

## Authentication page

Login

Name:

Ashwin

Phone number:

9826738812

Otp:

9835

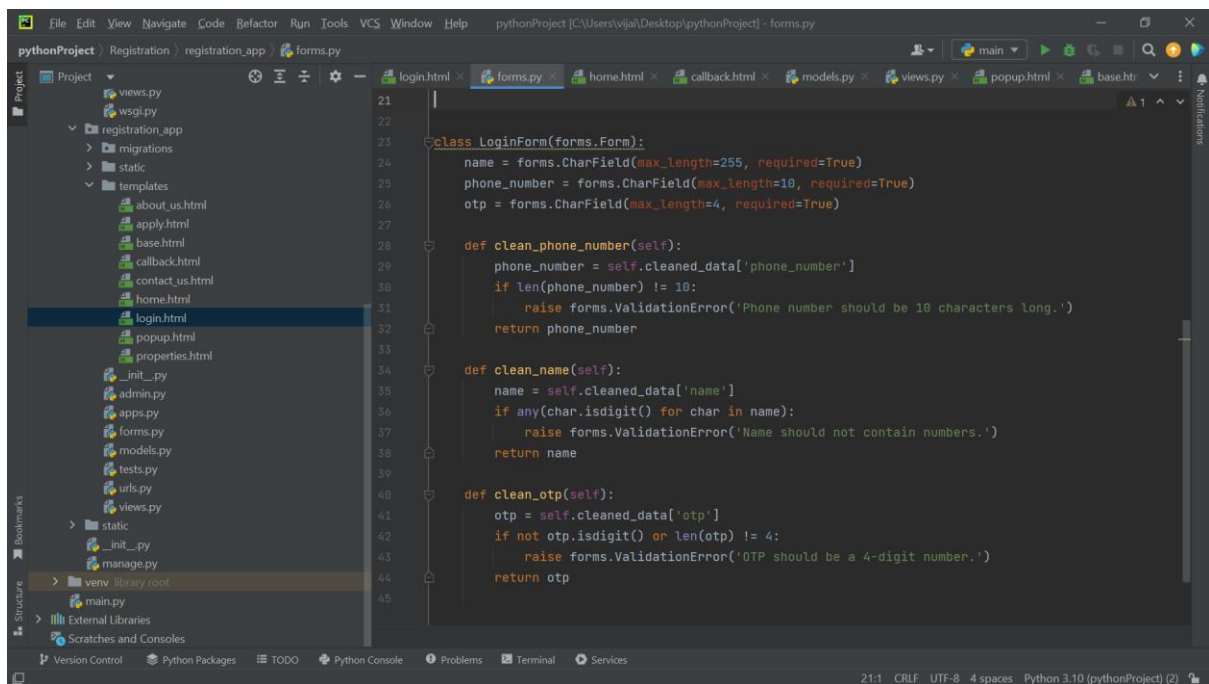
Enter

OTP sent to your mobile number.

Admin Login

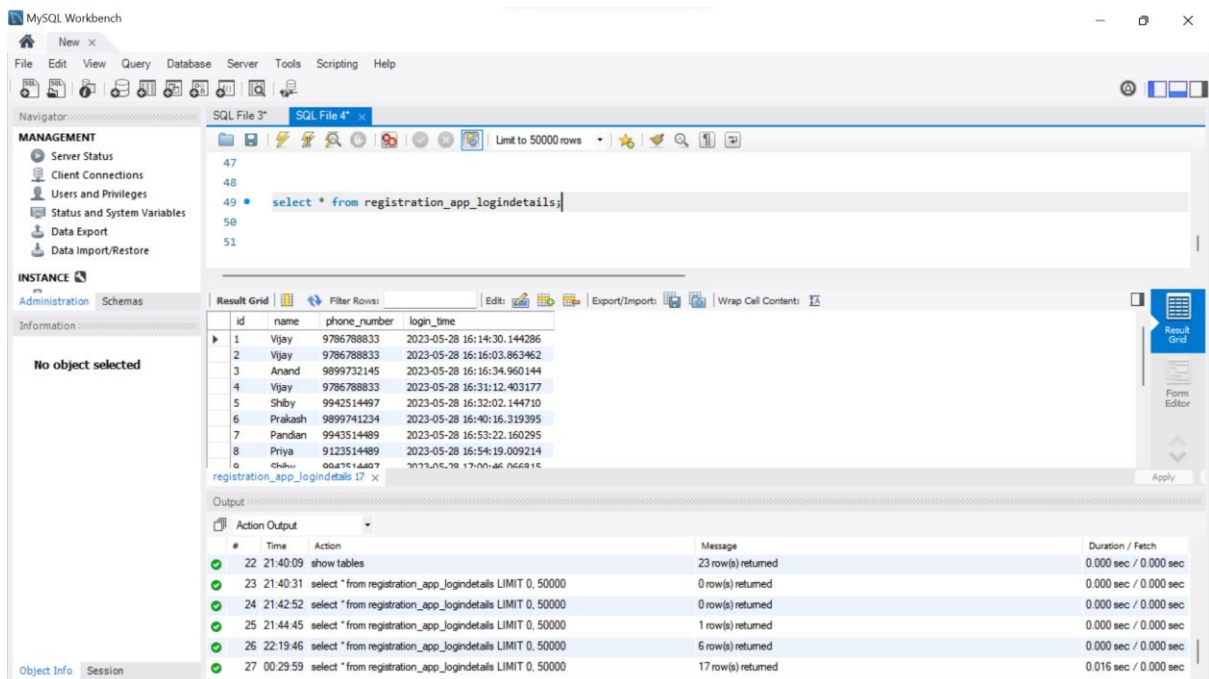
```
129 <body>
130 <div class="gradient-background"></div>
131 <div class="container">
132 <div class="title-container">
133 <div class="heading">
134 Vijay Realty
135 </div>
136 <p class="slogan">Search. See. Love.</p>
137 </div>
138 <div class="login-container">
139 <h2>Login</h2>
140 {% if error_message %}
141 <div class="alert alert-danger">{{ error_message }}</div>
142 {% endif %}
143 {% if success_message %}
144 <div class="alert alert-success">{{ success_message }}</div>
145 {% endif %}
146 <form method="POST" action="{% url 'login' %}">
147 <input type="text" value="{% csrf_token %}" />
148 <input type="text" value="{{ form }}" />
149 <button type="submit">Enter</button>
150 </form>
151 <p class="otp-sent-message"></p>
152 <div class="admin-login">
153 <a href="#" class="btn btn-link">Admin Login</a>
```

## Forms.py



```
21 |
22 |
23 | class LoginForm(forms.Form):
24 |     name = forms.CharField(max_length=255, required=True)
25 |     phone_number = forms.CharField(max_length=10, required=True)
26 |     otp = forms.CharField(max_length=4, required=True)
27 |
28 |     def clean_phone_number(self):
29 |         phone_number = self.cleaned_data['phone_number']
30 |         if len(phone_number) != 10:
31 |             raise forms.ValidationError('Phone number should be 10 characters long.')
32 |         return phone_number
33 |
34 |     def clean_name(self):
35 |         name = self.cleaned_data['name']
36 |         if any(char.isdigit() for char in name):
37 |             raise forms.ValidationError('Name should not contain numbers.')
38 |         return name
39 |
40 |     def clean_otp(self):
41 |         otp = self.cleaned_data['otp']
42 |         if not otp.isdigit() or len(otp) != 4:
43 |             raise forms.ValidationError('OTP should be a 4-digit number.')
44 |         return otp
45 |
```

Every login will be updated in our DB with Date and time



MySQL Workbench

SQL File 3\* SQL File 4\*

Limit to 50000 rows

select \* from registration\_app\_logindetails;

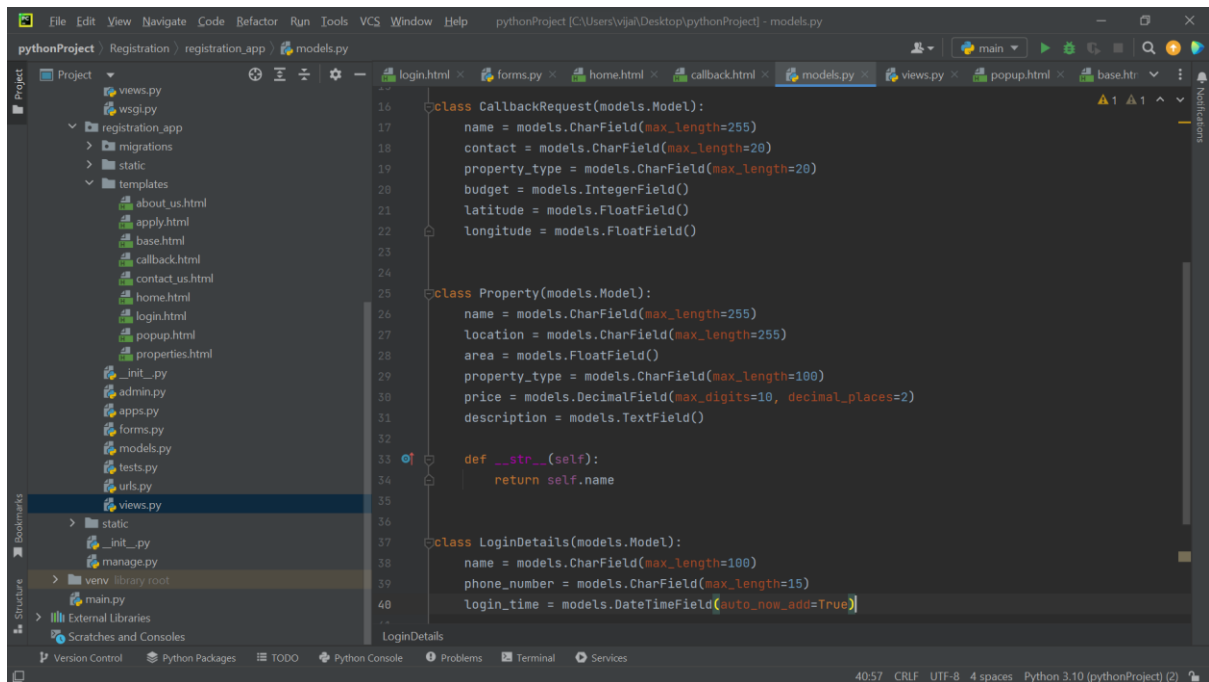
id	name	phone_number	login_time
1	Vijay	9786788833	2023-05-28 16:14:30.144286
2	Vijay	9786788833	2023-05-28 16:16:03.863462
3	Anand	9899732145	2023-05-28 16:16:34.960144
4	Vijay	9786788833	2023-05-28 16:31:12.403177
5	Shibhy	9942514497	2023-05-28 16:32:02.144710
6	Prakash	9899741234	2023-05-28 16:40:16.319395
7	Pandian	9943514489	2023-05-28 16:53:22.160295
8	Priya	9123514489	2023-05-28 16:54:19.009214
9	chandu	9942514497	2023-05-28 17:00:46.066815

registration\_app\_logindetails 17 x

Output

#	Time	Action	Message	Duration / Fetch
22	21:40:09	show tables	23 row(s) returned	0.000 sec / 0.000 sec
23	21:40:31	select * from registration_app_logindetails LIMIT 0, 50000	0 row(s) returned	0.000 sec / 0.000 sec
24	21:42:52	select * from registration_app_logindetails LIMIT 0, 50000	0 row(s) returned	0.000 sec / 0.000 sec
25	21:44:45	select * from registration_app_logindetails LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec
26	22:19:46	select * from registration_app_logindetails LIMIT 0, 50000	6 row(s) returned	0.000 sec / 0.000 sec
27	00:29:59	select * from registration_app_logindetails LIMIT 0, 50000	17 row(s) returned	0.016 sec / 0.000 sec

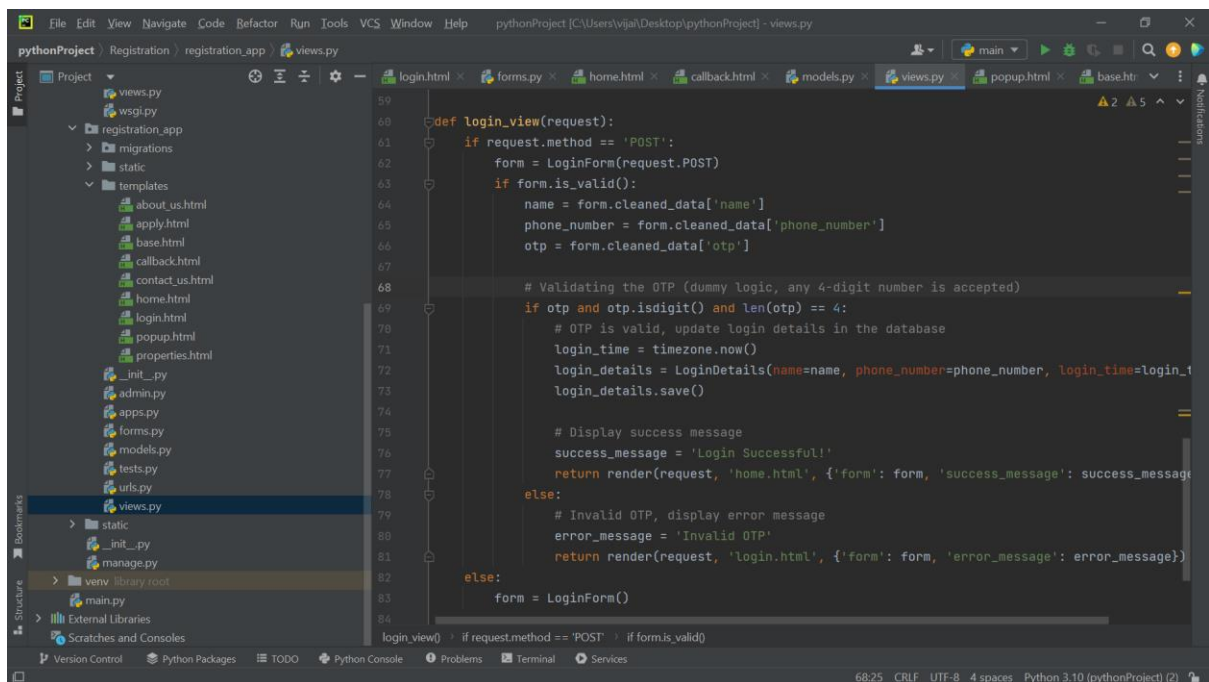
## Model.py



The screenshot shows the VS Code editor with the file `models.py` open. The left sidebar displays the project structure, including `views.py`, `registration_app`, `migrations`, `static`, and `templates`. The main editor area contains the following Python code:

```
16 class CallbackRequest(models.Model):
17     name = models.CharField(max_length=255)
18     contact = models.CharField(max_length=20)
19     property_type = models.CharField(max_length=20)
20     budget = models.IntegerField()
21     latitude = models.FloatField()
22     longitude = models.FloatField()
23
24
25 class Property(models.Model):
26     name = models.CharField(max_length=255)
27     location = models.CharField(max_length=255)
28     area = models.FloatField()
29     property_type = models.CharField(max_length=100)
30     price = models.DecimalField(max_digits=10, decimal_places=2)
31     description = models.TextField()
32
33     def __str__(self):
34         return self.name
35
36
37 class LoginDetails(models.Model):
38     name = models.CharField(max_length=100)
39     phone_number = models.CharField(max_length=15)
40     login_time = models.DateTimeField(auto_now_add=True)
```

## views.py



The screenshot shows the VS Code editor with the file `views.py` open. The left sidebar displays the project structure, including `views.py`, `registration_app`, `migrations`, `static`, and `templates`. The main editor area contains the following Python code:

```
59
60 def login_view(request):
61     if request.method == 'POST':
62         form = LoginForm(request.POST)
63         if form.is_valid():
64             name = form.cleaned_data['name']
65             phone_number = form.cleaned_data['phone_number']
66             otp = form.cleaned_data['otp']
67
68             # Validating the OTP (dummy logic, any 4-digit number is accepted)
69             if otp and otp.isdigit() and len(otp) == 4:
70                 # OTP is valid, update login details in the database
71                 login_time = timezone.now()
72                 login_details = LoginDetails(name=name, phone_number=phone_number, login_time=login_time)
73                 login_details.save()
74
75                 # Display success message
76                 success_message = 'Login Successful!'
77                 return render(request, 'home.html', {'form': form, 'success_message': success_message})
78             else:
79                 # Invalid OTP, display error message
80                 error_message = 'Invalid OTP'
81                 return render(request, 'login.html', {'form': form, 'error_message': error_message})
82         else:
83             form = LoginForm()
```