/bg_
        <div class="overlay"></div>
        <div class="overlay-2"></div>
        <div class="container">
            <div class="row no-gutters slider-text js-fullheight align-items-end justify-content-center">
                <div class="col-md-9 ftco-animate pb-5 mb-5 text-center">
                    <h1 class="mb-3 bread">Loan Calculator</h1>
                </div>
            </div>
        </div>
    </section>
```

```
<section class="ftco-section">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-8">
                <div class="row">
                    <div class="col-md-12">
                        <table class="table table-bordered">
                            <thead>
                                <tr>
                                    <th>Property Price</th>
                                    <th>Loan Period (years)</th>
                                    <th>Down Payment</th>
                                    <th>Interest Rate (%)</th>
                                    <th>Total Loan</th>
                                    <th>Monthly Installment</th>
                                </tr>
                            </thead>
                            <tbody>
                                <tr>
                                    <td><input type="number" id="propertyPrice" value="200000"></td>
                                    <td><input type="number" id="loanPeriod" value="20"></td>
                                    <td><input type="number" id="downPayment" value="20000"></td>
                                    <td><input type="number" id="interestRate" value="5"></td>
                                    <td id="totalLoan"></td>
                                    <td id="monthlyInstallment"></td>
                                </tr>
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

@include('include.footer')
```

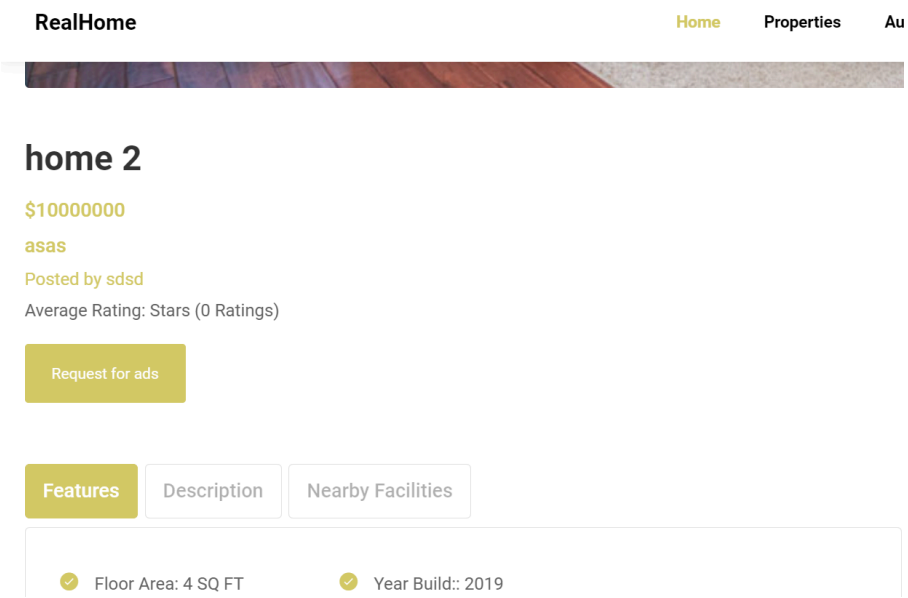
- This creates the interface for calculating loans. The inputs are taken in this tabular format, and the results are displayed on the total loan and the monthly installment part.



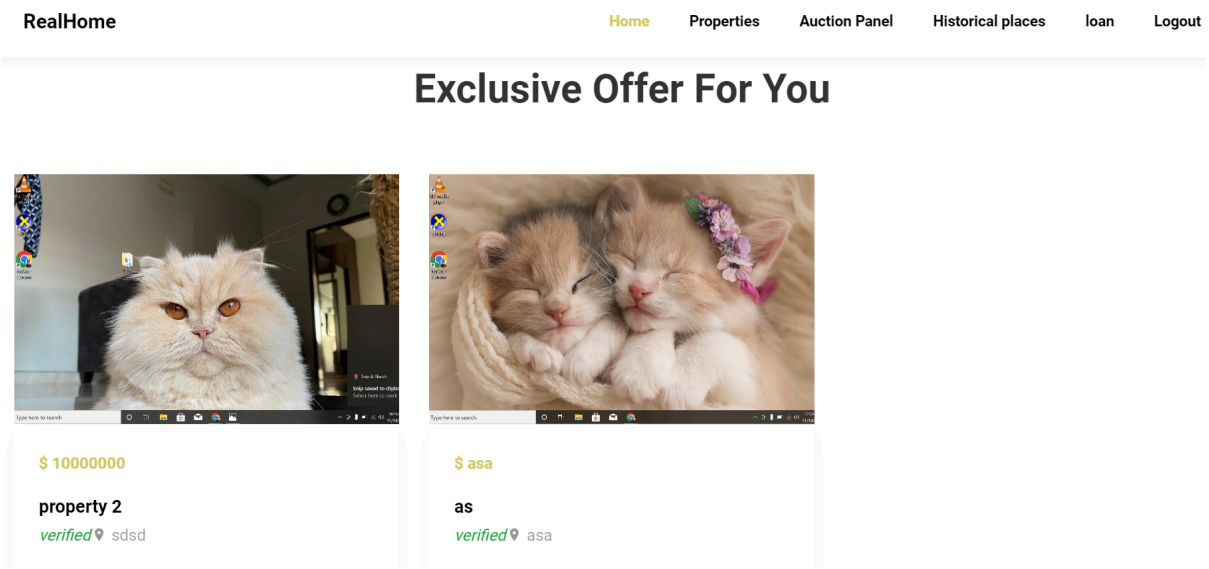
Feature 4 Advertisements

```
            @if ($post->user_ID==$user_id and $post->ads!='Active')
            <p><a href="{{ url('req_ads',$post->id) }}" class="btn btn-primary py-3 px-4">Request for ads</a></
            @endif
        </div>
    </div>
</div>
</div>
<div class="row">
    <div class="col-md-8 pills">
        <div class="bd-example bd-example-tabs">
            <div class="d-flex">
                <ul class="nav nav-pills mb-2" id="pills-tab" role="tablist">
                <li class="nav-item">
                    <a class="nav-link active" id="pills-description-tab" data-toggle="pill" href="#pills-descripti
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="pills-manufacturer-tab" data-toggle="pill" href="#pills-manufacturer" r
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="pills-nearby-tab" role="tab" aria-controls="pills-manufacturer" aria-ex
                </li>
                </ul>
```

- The request for advertisement button only appears when the current logged in user's ID and the ID of the user who posted this matches. This is to ensure no other user can request the advertisements other than the one who posted them.



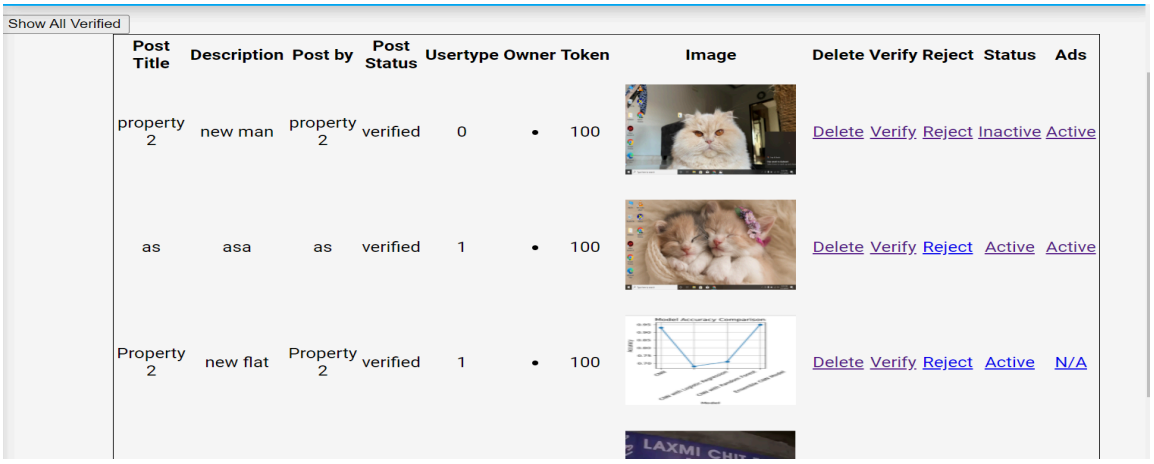

```
@if($post->ads == 'Active')
        <div class="property-wrap ftco-animate">
            <div class="img d-flex align-items-center justify-content-center" style="background-image: url(postima
                <a href="{{ url('single',$post->id) }}" class="icon d-flex align-items-center justify-content-cent
                    <span class="ion-ios-link"></span>
                </a>
                <div class="list-agent d-flex align-items-center">
                    <a href="#" class="agent-info d-flex align-items-center">
                        <div class="img-2 rounded-circle" style="background-image: url(home/images/person_1.jpg);"
                        <h3 class="mb-0 ml-2">Ben Ford</h3>
                    </a>
                    <div class="tooltip-wrap d-flex">
                        <a href="#" class="icon-2 d-flex align-items-center justify-content-center" data-toggle="t
                            <span class="ion-ios-heart"><i class="sr-only">Bookmark</i></span>
```



- If the ad status is active, then the ad will be shown on the welcome page.

```
iv class="page-content">
      <div class="row">
          <div class="col-md-10">
                    <th>Post by</th>
                    <th>Post Status</th>
                    <th>Usertype</th>
                    <th>Owner</th>
                    <th>Token</th>
                    <th>Image</th>
                    <th>Delete</th>
                    <th>Verify</th>
                    <th>Reject</th>
                    <th>Status</th>
                    <th>Ads</th>
                </tr>
  <td>
      <a href="{{ route('ads2', $singlePost->id) }}" class="btn btn-primary">{{ $singlePost->ads}}</a>
  </td>
```

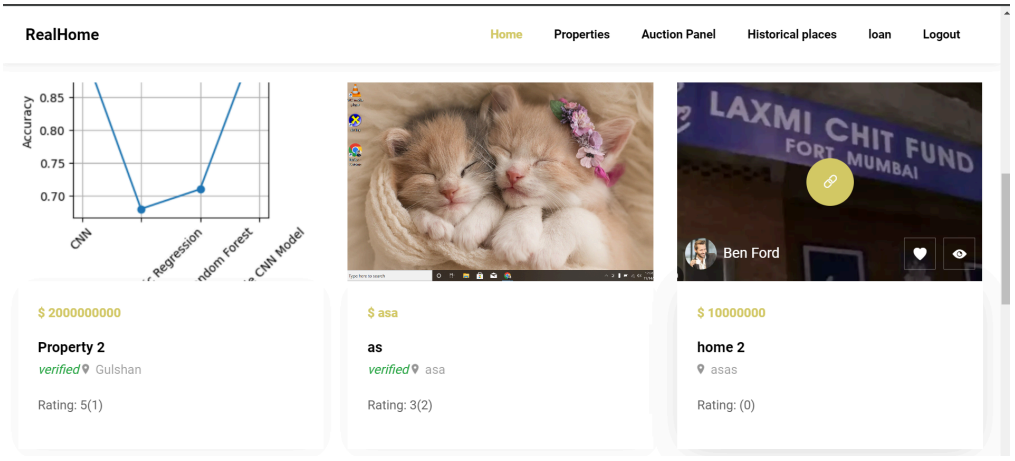
- This displays the ads section of the admin panel, which turns ads from N/A ti active to show the ads in the welcome page.



Feature 5 Leaderboard based on ratings

```
<section class="ftco-section goto-here">
    <div class="container">
        <div class="row">
            @php
                // Sort in descending order
                $sortedPosts = $post->sortByDesc('rating');
            @endphp

            @foreach($sortedPosts as $post)
                @if($post->status == 'Active')
                    <div class="col-md-4">
                        <div class="property-wrap ftco-animate">
                            <div class="img d-flex align-items-center justify-content-ce
                                <a href="{{ url('single',$post->id) }}" class="icon d-fl
                                    <span class="ion-ios-link"></span>
                                </a>
```

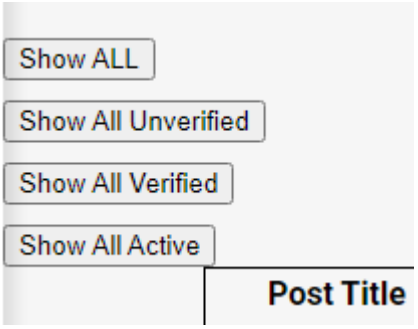
- Here, it sorts the posts in descending order based on the ratings of each.



- Here, we can see that the ones having the most average ratings are shown first.

Contribution of ID : 20201152, Name : Ayan Haider

Feature 1: Admin can filter through verified and unverified posts.



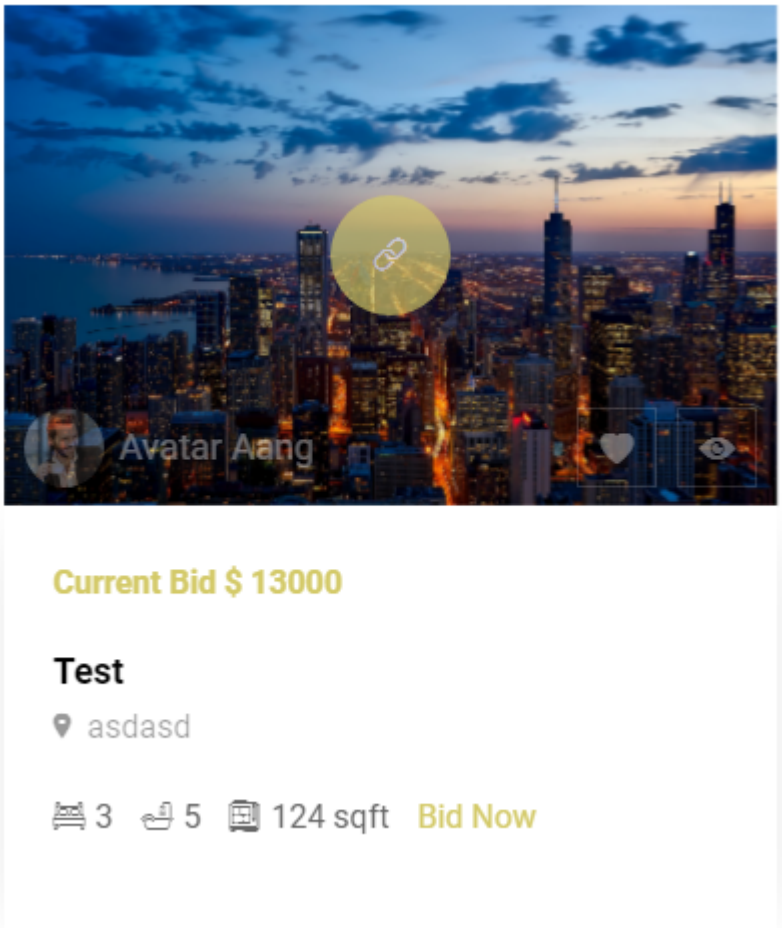

```
<div class="container">
<div class="row">
    <div class="col-md-10">
        <form action="{{url('admin_search_post')}}" method="get">
            @csrf

            <div>
                <input class="btn btn-danger" type="submit" value="Show ALL">
            </div>

            <div style = "padding-top: 10px;">
                <input class="btn btn-danger" type='submit' name='unverified' value="Show All Unverified"></button>
            </div>

            <div style = "padding-top: 10px;">
                <input class="btn btn-danger" type='submit' name='verified' value="Show All Verified"></button>
            </div>

            <div style = "padding-top: 10px;">
                <input class="btn btn-danger" type='submit' name='active' value="Show All Active"></button>
            </div>
    </div>
```

In the “admin.properties” page, a form is present that contains four buttons. Of the buttons the latter three, “unverified”, “verified”, and “active”, have a variable associated with them and if any of latter three are clicked then a non-empty string value is placed in that corresponding variable and the form data is directed to the Admin Controller’s “admin_search_post” function. Additionally, if the Show All button is clicked then all three variables will contain an empty string when the form is sent.

```
@foreach($post as $singlePost)
<tr>
    <td>{{ $singlePost->name }}</td>
    <td>{{ $singlePost->description }}</td>
    <td>{{ $singlePost->name }}</td>
    <td>{{ $singlePost->Post_status }}</td>
    <td>{{ $singlePost->user_type }}</td>
    <td>
        <img class="img_deg" src="postimage/{{ $singlePost->image }}">
    </td>
    <td>
        <a href="{{ url('delete_post', $singlePost->id) }}" class="btn btn-danger">Delete</a>
    </td>
    <td>
        <a href="{{ route('verify_post', $singlePost->id) }}" class="btn btn-outline-secondary">Verify</a>
    </td>
    <td>
        <a href="{{ route('reject_post', $singlePost->id) }}" class="btn btn-primary">Reject</a>
    </td>
    <td>
        <a href="{{ route('status_post', $singlePost->id) }}" class="btn btn-primary">{{ $singlePost->status}}</a>
    </td>

</tr>
@endforeach
```

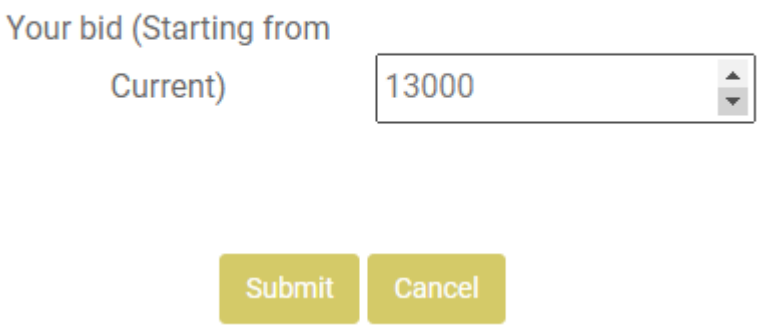
The “admin_search_post” function will return an array of entries from the Posts table which will be displayed to the user, instead of the initial array of entries which are displayed when the “admin.properties” page is first viewed.

Feature 2: Bidding system for buyers, they can bid with a value greater than the last bid.




```
<li><span class="flaticon-bed"></span>{{ $auction->bedroom }}</li>
<li><span class="flaticon-bathtub"></span>{{ $auction->bathroom }}</li>
<li><span class="flaticon-floor-plan"></span>{{ $auction->area }} sqft</li>
<li><a href="{{{ route('bid_auction', $auction->id) }}}" class="btn-buy-now">Bid Now</a>
```

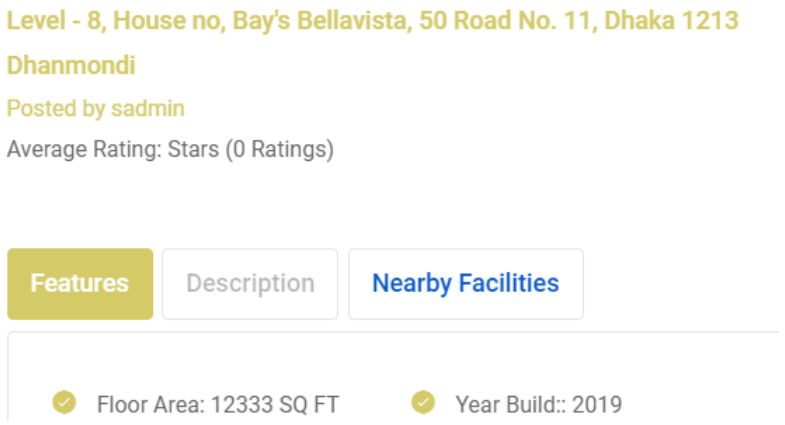
In the “home.panel” page, a Bid Now button is present for every auction that is displayed on the page that when clicked sends the id value of that particular auction to the Home Controller’s “bid_auction” function.




```
<form action="{{ route('bidding', $auction->id) }}" method="POST" enctype="multipart/form-data">
    @csrf
    <div class="div_center">
        <label>Your bid (Starting from Current)</label>
        <input type="number" name="bid" placeholder={{ $auction->base }}
            step="1000" min={{ $auction->base }}
            oninput="validity.valid||(value='');">
    </div>


    <div class="div_center">
        <input type="submit" class="btn btn-primary" value="Submit">

        <input type="submit" class="btn btn-primary" name="cancel" value="Cancel">
    </div>
```

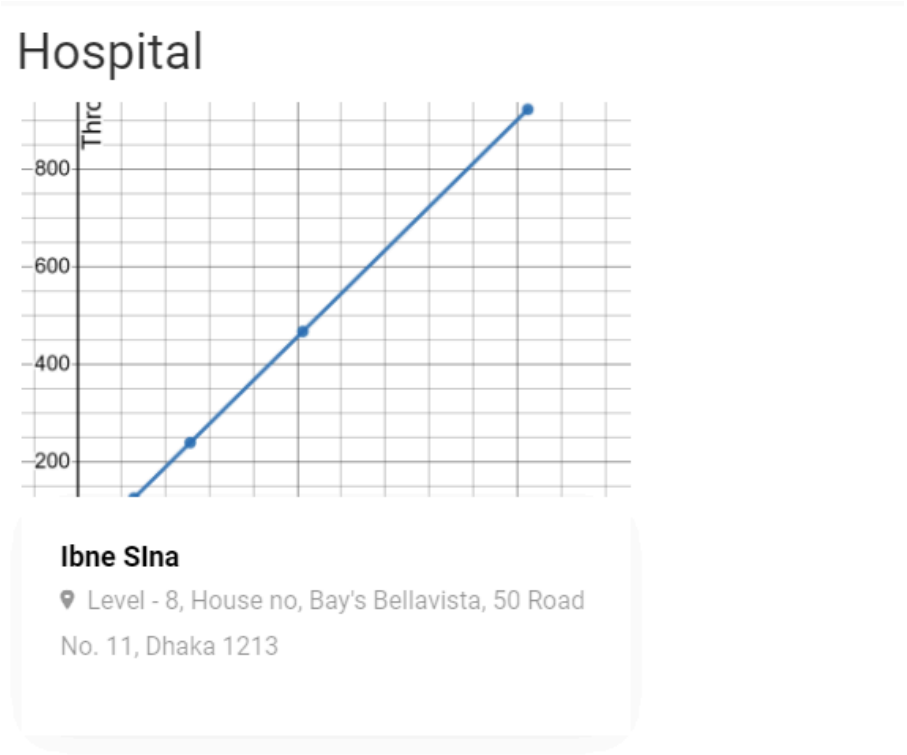
The user is then redirected to “home.bid_auction” page where there is a form where they can either cancel the current bid or bid a value which ensured to be greater than the current “base” value of that particular auction. The form data is then directed to the Home Controller’s “bidding” function.

Feature 3: Users can view nearby shops and malls through location search.



```
</li>
<li class="nav-item">
    <a class="nav-link" id="pills-manufacturer-tab" data-toggle="pill" href="#pills-manufacturer" role="tab" aria-controls="pills-manufacturer" aria-expanded="true">Description</a>
</li>
<li class="nav-item">
    <a class="nav-link" id="pills-nearby-tab" role="tab" aria-controls="pills-manufacturer" aria-expanded="true" href="{{url('nearby_facilities',$post->id)}}">Nearby Facilities</a>
</li>
</ul>
iv>
```

In the “single” page, a Bid Now button is present for the post being displayed. Clicking on the button sends the post’s “id” value to the Home Controller’s “nearby_facilities” function.



```
@foreach($types as $type)
    <div class="row">
        <div class="col-md-12 pills">
            <div class="row">
                    <h2>{{$type-> type}}</h2>
                    @foreach($nearby as $post)
                        @if ($post->type == $type->type)
                            <div class="col-md-4">
                                <div class="property-wrap ftco-animate">
                                    <div class="img d-flex align-items-center justify-content-center" style="background-image: url(nearbyimage/{{$post->image}});">
                                    </div>
                                    <div class="text">
                                        <h3 class="mb-0"><a href="{{ url('singleplace',$post-> id) }}">{{$post-> name}}</a></h3>
                                        <span class="location d-inline-block mb-3"><i class="ion-ios-pin mr-2"></i>{{$post-> address}}</span>
                                    </div>
                                </div>
                            </div>
                        @endif
                    @endforeach
            </div>
        </div>
    </div>
@endforeach
```

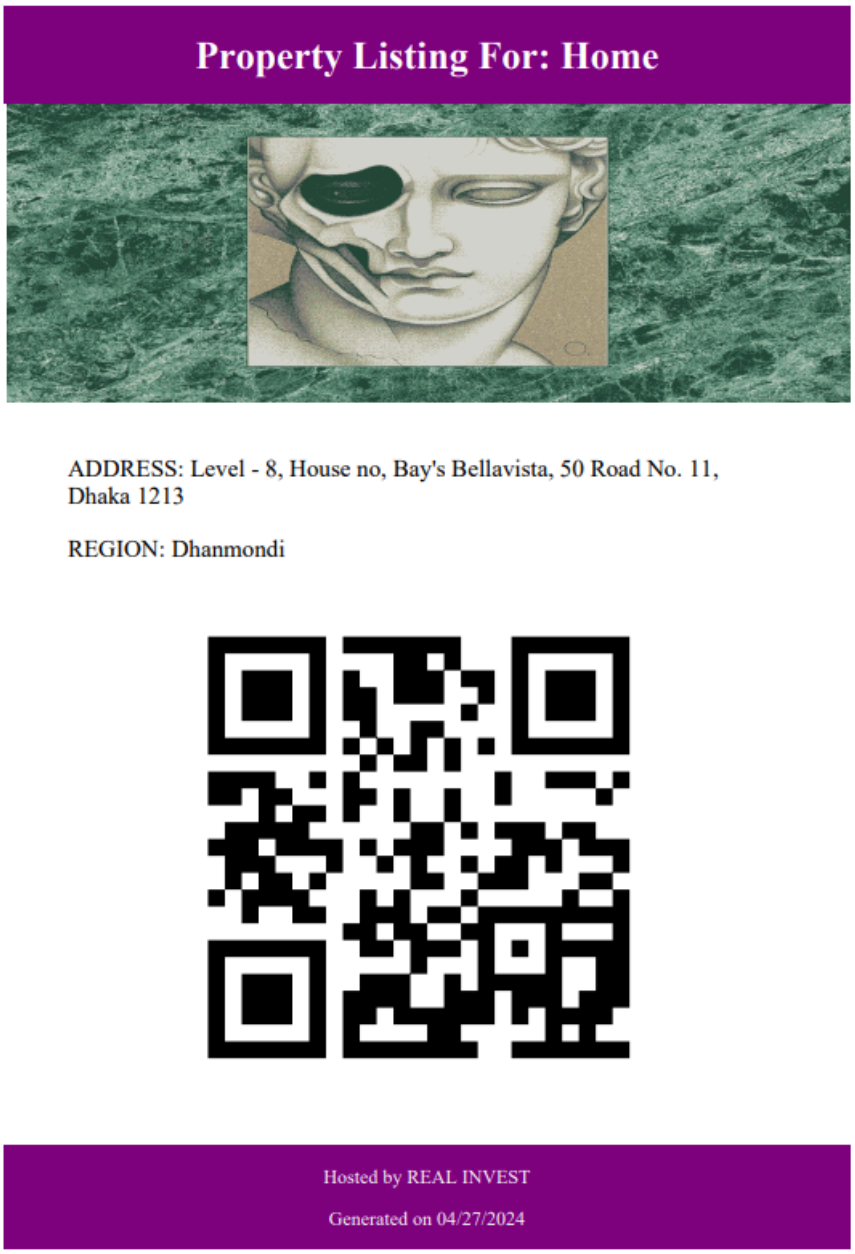
The user is then redirected to the “nearby” page where they will be displayed with entries from the Nearbies table found by the “nearby_facilities” function. Additionally these entries will also be segmented under the different “type” values of the entries.

Feature 4: Users can generate a pdf poster from the QR code of a property.

Generate Poster

```
        <p>{{ QrCode::size(255)->generate("http://127.0.0.1:8000/single/" . $post->id) }}</p>
    </div>
</div>
<div class="col-md-12">
    <p><a style="margin-top: 30px; margin-left: 60px;" href="{{ url('generate-pdf',$post->id) }}" class="btn btn-primary py-3 px-3.5">Generate Poster</a></p>
</div>
```

In the “single” page, a Generate Poster button is present under the QR code of the post being displayed. Clicking on the button sends the post’s “id” value to the PDF Controller’s “generatePDF” function. “admin_search_post” function, which returns a .pdf file.

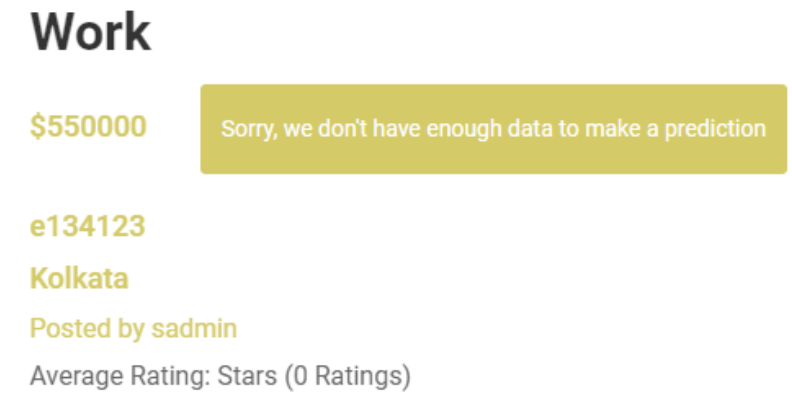


The user will then see a “property_qr.pdf” file being automatically downloaded, which is a poster unique to the particular post being displayed. The user can then either print the pdf or open it on their respective pdf viewing software.

Feature 5: Price prediction for each property using Machine Learning.


```
<div class="row justify">
    <div class="col-md-1" style="margin-left: 0px; margin-top: 10px">
        <span class="subheading">${{$post-> price}}</span>
    </div>
    <div class="col-md-6">
        @if($prediction==null)
            <p><a style="margin-left: 0px;" href="{{ url('predict_price',$post->id) }}" class="btn btn-primary py-3 px-3.5">Estimate Price</a></p>
        @elseif($prediction=="Sorry, we don't have enough data to make a prediction")
            <p><a style="margin-left: 0px;" href="{{ url('predict_price',$post->id) }}" class="btn btn-primary py-3 px-3.5">{{$prediction}}</a></p>
        @else
            <p><a style="margin-left: 0px;" href="{{ url('predict_price',$post->id) }}" class="btn btn-primary py-3 px-3.5">Estimate: ${{$prediction}}</a></p>
        @endif
    </div>
</div>
```

In the “single” page, an Estimate Price button is initially present beside the price of the post being displayed when the page is first viewed. Clicking on the button sends the post’s “id” value to the Predict Controller’s “predict_price” function, which returns either an error message or a “price” value.



The page will then refresh and if an error message was sent by the “predict_price” function, the message ("Sorry, we don't have enough data to make a prediction") will be displayed instead on the Estimate Price button.

However, if an error message is not sent then an estimated price value will be displayed instead on the Estimate Price button.

Backend Development

Contribution of ID : 20201075, Name : Atif Ronan

Feature 1:- Custom Authentication

Registration:- Here at first, using validate we are checking whether the user gave the desired input which are email, name, number and password. After that we save the the input in an array called $data and we also hash the password while making the data so that it is secure. After that we create new user by inserting the $data in our table users in our database.

```
function registrationUser(Request $request){
    $request-> validate([
        'email' => 'required',
        'name' => 'required',
        'number'=> 'required',
        'password' => 'required'
    ]);

    $data['email'] = $request-> email;
    $data['name'] = $request-> name;
    $data['number'] = $request->number;
    $data['password'] = Hash::make($request->password);

    $user= User::create($data);

    if (!$user){
        return redirect(route('registration'))->with("error",'Registration Error!!');
    }
    return redirect(route('login'))->with("success",'Registration Complete!!');
}
```

Login:- Here at first, using validate we are checking whether the user gave the desired input which are email and password. After that we assign the email and password to $user variable. After that, using **Auth::attempt()** method we try to authenticate the user. If it is successful it redirects the user to the route home. And if it fails it redirects the user to the login page with error message “Login Error”.

```php
function loginUser(Request $request){
    $request->validate([
        'email' => 'required',
        'password' => 'required'
    ]);

    $user = $request->only('email','password');
    if (Auth:: attempt($user)){
        return redirect()->intended(route('home'));
    }
    return redirect(route('login'))->with("error",'Login Error');
}
```

Feature 2:- Admin panel:-

Here, after login in using $usertype=Auth::user()->type; we retrieve the currently authenticated user and check users type. If its 1, the user is returned to a view called welcome in our admin folder which is our admin panel. Else it returns to the welcome page of the normal user.

```php
public function welcome(){
    $post=Post::all();
    return view('welcome', compact('post'));
}

public function redirect(){
    $usertype=Auth::user()->type;
    if ($usertype=='1'){
        return view('admin.welcome');
    }
    $post=Post::all();
    return view('welcome', compact('post'));
}
```

Feature 3:- Check Seller History:-

Here, users can check seller history, which is done using the function below. Here, the id of the seller is sent to the function as the parameter $username. Using the line $post = Post::where('username','=',$username)->get(), we find every record in the posts table that has the value $username in the column username of the posts table. After that, we return the view history with a compact post, which is the retrieved data.

```
public function history($username){
    $post = Post::where('username','=',$username)->get();
    return view('history', compact('post'));
}
```

Feature 4:- Users can view nearby Historical infrastructure and its detailed information:-

Add historical places:-

Here, admins can add historical places using the function below. The admin input is caught through the parameter $request. Then we save the inputs in an array called $post and save it. After the we redirect the user with a success message.

```
public function postnearby(Request $request)

{
    $post = new Nearby;
    $post-> name = $request-> title;
    $post -> address = $request -> address;
    $post-> region = $request-> region;
    $post -> type =$request -> type;
    $post -> description =$request -> description;
    $image = $request-> image;
    if($image)
    {

    $imagename =time().'.'.$image -> getClientOriginalExtension();
    $request -> image ->move('nearbyimage',$imagename);
    $post -> image = $imagename;


    }
    $post -> save();

    return redirect()->back()->with('message','Post Added Successfully');
}
```

View detailed historical place:-

The function below is responsible for showing the detailed version of an individual historical place where we find the specific individual place using the $post = Nearby::find($id) and return it.

```php
public function singleplace($id){
    $post = Nearby::find($id);
    return view('singleplace', compact('post'));
}
```

Find houses near a specific historical place:-

Here, through the function below. The address of the historical place is passed through the parameter $address. Using $post=Post::where('address', 'LIKE', '%' . $address . '%')->get(), We perform a Laravel Eloquent Orm query where we try to find all the data where $addresss matches or partially matches with address column in posts table. Then we return the data.

```php
public function specificpost($address){
    $post=Post::where('address', 'LIKE', '%' . $address . '%')->get();
    return view('post', compact('post'));
}
```

Feature 5:- The Users can buy shares of real estate:-

Buy shares of real estate:-

1) The user can entirely buy real estate. Firstly, we find the post in the posts table using $id and save it in $post. Then, the function updates the value Status as Sold, token as 0, and owner as '100-'-$user_ID(indicating that the user owns 100% of the real estate) of the data $post. After that, we save the user data in the $user variable and update the owned column to owned.',.''100-''.id(indicating the user owns 100% of that individual real estate).

```php
public function singleplacebuy($id)
{
    $post = Post::find($id);
    if (!$post) {
        return redirect()->back()->with('error', 'Post not found');
    }

    $post->Status = 'Sold';
    $post->token = '0';
    $user_ID = Auth::user()-> id;
    $post->owner = '100-'.$user_ID;
    $post->save();
    $user =Auth::User();
    $user->owned=$user->owned.','."100-".$id;
    $user-> save();
    return redirect()->back()->with('success', 'Property Bought Successfully');
}
```

2) The user can buy shares of the real estate. Firstly, we find the post in the posts table using $id and save it in $post. After that, we calculate the difference between the current tokens remaining and the user's desired tokens, and we save it in token, then in owner, we save $post->owner.','.($request->percent).'-'.$user_ID (indicating this user owns this much share of the real estate). Then we check to see if the remaining token is 0 or not. If its 0, we set Status as Sold in the $post and save it. After that, we save the user data in the $user variable and update the owned column to $user->owned=$user->owned.','.($request->percent)."-".$id.',' ((indicating this user owns this much share of the real estate)

```php
public function share($id,Request $request)
{
    //dd($request->percent);

    $post = Post::find($id);
    if (!$post) {
        return redirect()->back()->with('error', 'Post not found');
    }
    $token =(int)($post->token)-(int)($request->percent);
    if ($token<0){
        return redirect()->back()->with('failure', 'percentage out of index');
    } else {
        $post->token = $token;
        $user_ID = Auth::user()-> id;
        $post->owner = $post->owner.','.($request->percent).'-'.$user_ID;
        if ($post->token==0){
            $post->Status = 'Sold';
        }
        $post->save();
        $user =Auth::User();
        $user->owned=$user->owned.','.($request->percent)."-".$id.',';
        $user-> save();
    }
    return redirect()->back()->with('success', 'Share Bought Successfully');
}
```

Contribution of ID : 20201069, Name : Nafiun Al Amin

Feature 1: Verifying Properties

- In this feature, administrators can verify a post from the admin panel. The code for the admin controller is given below.

```
    public function verify_post($id)
{
    $post = Post::find($id);
    if (!$post) {
        return redirect()->back()->with('error', 'Post not found');
    }

    $post->post_status = 'verified';
    $post->save();

    return redirect()->back()->with('success', 'Post verified successfully');
}
```

- This function looks into the post table to find the post using "find($id)." Then, when the post is found, it changes the post_status section of the Post table to "verified."

Feature 2: Advertisement system

The Home and the Admin Controller codes are given below.

```
public function ads2($id)
{
    $post = Post::find($id);
    if (!$post) {
        return redirect()->back()->with('error', 'Post not found');
    }
    if ($post->ads=='Active')
    {
        $post->ads = 'N/A';
        $post->save();

        return redirect()->back()->with('success', 'Ads inactive');
    } else{
        $post->ads = 'Active';
        $post->save();

        return redirect()->back()->with('success', 'Ads active');
    }
}
```

- This function allows the admin to make the advertisements active or inactive. If the post is inactive, then the function will change it to N/A and if the post is active, then the function will change it to active.

```php
public function req_ads($id)
{
    $post = Post::find($id);
    if (!$post) {
        return redirect()->back()->with('error', 'Post not found');
    }
    if ($post->ads!='Pending')
    {
        $post->ads = 'Pending';
        $post->save();

        return redirect()->back()->with('success', 'Ads inactive');
    }
    return redirect()->back();
}
```

- This code is from the home controller. Here, users can request for ads and the function changes the ad status to “pending”, which is later activated or rejected by the admin in the admin panel.

Feature 3: Loan Calculator

```javascript
function calculate() {
    var propertyPrice = parseFloat(document.getElementById('propertyPrice').value);
    var loanPeriod = parseInt(document.getElementById('loanPeriod').value);
    var downPayment = parseFloat(document.getElementById('downPayment').value);
    var interestRate = parseFloat(document.getElementById('interestRate').value);

    var totalLoan = propertyPrice - downPayment;
    document.getElementById('totalLoan').innerHTML = totalLoan.toFixed(2);

    var monthlyInterestRate = (interestRate / 100) / 12;
    var numberOfPayments = loanPeriod * 12;

    var monthlyInstallment = (totalLoan * monthlyInterestRate) / (1 - Math.pow(1 + monthlyInterestRate,
    document.getElementById('monthlyInstallment').innerHTML = monthlyInstallment.toFixed(2);
}
```

This function uses several variables and uses those variables for calculating the total loan, monthly interest rate and the number of payments. After that, the monthly installment amount is calculated using those variables.

Feature 4: LeaderBoard Based on Ratings

```php
public function post(){
    $post=Post::all();
    return view('post', compact('post'));
}
```

The post function comes from the home controller and retrieves all the features from the post table, which includes the ratings section.

```
    @php
        // Sort in descending order
        $sortedPosts = $post->sortByDesc('rating');
    @endphp
```

Later on, the posts are simply sorted in descending order based on the ratings to provide a leaderboard like structure.

Feature 5: Rating system

```php
public function rate(Request $request, Post $post)
    {

        $validatedData = $request->validate([
            'rating' => 'required|integer|between:1,5',
        ]);


        $currentRating = $post->rating ?? 0;
        $numberOfRatings = $post->number_of_ratings ?? 0;


        $newNumberOfRatings = $numberOfRatings + 1;
        $newRating = ($currentRating * $numberOfRatings + $validatedData['rating']) / $newNumberOfRatings;


        $post->rating = $newRating;
        $post->number_of_ratings = $newNumberOfRatings;
        $post->save();


        return redirect()->back()->with('success', 'Rating submitted successfully.');
    }
```

The function first checks if the ratings given are between 1 and 5. The current rating, if there are no ratings, will be set to 0. Then, the function will also take in the number of ratings by increasing its value by 1 each time someone rates. Eventually, using the old rating and the new rating and the number of ratings, the average rating is calculated, and it is then stored in the rating section of the post table.

Contribution of ID : 20201104, Name : Nusaiba Zaman

Feature 1: Sellers are able to post pictures and descriptions of a property.

```
    public function poster(Request $request)

    {

        $name = Auth::user()-> name;
        $user_type = Auth::user()->type;
        $user_ID = Auth::user()-> id;
        //dd($user_ID);
        $post = new Post;
        $post-> name = $request-> title;
        $post -> address = $request -> address;
        $post-> bedroom = $request-> bedroom;
        $post -> bathroom = $request -> bathroom;
        $post-> garage = $request-> garage;
        $post -> stories = $request -> stories;
        $post -> area=$request->area;
        $post -> description=$request-> description;
        $post -> price=$request-> price;
        $post -> number=Auth::user()->number;
        $post -> Post_status = 'active';
        $post -> username = $name;
        $post -> user_ID = $user_ID;
        $post -> user_type = $user_type;

        $image = $request-> image;
    if($image)
    {

    $imagename =time().'.'.$image -> getClientOriginalExtension();
    $request -> image ->move('postimage',$imagename);
    $post -> image = $imagename;


    }


    $post -> save();

     return redirect()->back()->with('message','Post Added Successfully');
}
```

This function allows users to post properties for sale. Inputs are taken into several columns of the poster table. The property name, address, bedroom, bathroom, garage, stories, area, description and price along with the image are taken as input from the user.

Feature 2: Each property will have their own comment section.

```
public function comment_add($id, Request $request)
{
    $request->validate([
        'content' => 'required|string',
    ]);


    $comment = new comments();
    $comment->name = Auth::user()->name;
    $comment->content = $request->content;
    $comment->post_id=$id;
    $comment->user_id = Auth::id();

    $comment->save();

    return redirect()->back()->with('success', 'Comment added successfully.');
}
```

There are comment sections under each property posted. The comment_add function first checks whether the comment is a string and then new comments are added to the table. The user name, post status and user id information are taken from the user that is logged in. The content is provided in the comment section by user.

Feature 3: Users can like or dislike a comment for each property.

```
public function like($id)
{
    $comment = comments::findOrFail($id);
    $comment->increment('likes');
    return redirect()->back()->with('success', 'Comment liked successfully');
}

public function dislike($id)
{
    $comment = comments::findOrFail($id);
    $comment->increment('dislikes');
    return redirect()->back()->with('success', 'Comment disliked successfully');
}
```

```
    public function likeOrDislike(Request $request, $id)
{
    $action = $request->input('action');
    $comment = comments::find($id);


    if ($action == 'like' && $request->session()->get("disliked_$id")) {
        $comment->dislikes--;
        $request->session()->forget("disliked_$id");
    } elseif ($action == 'dislike' && $request->session()->get("liked_$id")) {
        $comment->likes--;
        $request->session()->forget("liked_$id");
    }

    if ($action == 'like') {
        $comment->likes++;
        $request->session()->put("liked_$id", true);
    } elseif ($action == 'dislike') {
        $comment->dislikes++;
        $request->session()->put("disliked_$id", true);
    }

    $comment->save();

    return redirect()->back();
    }
```

The functions, like, dislike and likeOrDislike are needed in the comment section of each property post, to add likes and dislikes on comments. The like() function increments the number of likes for a particular comment on the comment table. The dislike() function increments the number of dislikes in the comment table for a particular post. The likeOrDislike() function checks if the input is like and the previous input is dislike, if its true, it will forget the dislike and increment the like section in the comment table. If the input is dislike and the previous input is like, then the function forgets the dislike and increments like.

Feature 4: Seller can posts in Auction Panel and Users can view

```
    public function added_auction(Request $request)

    {
        $name = Auth::user()-> name;
        $user_type = Auth::user()->type;
        $user_ID = Auth::user()-> id;
        //dd($user_ID);
        $auction = new Auction;
        $auction-> name = $request-> title;
        $auction -> address = $request -> address;
        $auction-> bedroom = $request-> bedroom;
        $auction -> bathroom = $request -> bathroom;
        $auction-> garage = $request-> garage;
        $auction -> stories = $request -> stories;
        $auction -> area=$request->area;
        $auction -> description=$request-> description;
        $auction -> price=$request-> price;
        $auction -> number=Auth::user()->number;
        $auction -> base=$request-> base;
        $auction -> Post_status = 'active';
        $auction -> user_ID = $user_ID;
        $auction -> user_type = $user_type;
        $image = $request-> image;
    if($image)
    {
    $imagename =time().'.'.$image -> getClientOriginalExtension();
    $request -> image ->move('imager',$imagename);
    $auction -> image = $imagename;
    }
    $auction -> save();

     return redirect()->back()->with('message','Post Added Successfully');
}
```

This function allows the user to post property information for auctioning. The inputs taken are name of property, address, bedroom, bathroom, garage, stories, area, description, price, image and base price. These inputs are taken into columns in the added_auction table.

Feature 5: Property post in Admin Panel with delete, status button, Auction Posts in Admin Panel

```
@foreach($auction as $singlePost)
<tr>

    <td>{{ $singlePost->description }}</td>
    <td>{{ $singlePost->name }}</td>




    <td>{{ $singlePost->Post_status }}</td>
    <td>{{ $singlePost->user_type }}</td>
    <td>{{ $singlePost->base }}</td>
    <td>{{ $singlePost->price }}</td>
```

The data is being retrieved from the auction table.

```
    public function delete_post($id)
{
    $post = Post::find($id);
    if (!$post) {
        return redirect()->back()->with('error', 'Post not found');
    }

    $post->delete();

    return redirect()->back()->with('success', 'Post deleted successfully');
}
```

The delete_post function deletes the post from the post table by taking in the id.

```php
    public function status_post($id)
{
    $post = Post::find($id);
    if (!$post) {
        return redirect()->back()->with('error', 'Post not found');
    }
    if ($post->status=='Active')
    {
        $post->status = 'Inactive';
        $post->save();

        return redirect()->back()->with('success', 'Post inactive');
    } else{
        $post->status = 'Active';
        $post->save();

        return redirect()->back()->with('success', 'Post active');
    }
}
```

Th status_post function checks the status of the post and changes it to inactive if the post status is already active or active if the post status is inactive.

Contribution of ID : 20201152, Name : Ayan Haider

Feature 1: Admin can filter through verified and unverified posts.

```
public function admin_search_post(Request $request)
{
    $message="";

    if ($request->verified != "")
    {
       $post=DB::table('posts')->where('Post_status','=','verified')->get();
    }

    else if ($request->unverified != "")
    {
       $post=DB::table('posts')->where('Post_status','!=','verified')->get();
    }

    else if ($request->active != "")
    {
       $post=DB::table('posts')->where('Post_status','=','active')->get();
    }

    else
    {
        $post = Post::all();

        return view('admin.properties',compact('post'));
    }

    if (count($post) == 0)
    {
        $message = "NO MATCHES FOUND";
    }

    return view('admin.properties',compact('post','message'));
```

In the Admin Controller, based on whichever variable of the Request ("verified", "unverified", "active") contains a non-empty string, in this function the Posts table is searched to find all the posts which have a Post_status of "verified", which dont have a Post_status of "verified", or which have a Post_status of "active", respectively. Additionally, if all of the three variables of Request contain an empty string then all the entries of the Posts table are fetched. This array of posts are then returned to the "admin.properties" page.

Feature 2: Bidding system for buyers, they can bid with a value greater than the last bid.

```
public function bid_auction($id)
{
    $auction_ID = $id;

    $auction = DB::table('auctions')->where('id','=',$auction_ID)->get();
    return view('home.bid_auction',compact('auction'));
}
```

In the Home Controller, based on the $id of an auction, in this function the Auctions table is searched to find the auction with that particular id and it is sent to the "home.bid_auction" page.

```
public function bidding(Request $request, $id)
{
    $user_id = Auth::user()-> id;
    $auction_ID = $id;
    $cancel = $request->cancel;

    $bid = $request->bid;

    if($cancel == "Cancel")
    {
        $auction = Auction::all();
        return view('home.panel',compact('auction'));
    }

    else
    {
        $auction = Auction::find($auction_ID);

        $auction->base = $bid;
        $auction->bidder_id = $user_id;
        $auction->save();

        $auction = Auction::all();
        return view('home.panel',compact('auction'));
    }

}
```

Additionally, in this function, if the $cancel variable contains a “Cancel” string, the page is redirected to “home.panel” to cancel the bidding. If not, then the Auctions table is searched to find the auction containing $id. For that particular auction, the values of “base” and “bidder_id” are changed to the value of the currently “authenticated user’s id” and the value of “bid” in Request respectively. The page is then redirected to “home.panel”.

Feature 3: Users can view nearby shops and malls through location search.

```php
public function nearby_facilities($id){
    $post = Post::where('id' , '=' , $id)->get();
    $address= $post[0]->address;
    $region= $post[0]->address;

    $nearby=Nearby::where('address', 'LIKE', '%' . $address . '%')
    ->orWhere('region','=',$region)
    ->where('type','!=','industry')
    ->get();

    $types=Nearby::distinct()
        ->where('address', 'LIKE', '%' . $address . '%')
    ->orWhere('region','=',$region)
        ->where('type','!=','industry')->get('type');


    return view('nearby', compact('nearby','types'));
}
```

In the Home Controller, based on the $id of a particular post, in this function the Posts table is searched to find the property with that particular id. From this property, the values of "address" and "region" are obtained and used to search the Nearbies table for entries which contain either the same "address" or same "region". Additionally, a list of the "type" values of these entries are also obtained. These two are then sent to the "nearby" page.

Feature 4: Users can generate a pdf poster from the QR code of a property.

```php
class PDFController extends Controller
{
    public function generatePDF($id){
        $post = Post::where('id' , '=' , $id)->get();


        $img = QrCode::size(255)->generate("http://127.0.0.1:8000/single/" . $id);



        $data = [
            'post' => $post,
            'date' => date('m/d/Y'),
            'site' => "REAL INVEST",
            'img' => $img
        ];

        $pdf = PDF::loadView('pdf.generatePoster', $data);
        return $pdf->download('property_qr.pdf');
    }
}
```

In the PDF Controller, based on the $id of a particular post, in this function the Posts table is searched to find the property with that particular id. From this property, a QR code image is

generated using the QrCode function. This image along with other data, including the found post, the date, and the website name, are sent to the “pdf.generatePoster” page.

```
<body>
    @foreach($post as $post)

    <div class="header">
        <h1 font-size: 28;"> Property Listing For: {{ $post->name }} </h1>
    </div>

    <br/>
    <br/>
    <br/>
    <br/>

    <table>
        <tbody>

            <tr>
                <td>
                    <img class="img_deg" src="postimage/{{ $post->image }}" style="width:700px; height:250px;">
                </td>
            </tr>
        </tbody>
    </table>

    <br/>
    <p style="font-size: 15; margin-left: 40; margin-right: 40;"> ADDRESS: {{ $post->address }} </p>
    <p style="font-size: 15; margin-left: 40; margin-right: 40;"> REGION: {{ $post->region }} </p>
    <br/>
    <br/>

    <table>
        <tbody>

            <tr>
                <td>
                    <img src=" {!! base64_encode($img) !!}" width="350" height="350" class="center2">
                </td>
            </tr>

        </tbody>



        <div class="footer">
          <p>Hosted by {{$site}}</p>
          <p>Generated on {{ $date }}</p>
        </div>

    </table>

    @endforeach
```

In the “pdf.generatePoster” page an HTML document is created using the received data to make a document including details of the post such as “name”, “address”, “region”, QR-code image, as well as the current date and name of website. This HTML doc is then sent back to the PDF Controller function where it is converted to a pdf. This pdf is redirected to the “single” page where it is automatically downloaded into the user’s device.

Feature 5: Price prediction for each property using Machine Learning.

```php
class PredictController extends Controller
{
    public function predict_price($id){
        $post = Post::where('id' , '=' , $id)->get();

        $data = [
            'No. Beds' => $post[0]->bedroom,
            'No. Baths' => $post[0]->bathroom,
            'Area' => $post[0]->area,
            'Region' => $post[0]->region

        ];

//          echo($post[0]->address);

        $prediction =  Http::post('http://127.0.0.1:5000/predictprice', $data);
        $post = Post::find($id);

        if($prediction->successful())
        {
            return view('single', compact('post','prediction'));
        }
        else
        {
            $prediction = "Sorry, we don't have enough data to make a prediction";
            return view('single', compact('post','prediction'));
        }

    }
}
```

In the Predict Controller, based on the $id of a particular post, in this function the Posts table is searched to find the property with that particular id. From this property a variable $data, including the post's "bedroom", "bathroom", "area", and "region" values, are sent to the Predict Price api which will need to be running in the "app.py" file.

```python
import ...

app = Flask(__name__)

# Load the pre-trained model
model_file_path = 'model.pkl'
try:
    model = joblib.load(model_file_path)
except Exception as e:
    print(f"An error occurred while loading the model: {e}")
    model = None

# Load label encoders
label_encoder_region = LabelEncoder()
# label_encoder_gender = LabelEncoder()
# label_encoder_sleep_disorder = LabelEncoder()
# label_encoder_occupation = LabelEncoder()  # Add label encoder for Occupation

# Load the dataset (for label encoder fitting)
file_path = 'property_prices.csv'
df = pd.read_csv(file_path)

# Fit label encoders
label_encoder_region.fit(df['Region'])


# Endpoint for prediction
@app.route('/predictprice', methods=['POST'])
def predict_sleep():
    if model is None:
        return jsonify({'error': 'Model not loaded!'}), 500

    data = request.json
    app.logger.info("Received data: %s", data)

    # Check if all required fields are present in the input data
    # required_fields = ['Age', 'Gender', 'DailySteps', 'Occupation']

    required_fields = ['No. Beds', 'No. Baths', 'Area', 'Region']
```

```
    if not all(field in data for field in required_fields):
        return jsonify({'error': 'Missing required fields'}), 400

    try:
        # Prepare input data for prediction
        # age = float(data[''])
        # gender = data['Gender']

        beds = float(data['No. Beds'])
        baths = float(data['No. Baths'])
        area = float(data['Area'])

        region = data['Region']

        app.logger.info("Received input: Beds=%s, Baths=%s, Area=%s, Region=%s", beds, baths, area, region)

        # Convert categorical variables to numerical using label encoding
        # gender_encoded = label_encoder_gender.transform([gender])[0]
        region_encoded = label_encoder_region.transform([region])[0]

        app.logger.info("Encoded Region %s", region_encoded)

        # Make predictions for new data
        prediction = model.predict([[beds, baths, area, region_encoded]])

        app.logger.info("Prediction: %s", prediction)

        # Mapping numeric labels to categories
        # predicted_category = label_encoder_sleep_disorder.inverse_transform(prediction)

        return str(int(round(prediction[0])))

    except Exception as e:
        app.logger.error("Prediction error: %s", e)
        return jsonify({'error': f'Prediction error: {e}'}), 400

# Route for the root URL
@app.route('/')
def index():
```

In the Predict Price api, the $data values are used as input for a trained RandomForestRegressor model, which is stored in a .pkl file, which uses the post's "bedroom", "bathroom", "area", and "region" values to try and predict a price value, after performing some data preprocessing. This prediction is sent back to the Predict Controller function where the found post and either the successfully predicted price value or a failure message is sent back to the "single" page

```
import ...

# Load the dataset from the CSV file
file_path = 'property_prices.csv'
df = pd.read_csv(file_path)

# Select relevant columns including the "Occupation" column
selected_columns = ['No. Beds', 'No. Baths', 'Area', 'Region', 'Price']
df = df[selected_columns]

# Drop rows with missing values in the selected columns
df.dropna(subset=selected_columns, inplace=True)


label_encoder_region = LabelEncoder()
# Convert categorical variables to numerical using label encoding
label_encoder_region.fit(df['Region'])
# label_encoder_sleep_disorder = LabelEncoder()
# label_encoder_occupation = LabelEncoder()  # Add label encoder for Occupation

df['Region'] = label_encoder_region.fit_transform(df['Region'])
# df['Sleep Disorder'] = label_encoder_sleep_disorder.fit_transform(df['Sleep Disorder'])
# df['Occupation'] = label_encoder_occupation.fit_transform(df['Occupation'])  # Encode Occupation

# Split the data into features (X) and target variable (y)
X = df[['No. Beds', 'No. Baths', 'Area', 'Region']]
y = df['Price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a RandomForestClassifier model
model = RandomForestRegressor()
# model = RandomForestClassifier()

# Train the model
model.fit(X_train, y_train)

# Print message indicating that the model is trained
print('Model trained successfully!')
```

```python
# Evaluate the model on the test set
y_pred = model.predict(X_test)

# Display accuracy and classification report
r2 = model.score(X_test, y_test)
print(f': {r2:.2f}')


# Save the trained model
model_file_path = "model.pkl"
full_model_path = os.path.join(os.getcwd(), model_file_path)
try:
    joblib.dump(model, full_model_path)
    print("Model saved successfully as 'model.pkl'")
except Exception as e:
    print(f"An error occurred while saving the model: {e}")
```

Additionally, in order to retrain the model the file “prcode.py” can be run where data (“bedroom”, “bathroom”, “area”, “region”, and “price” values) from a sepecified external dataset (we used “property_prices.csv) will be extracted and preprocessed to train the RandomForestRegressor model which will then be evaluated to display the r2-Score (0.86 for RandomForestRegressor) and store the trained model on the specified .pkl file (we used “model.pkl).

Source Code Repository

https://github.com/NafiunAlAmin/RealHome-CSE471-Project-
Please look into the branches for the latest iterations.

https://drive.google.com/drive/folders/1JCi2P8LLTj4s5hhNIxTDtt3KdlFe-TWV?usp=sharing
Additional files and dataset are required for Feature 6 of Module 4 (too large to upload to Github)

Conclusion

The real estate market in Bangladesh is thriving thanks to growing urbanization, population expansion, initiatives by the government, and a variety of other reasons. Which is why RealHome will flourish in this industry due to it being user friendly interface, dynamic and has resourceful features, opportunity for investments and growth for buyers and sellers, and many more. RealHome is a solution-oriented platform, and it will revolutionize the future industry of Real Estate.

References

Barua, S., Mridha, A.H.A.M., & Khan, R. H. (2010, 1). Housing, real estate sector in Bangladesh present status and policies implications. *ASA University Review*, *4*, 239-253. https://www.researchgate.net/publication/286181851_Housing_real_estate_sector_in_Bangladesh_present_status_and_policies_implications

Unity Coding. (2024, 3). Deploying Machine Learning Models in Laravel: A Step-by-Step Guide. *Unity Coding Youtube.* https://www.youtube.com/watch?v=jgz8lr32brY&t=66s

Jahan S.H. (2023). Bangladesh Property Price Data. *Kaggle.* https://www.kaggle.com/datasets/shafayathossainjahan/bangladesh-property-pricces