Employee Management System

PROJECT REPORT
OF MINI PROJECT

BACHELOR OF TECHNOLOGY
Data Science Dept **(VIIth Semester)**

SUBMITTED BY
Mohammad Arman (2101331540063)
Mohammad Ali (2101331540062)
Rahul Arya (2201331549016)

SUBMITTED TO
Mr. Sovers Singh Bisht

NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY,
GREATER NOIDA, UTTAR PRADESH

STUDENT'S DECLARATION

We hereby certify that the work which is being presented in the Mini project report entitled" "Employee Management System" in fulfillment of the requirement for the award of the Degree of Bachelor of Technology in Department of Data Science of Noida Institute of Engineering and Technology is an authentic record of our own work carried out during 7th semester.

Date: 20th November 2024

Name of the Student
Mohammad Arman
Mohammad Ali
Rahul Arya

The Mini project viva-voce examination of Mohammad Arman (2101331540063), Mohammad Ali (2101331540062), Rahul Arya (2201331549016) of B.Tech Data Science has been held on 20th November 2024.

Signature of:

Project Guide:_____

Head of Dept:
(Stamp of organization)

External Examiner:_____

Internal Examiner _____

ACKNOWLEDGEMENT

We are highly grateful to the Dr. Vinod M Kapse Director, **Noida Institute of Engineering and Technology**, Graeter Noida, for providing this opportunity.

The constant guidance and encouragement received from Dr. Manali Gupta, HOD (Data Science dept.), NIET, Greater Noida has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Mr. Sovers Singh Bisht for their project guide, without the wise counsel and able guidance, it would have been impossible to complete the report in this manner.

We express gratitude to other faculty members of Data Science Department of NIET for their intellectual support throughout the course of this work.

Finally, the authors are indebted to all whosoever have contributed in this report work.

Mohammad Arman
Mohammad Ali

Rahul Arya

INTRODUCTION

- The Django Employee Management System is a web application designed to simplify the management of employee data, enabling organizations to efficiently handle employee records,roles, departments, attendance, and leave management.

- The system allows administrators to perform CRUD operations on employee data whileproviding employees with secure access to view and manage their profiles, ensuring streamlined human resource operations within the organization.

- Built using Django's Model-View-Template (MVT) architecture, the system ensures a clear separation of database, business logic, and user interface, making it easy to maintain, extend,and scale as required.

- Featuring role-based access control, secure authentication, and integration with PostgreSQLfor data storage, the system provides a robust platform for employee management, offering administrators complete control while providing employees with self-service capabilities.

PROBLEM DEFINATION

- Managing employee information, attendance, and leave requests manually or using outdatedsystems can lead to inefficiencies, errors, and difficulties in data retrieval for HR departments.Organizations often struggle with maintaining up-to-date employee records, ensuring secure access to sensitive information, and tracking employee attendance and leave in an organizedway. As organizations grow, the complexity of handling employee data increases, making manual processes prone to inaccuracies, data loss, and poor decision-making.

- The problem lies in the lack of a centralized, user-friendly, and automated system that can streamline employee data management, offer secure role-based access, and ensure efficienthandling of attendance, leaves, and payroll. There is a need for a scalable solution that provides real-time access to employee information, reduces human error, and simplifies administrative tasks, thereby improving HR efficiency and overall organizational performance.

- The proposed solution aims to develop a web-based Employee Management System that addresses these challenges by automating key HR functions while ensuring data security andeasy accessibility.

PURPOSE OF THE PROJECT

- **Streamline Employee Data Management**: To automate and centralize the management of employee information, including personal details, roles, departments, and salaries, ensuring easy access and minimizing errors in data handling.

- **Enhance HR Efficiency**: To reduce the administrative burden on HR by providing a platformfor efficiently managing employee attendance, leave requests, and approvals, improving workflow and decision-making.

- **Implement Secure Access Control**: To ensure data security by implementing role-basedaccess control, allowing administrators to manage sensitive information while providingemployees with limited, secure access to their own profiles.

- **Enable Scalability and Integration**: To develop a scalable solution that can be easily extended or integrated with other systems, allowing the organization to adapt the platform asit grows, while maintaining a seamless user experience.

TOOLS AND TECHNOLGY USED

- **Django (Python Framework):** Used as the core framework for building the web application,implementing the Model-View-Template (MVT) architecture for efficient data management and user interface design.

- **HTML/CSS/JavaScript:** Utilized for front-end development, ensuring a responsive anduser-friendly interface for both administrators and employees.

- **SQLite (Django Built-in Database):** A lightweight, embedded relational database used forstoring and managing employee data, providing easy integration with Django and efficientdata handling during development.

- **Bootstrap:** Used for creating a responsive design, ensuring the application works seamlesslyacross different devices and screen sizes.

- **Django REST Framework (DRF):** Employed to create RESTful APIs, enabling communication between the front-end and back-end, and allowing potential integration withother systems.

- **Python:** The primary programming language used for developing the back-end logic andfunctionality of the application.

Software Requirement Specifications (SRS)

Introduction

- The Employee Management System is a web-based application developed using Django. It isdesigned to help organizations streamline and automate employee data management, including personal details, roles, attendance, and leave tracking. The system aims to enhance HR efficiency and ensure data security through role-based access control. This document outlines the functional and non-functional requirements of the system, serving as a guideline for development, testing, and deployment.

Functional Requirements

1. User Authentication and Authorization:

- Secure login system for administrators and employees.
- Role-based access control to limit user access based on roles (Admin, Employee).

2. Employee Management:

- Admins can add, update, and delete employee records.
- Employees can view and update their profiles, including personal details and job information.

3. Attendance Management:

- Employees can log daily attendance.
- Admins can view and manage attendance records for all employees.

4. Leave Management:

- Employees can submit leave requests and check leave balances.
- Admins can approve or reject leave requests and manage leave records.

5. Department and Role Management:

- Admins can assign and update departments and roles for employees.
- A clear hierarchy of roles and departments is maintained for smooth workflow management.

6. Salary Management:

- Admins can manage employee salaries, including adding, updating, and deleting salaryrecords.

Non-Functional Requirements

1. Performance:

- The system should handle multiple simultaneous users without performance degradation.
- Pages should load within 2-3 seconds under normal load.

2. Security:

- Implement HTTPS for secure communication.
- User data should be encrypted in storage and transit.
- Role-based access to ensure only authorized users can modify sensitive information.

3. Scalability:

- The system should be able to scale easily to accommodate future growth, such as more usersand additional features.

4. Usability:

- The system should have an intuitive and user-friendly interface.
- Both administrators and employees should be able to perform tasks with minimal training.

5. Reliability:

- The system should ensure high availability, with minimal downtime.
- Regular data backups should be in place to prevent data loss.

6. Maintainability:

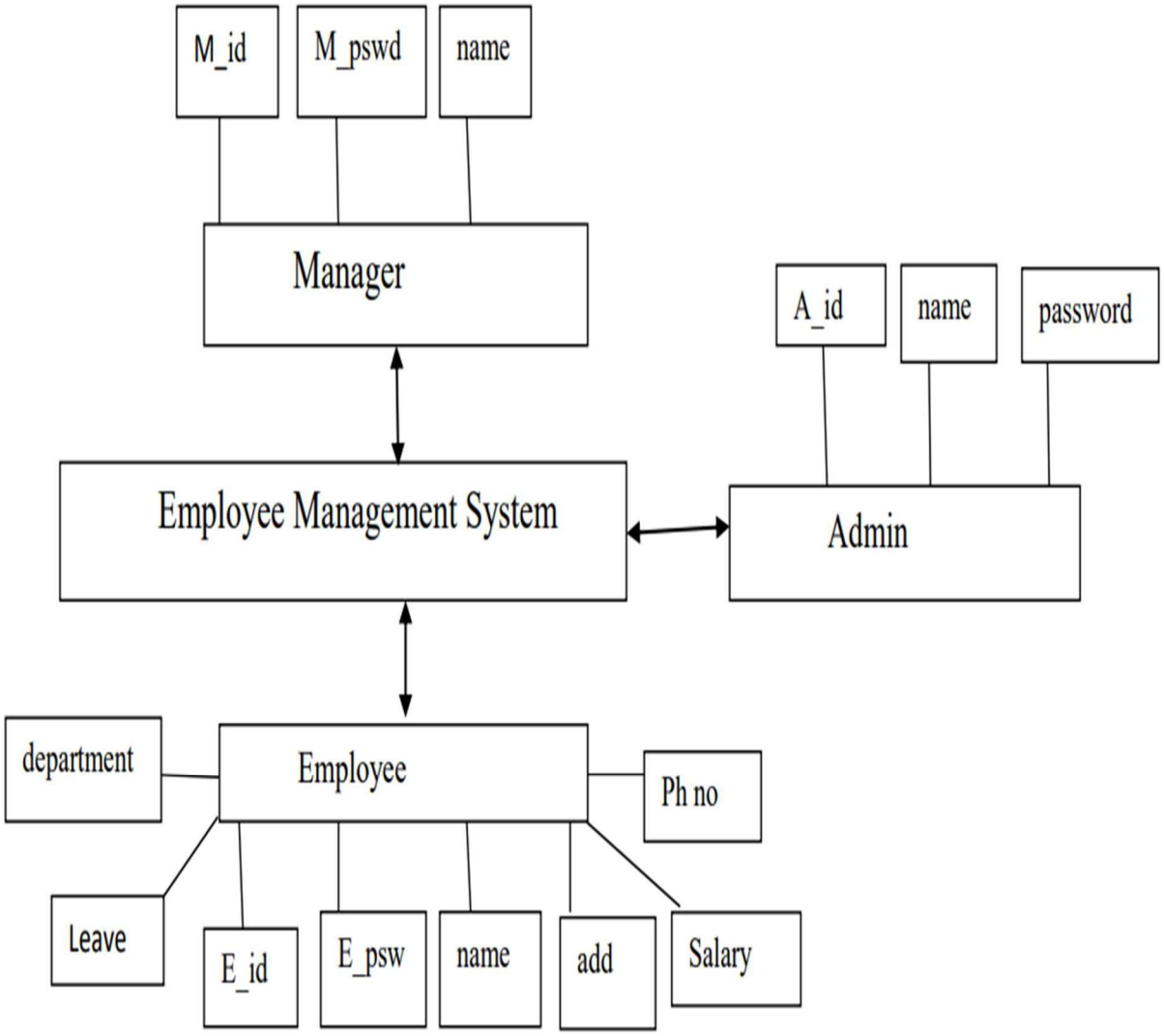
- The code should follow Django best practices, with proper documentation for easy futureupdates.
- Modularity in design to allow the addition of new features with minimal changes to existingfunctionality.

7. Compatibility:

- The system should work across different web browsers (Chrome, Firefox, Edge) and beresponsive on various devices (desktop, tablet, mobile).

These specifications serve as the foundation for the development and evaluation of the Employee Management System, ensuring it meets organizational needs efficiently and effectively.

SYSTEM DESIGN



The Employee Management System is a comprehensive web application developed using Python, Django, HTML, CSS, and SQLite3. The system is designed to streamline the management of employees, their departments, and administrative operations in an organization. The project is structured to cater to three main user roles: Admin, Manager, and Employee.

Key Features and Functionality

Admin Module: The Admin is at the core of the system, responsible for high-level control and management. Admins have credentials consisting of an A_id, name, and password. They oversee the database, manage employee and manager records, and assign department roles. This ensures that the organization's workflow remains efficient and secure.

Manager Module: Managers operate under the Admin's purview and manage employees directly. Each Manager is identified by a unique M_id, M_pswd (password), and name. Their role includes overseeing department functions, handling leave requests, and maintaining communication with their assigned team members.

Employee Module: Employees are the backbone of the organization. Each Employee has a unique E_id, E_psw (password), name, address, phone number (Ph_no), and salary. Employees are also associated with specific departments and can submit leave requests, which are processed by their respective managers.

System Design

The system utilizes a relational database structure provided by SQLite3. Each entity (Admin, Manager, and Employee) is represented as a table, with foreign key relationships ensuring data integrity. For example:

- Employees are linked to their respective departments.
- Managers oversee the employees within their assigned departments.
- Admins have control over both managers and employees.

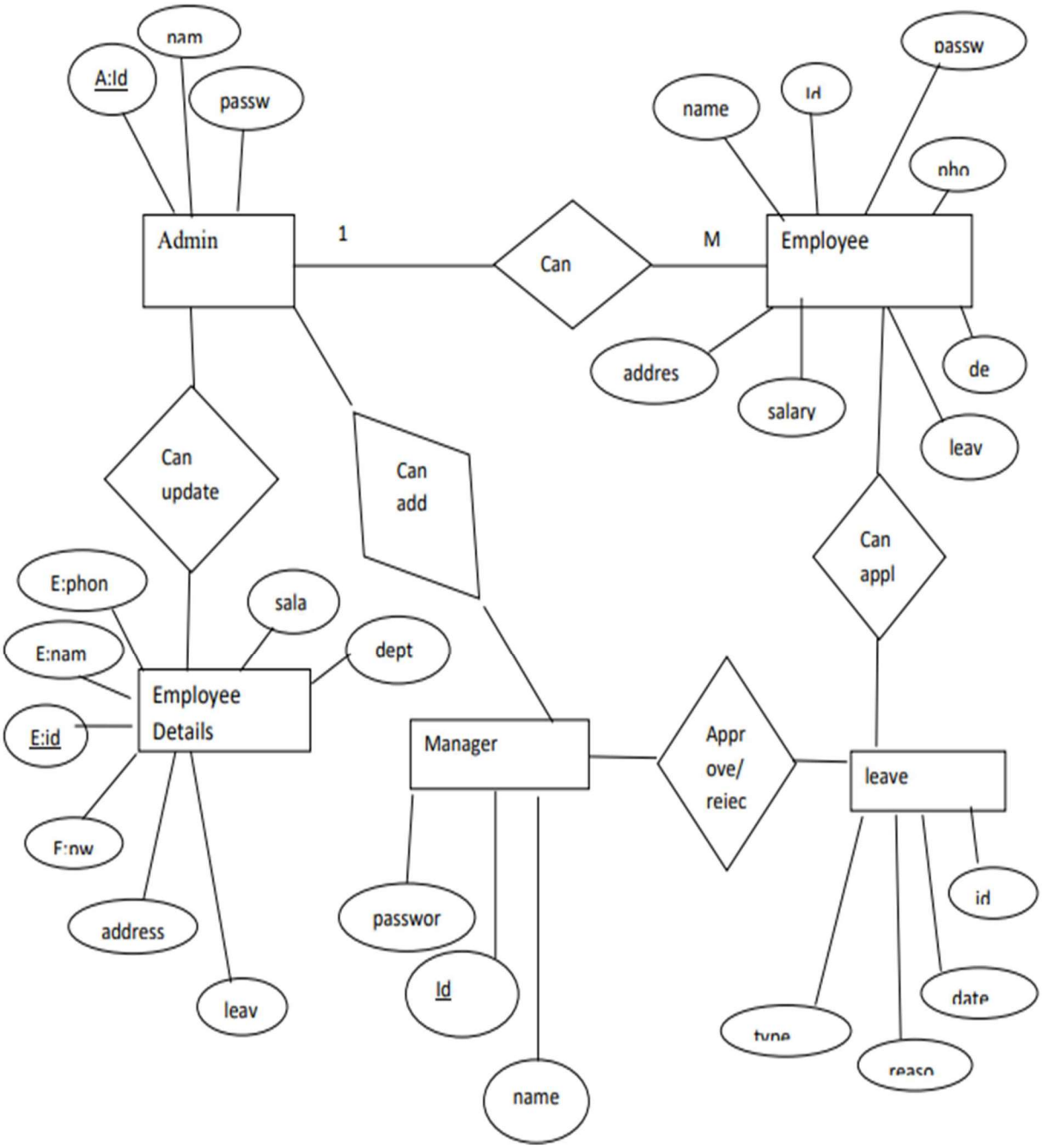
Technology Stack

Backend: Python and Django handle the server-side logic, ensuring secure authentication and efficient handling of user requests.

Frontend: HTML and CSS provide an intuitive and responsive interface, enabling seamless interaction for all users.

Database: SQLite3 ensures lightweight and reliable data storage with tables representing Admin, Manager, Employee, and associated records like leave requests.

Entity relationship diagram:



IMPLEMENTATION DETAILS:

Pseudo codes

Pseudo code for admin login

```
Begin

    If (admin ID exists in the database and password
        matches) Allow access to employee
        management page

    Else

        Display "Invalid admin ID/password".

    End if

End
```

Pseudo code for manager login page

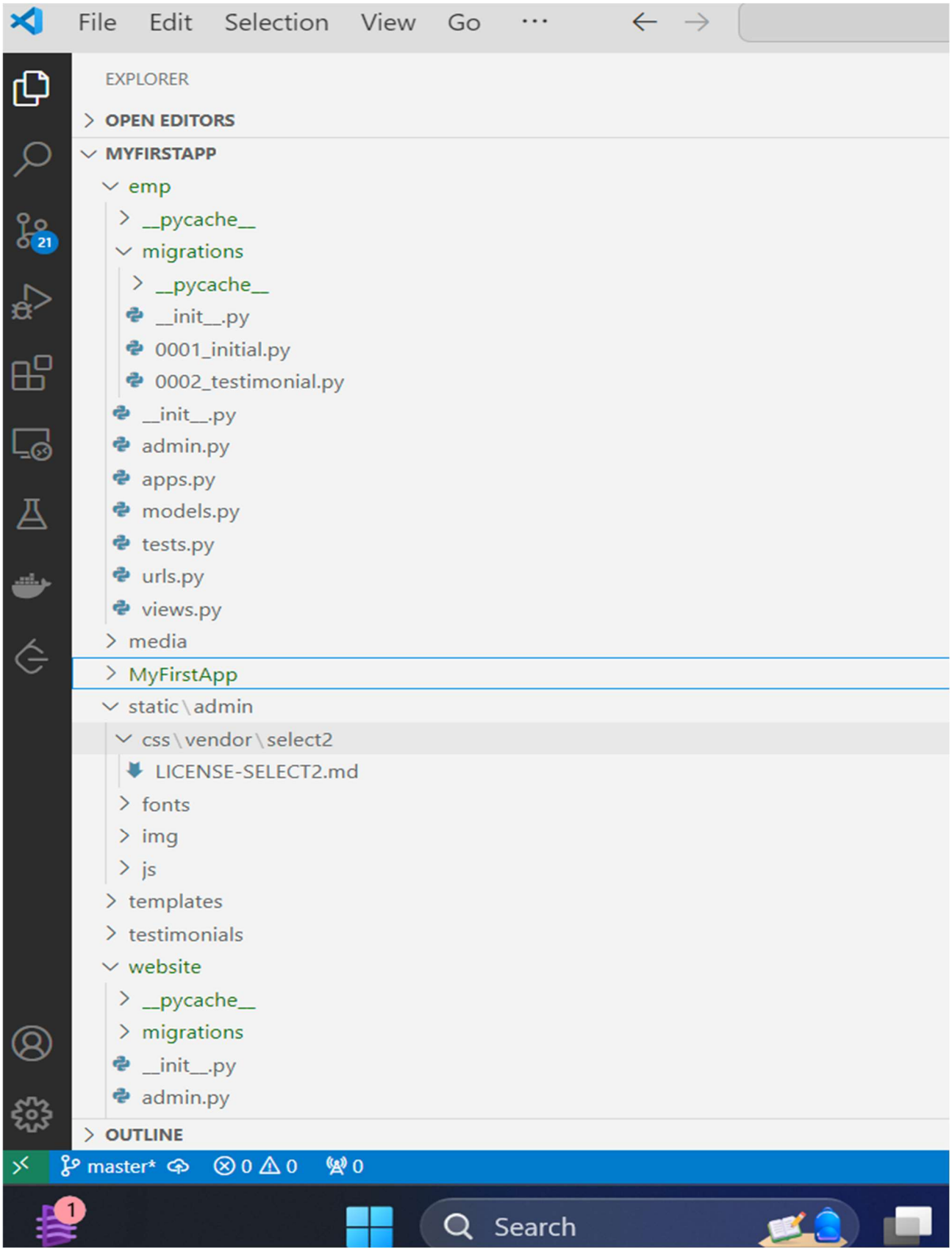
```
Begin

    If (User id exists in the database and password
        matches) Allow access and redirect to
        manager dashboard

    Else

        Display "Invalid User id or password"

    End if

End
```

Pseudo code for employee login page

```
Begin

    If (Employee id exists in the database and password
        matches) Allow access and redirect to employee
        dashboard

    Else

        Display "Invalid User id or password"

End if End
```

SYSTEM IMPLEMENTATION

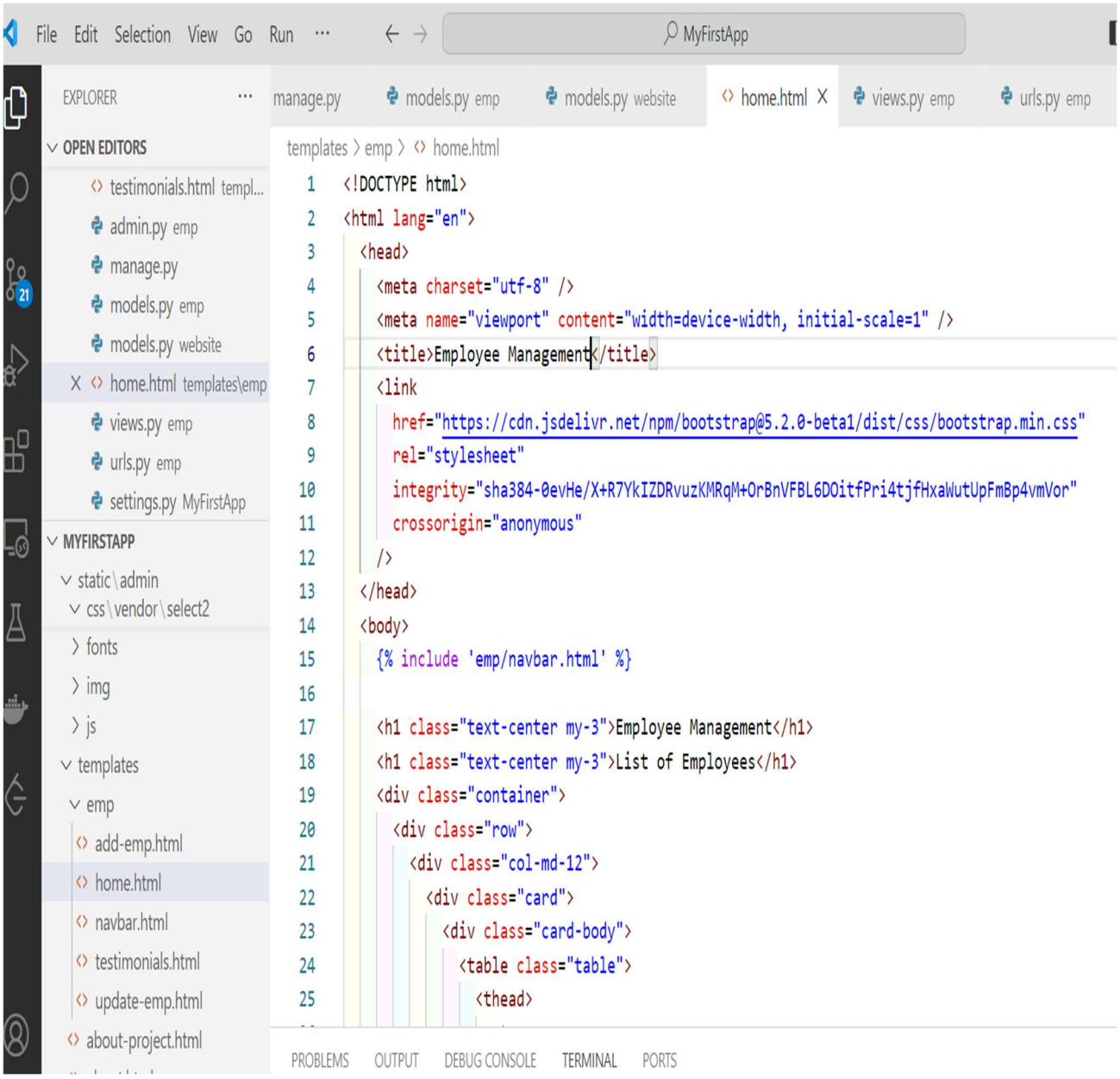
File directory

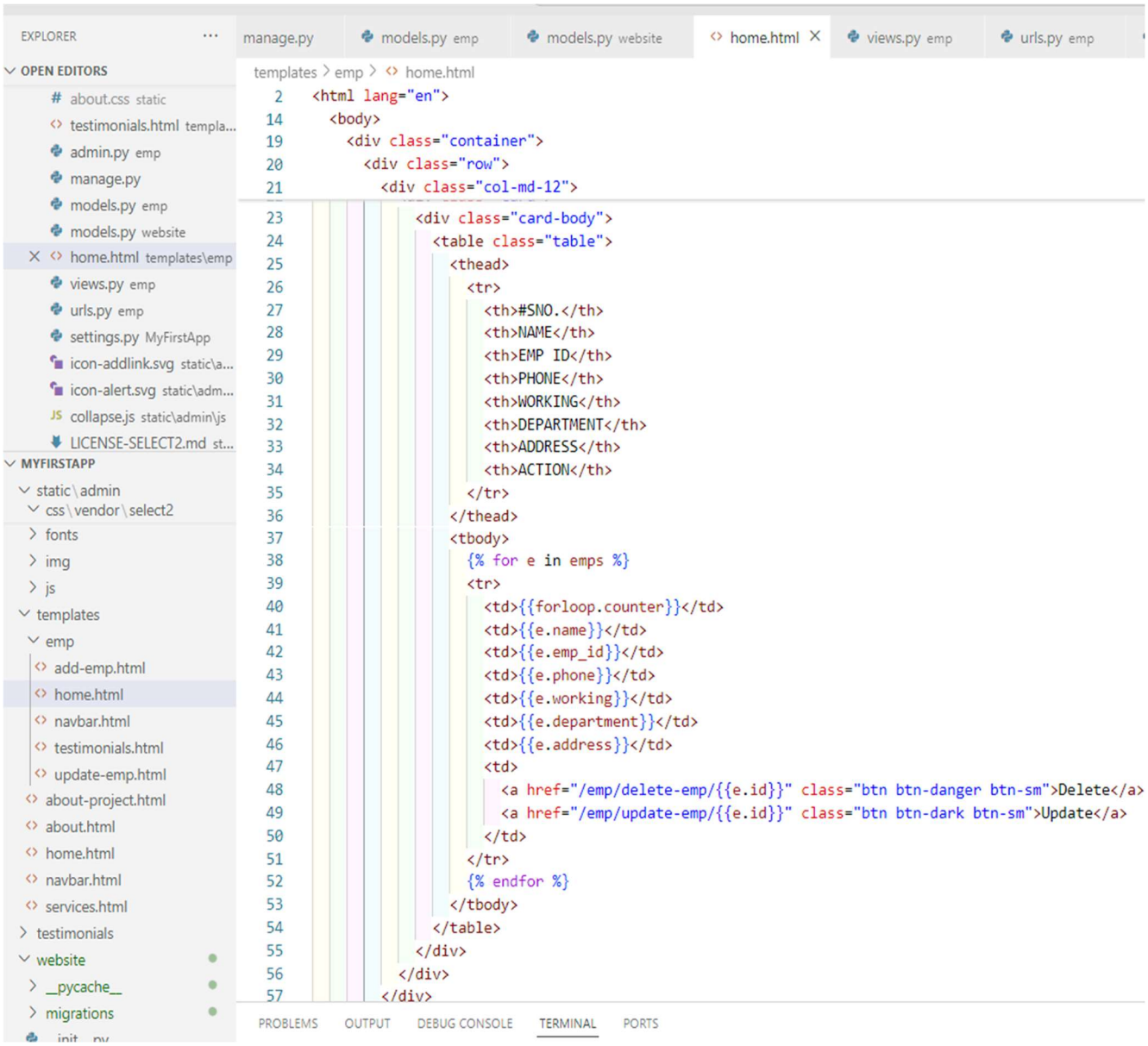


HOMEPAGE FOR EMPLOYEE MANAGEMENT SYSTEM

Coding:

home.html




EXPLORER
OPEN EDITORS
about.css static
testimonials.html templa...
admin.py emp
manage.py
models.py emp
models.py website
home.html templates\emp
views.py emp
urls.py emp
settings.py MyFirstApp
icon-addlink.svg static\a...
icon-alert.svg static\adm...
collapse.js static\admin\js
LICENSE-SELECT2.md st...
MYFIRSTAPP
static\admin
css\vendor\select2
fonts
img
js
templates
emp
add-emp.html
home.html
navbar.html
testimonials.html
update-emp.html
about-project.html
about.html
home.html
navbar.html
services.html
testimonials
website
__pycache__
migrations
manage.py
models.py emp
models.py website
home.html
views.py emp
urls.py emp
templates > emp > home.html
<html lang="en">
<body>

<table class="table">
<thead>
<tr>
<th>#SNO.</th>
<th>NAME</th>
<th>EMP ID</th>
<th>PHONE</th>
<th>WORKING</th>
<th>DEPARTMENT</th>
<th>ADDRESS</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
{% for e in emps %}
<tr>
<td>{{forloop.counter}}</td>
<td>{{e.name}}</td>
<td>{{e.emp_id}}</td>
<td>{{e.phone}}</td>
<td>{{e.working}}</td>
<td>{{e.department}}</td>
<td>{{e.address}}</td>
<td>
Delete
Update
</td>
</tr>
{% endfor %}
</tbody>
</table>

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

Navbnar.html

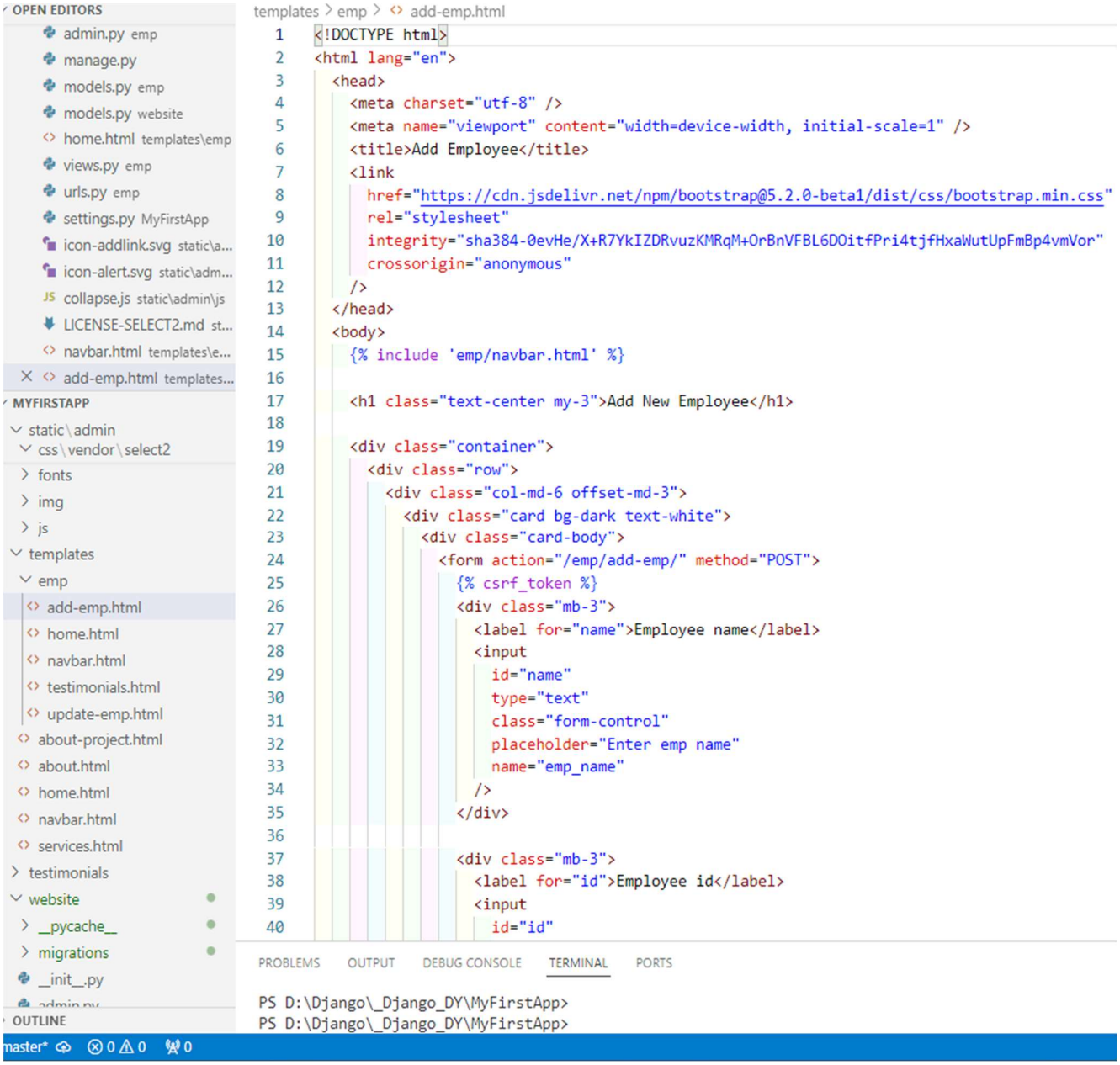
templates > emp > navbar.html

```
<nav class="navbar navbar-expand-lg bg-warning">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Employee Management</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/emp/home/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/emp/add-emp/">Add Employee</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/admin/">Admin</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/emp/testimonials/">Testimonials</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Dropdown
          </a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item" href="#">View Employee</a></li>
            <li><a class="dropdown-item" href="/about-project/">About</a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Follow</a></li>
          </ul>
        </li>
        {% comment %} <li class="nav-item">
          <a class="nav-link disabled" aria-disabled="true">Disabled</a>
        </li> {% endcomment %}
      </ul>
      {% comment %} <form class="d-flex" role="search">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form> {% endcomment %}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Django\_Django_DY\MyFirstApp>
```

Add-emp.html


OPEN EDITORS
admin.py emp
manage.py
models.py emp
models.py website
home.html templates\emp
views.py emp
urls.py emp
settings.py MyFirstApp
icon-addlink.svg static\a...
icon-alert.svg static\adm...
collapse.js static\admin\js
LICENSE-SELECT2.md st...
navbar.html templates\e...
add-emp.html templates...
MYFIRSTAPP
static\admin
css\vendor\select2
fonts
img
js
templates
emp
add-emp.html
home.html
navbar.html
testimonials.html
update-emp.html
about-project.html
about.html
home.html
navbar.html
services.html
testimonials
website
__pycache__
migrations
__init__.py
OUTLINE
templates > emp > add-emp.html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Add Employee</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous"
/>
</head>
<body>
{% include 'emp/navbar.html' %}
<h1 class="text-center my-3">Add New Employee</h1>

<form action="/emp/add-emp/" method="POST">
{% csrf_token %}

<label for="name">Employee name</label>
<input
id="name"
type="text"
class="form-control"
placeholder="Enter emp name"
name="emp_name"
/>

<label for="id">Employee id</label>
<input
id="id"
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS D:\Django_Django_DY\MyFirstApp>
PS D:\Django_Django_DY\MyFirstApp>
master*
0 0
0

templates > emp > add-emp.html

```
<html lang="en">
  <body>
    <div class="container">
      <div class="row">
        <div class="col-md-6 offset-md-3">
                    <input id="working" name="emp_working" type="checkbox" class="form-check-input">
                  </div>
                  <div class="mb-3 ">
                    <label for="department">Departments</label>
                    <select name="emp_department" id="department" class="form-control">
                        <option value="CSE">Computer Science</option>
                        <option value="ME">Mechanical Engg</option>
                        <option value="EE">Electrical Engg</option>
                    </select>
                  </div>
                  <div class="container text-center">
                      <button class="btn btn-primary">
                          Add Employee
                      </button>
                  </div>
                </form>
              </div>
            </div>
          </div>
        </div>
      </div>

      <form action="">
        {{form.as_p}}
      </form>

      <script
        src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
        crossorigin="anonymous"
      ></script>
    </body>
</html>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Django\_Django_DY\MyFirstApp>
PS D:\Django\_Django_DY\MyFirstApp>
```

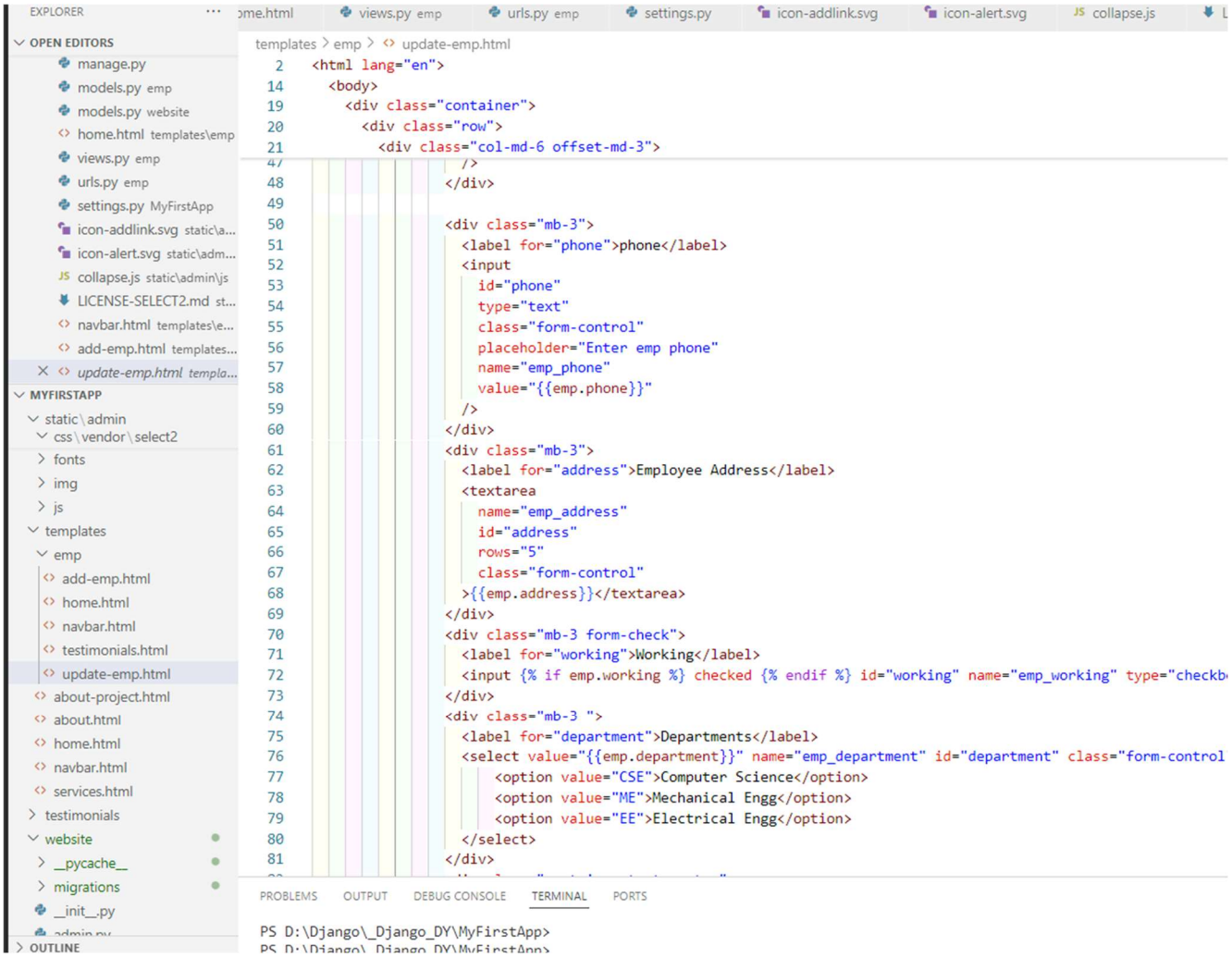
Update-emp.html

templates > emp > update-emp.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Update Employee</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
      crossorigin="anonymous"
    />
  </head>
  <body>
    {% include 'emp/navbar.html' %}

    <h1 class="text-center my-3">Update Employee</h1>

    <div class="container">
      <div class="row">
        <div class="col-md-6 offset-md-3">
          <div class="card bg-dark text-white">
            <div class="card-body">
              <form action="/emp/do-update-emp/{{emp.id}}" method="POST">
                {% csrf_token %}
                <div class="mb-3">
                  <label for="name">Employee name</label>
                  <input
                    id="name"
                    type="text"
                    class="form-control"
                    placeholder="Enter emp name"
                    name="emp_name"
                    value="{{emp.name}}"
                  />
                </div>

                <div class="mb-3">
                  <label for="id">Employee id</label>
                  <input
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Django\_Django_DY\MyFirstApp>
```


EXPLORER
OPEN EDITORS
manage.py
models.py emp
models.py website
home.html templates\emp
views.py emp
urls.py emp
settings.py MyFirstApp
icon-addlink.svg static\a...
icon-alert.svg static\adm...
collapse.js static\admin\js
LICENSE-SELECT2.md st...
navbar.html templates\e...
add-emp.html templates...
update-emp.html templa...
MYFIRSTAPP
static\admin
css\vendor\select2
fonts
img
js
templates
emp
add-emp.html
home.html
navbar.html
testimonials.html
update-emp.html
about-project.html
about.html
home.html
navbar.html
services.html
testimonials
website
__pycache__
migrations
__init__.py
OUTLINE
views.py emp
urls.py emp
settings.py
icon-addlink.svg
icon-alert.svg
collapse.js
templates > emp > update-emp.html
<html lang="en">
<body>

/>

<label for="phone">phone</label>
<input
id="phone"
type="text"
class="form-control"
placeholder="Enter emp phone"
name="emp_phone"
value="{{emp.phone}}"
/>

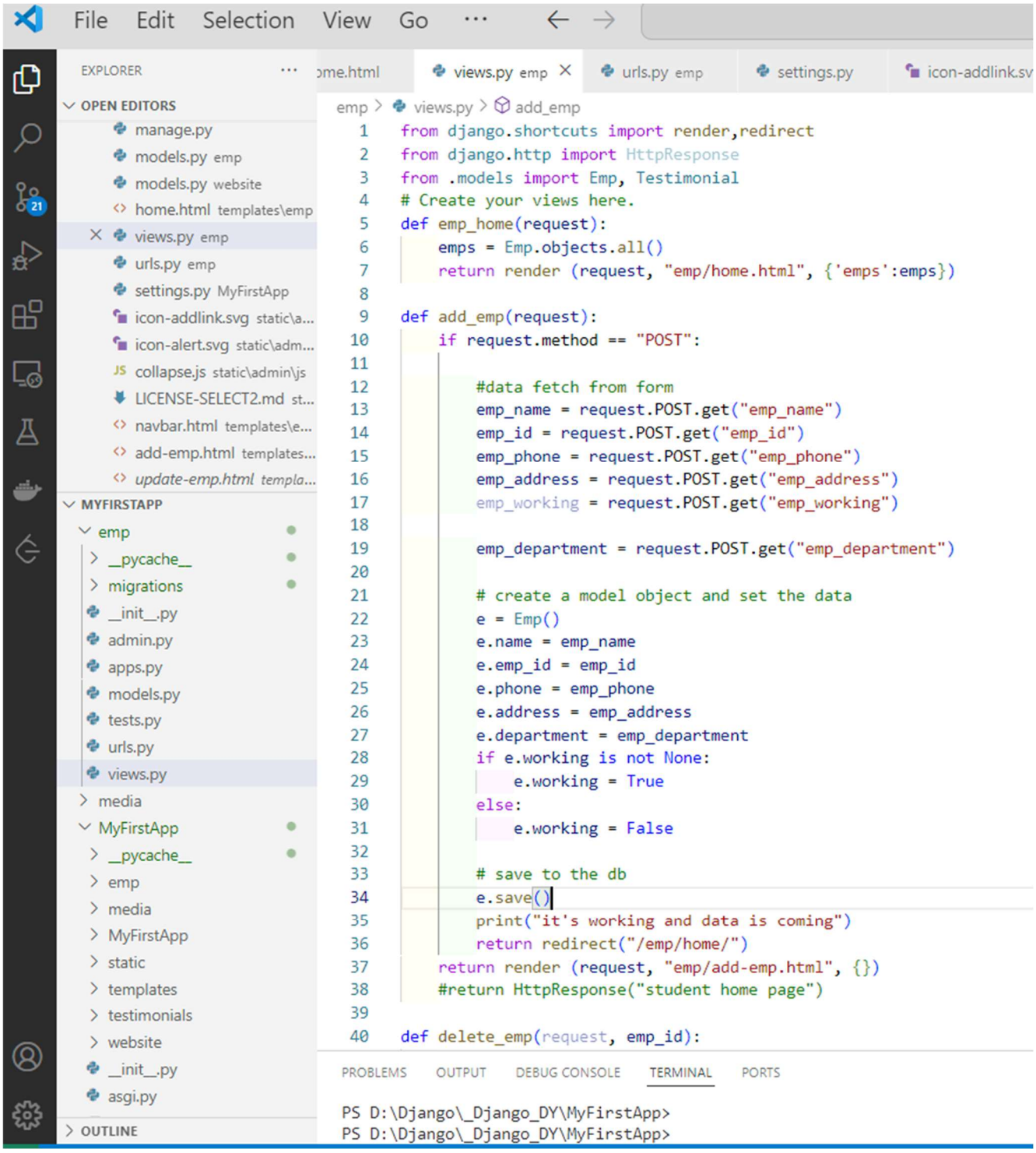
<label for="address">Employee Address</label>
<textarea
name="emp_address"
id="address"
rows="5"
class="form-control"
>{{emp.address}}</textarea>

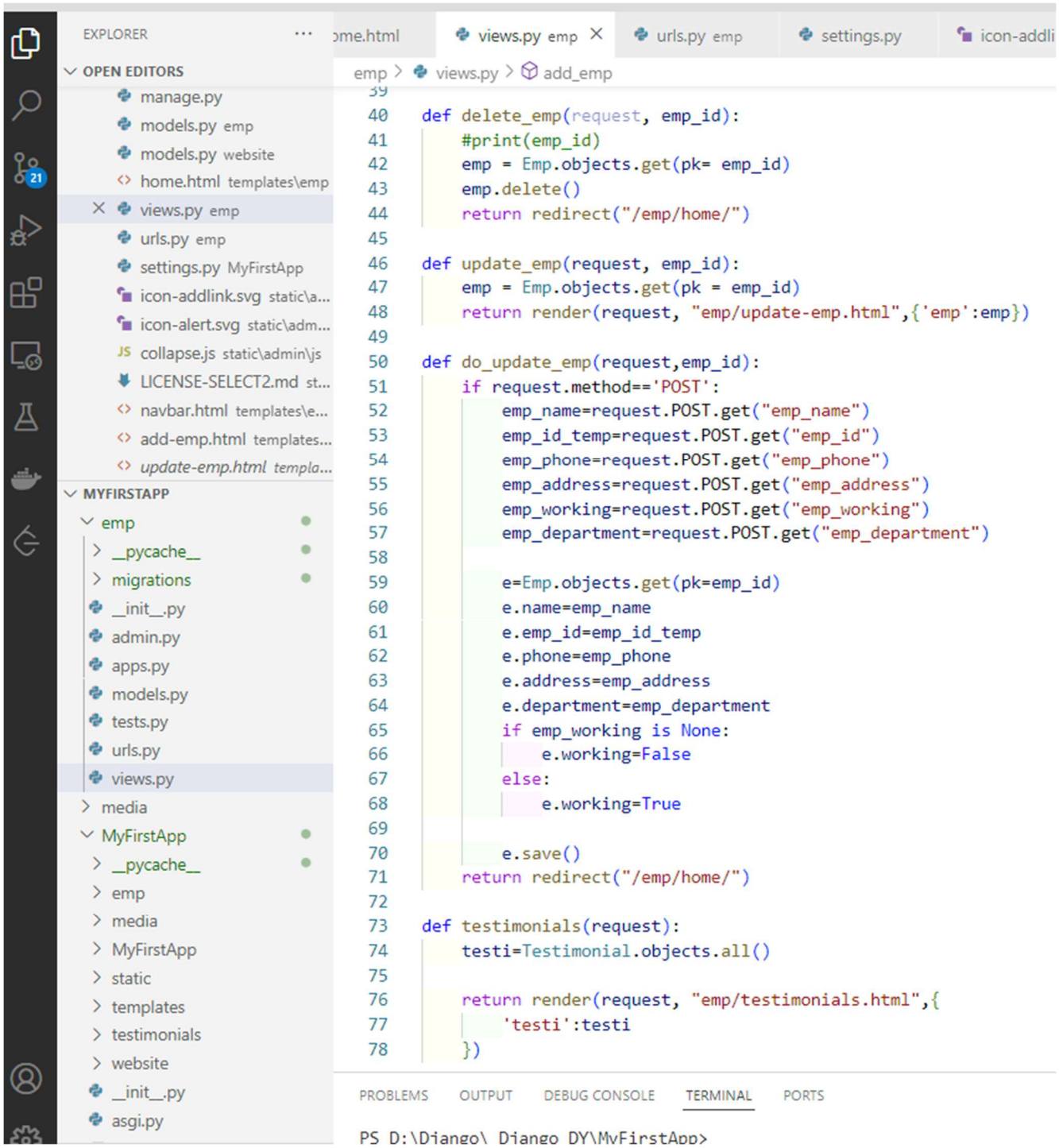
<label for="working">Working</label>
<input {% if emp.working %} checked {% endif %} id="working" name="emp_working" type="checkb

<label for="department">Departments</label>
<select value="{{emp.department}}" name="emp_department" id="department" class="form-control
<option value="CSE">Computer Science</option>
<option value="ME">Mechanical Engg</option>
<option value="EE">Electrical Engg</option>
</select>

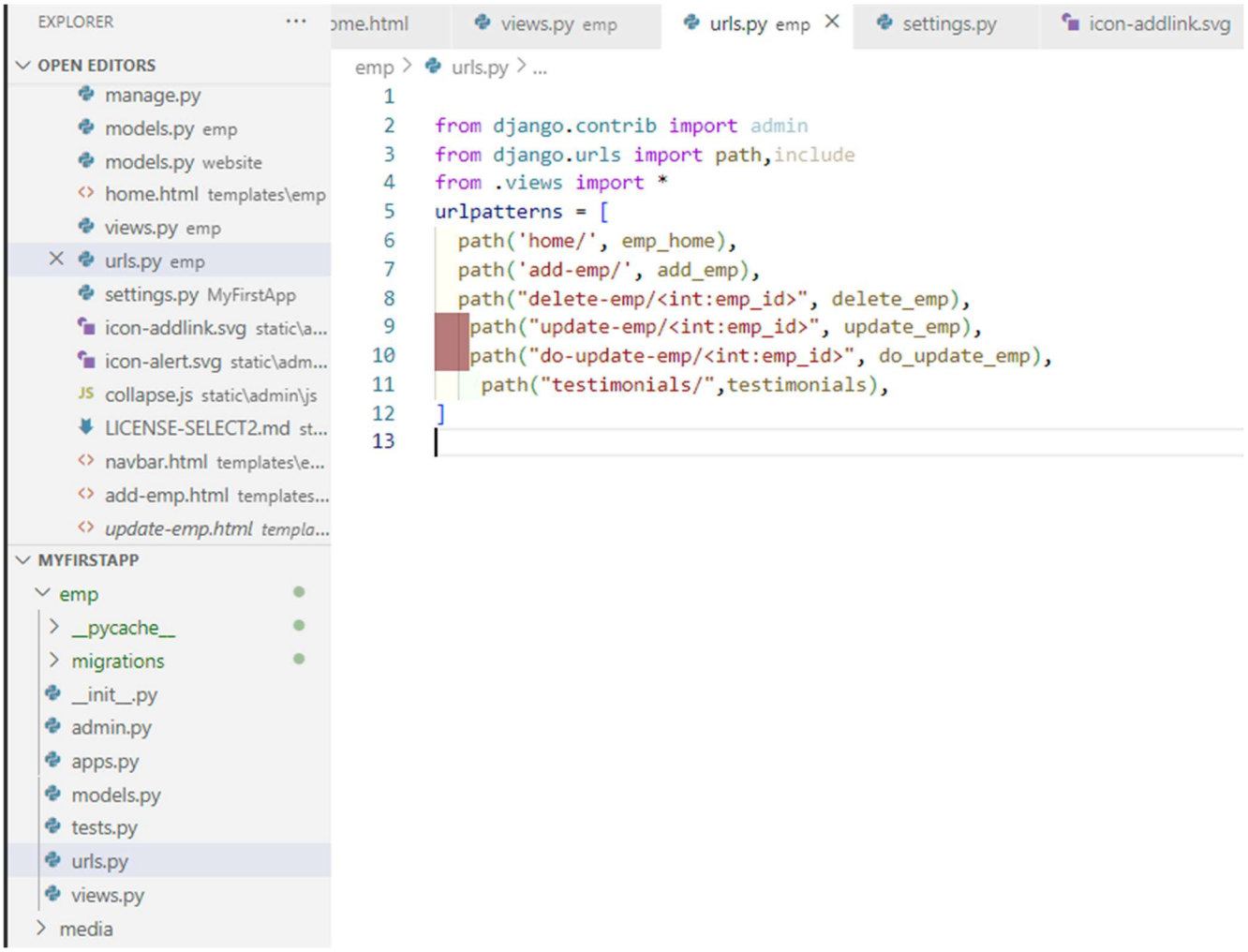
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS D:\Django_Django_DY\MyFirstApp>

views.py

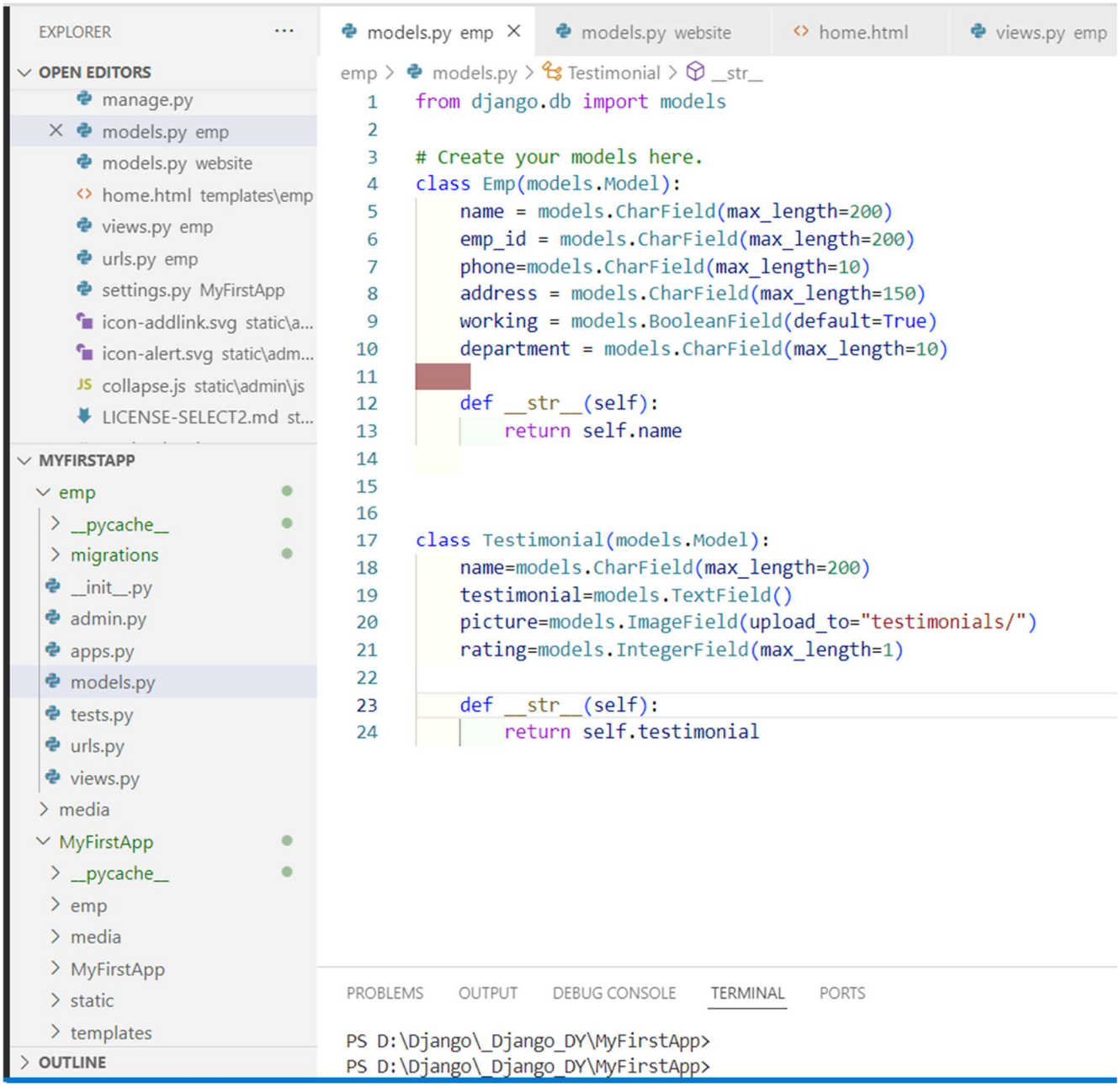

File Edit Selection View Go ···
EXPLORER
OPEN EDITORS
manage.py
models.py emp
models.py website
home.html templates\emp
views.py emp
urls.py emp
settings.py MyFirstApp
icon-addlink.svg static\a...
icon-alert.svg static\adm...
collapse.js static\admin\js
LICENSE-SELECT2.md st...
navbar.html templates\e...
add-emp.html templates...
update-emp.html templa...
MYFIRSTAPP
emp
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
media
MyFirstApp
__pycache__
emp
media
MyFirstApp
static
templates
testimonials
website
__init__.py
asgi.py
OUTLINE
ome.html
views.py emp
urls.py emp
settings.py
icon-addlink.sv
emp > views.py > add_emp
1 from django.shortcuts import render,redirect
2 from django.http import HttpResponse
3 from .models import Emp, Testimonial
4 # Create your views here.
5 def emp_home(request):
6 emps = Emp.objects.all()
7 return render (request, "emp/home.html", {'emps':emps})
8
9 def add_emp(request):
10 if request.method == "POST":
11
12 #data fetch from form
13 emp_name = request.POST.get("emp_name")
14 emp_id = request.POST.get("emp_id")
15 emp_phone = request.POST.get("emp_phone")
16 emp_address = request.POST.get("emp_address")
17 emp_working = request.POST.get("emp_working")
18
19 emp_department = request.POST.get("emp_department")
20
21 # create a model object and set the data
22 e = Emp()
23 e.name = emp_name
24 e.emp_id = emp_id
25 e.phone = emp_phone
26 e.address = emp_address
27 e.department = emp_department
28 if e.working is not None:
29 e.working = True
30 else:
31 e.working = False
32
33 # save to the db
34 e.save()
35 print("it's working and data is coming")
36 return redirect("/emp/home/")
37 return render (request, "emp/add-emp.html", {})
38 #return HttpResponse("student home page")
39
40 def delete_emp(request, emp_id):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Django_Django_DY\MyFirstApp>
PS D:\Django_Django_DY\MyFirstApp>


EXPLORER
OPEN EDITORS
manage.py
models.py emp
models.py website
home.html templates\emp
views.py emp
urls.py emp
settings.py MyFirstApp
icon-addlink.svg static\a...
icon-alert.svg static\adm...
collapse.js static\admin\js
LICENSE-SELECT2.md st...
navbar.html templates\e...
add-emp.html templates...
update-emp.html templa...
MYFIRSTAPP
emp
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
media
MyFirstApp
__pycache__
emp
media
MyFirstApp
static
templates
testimonials
website
__init__.py
asgi.py
ome.html
views.py emp
urls.py emp
settings.py
icon-addli
emp > views.py > add_emp
39
40 def delete_emp(request, emp_id):
41 #print(emp_id)
42 emp = Emp.objects.get(pk= emp_id)
43 emp.delete()
44 return redirect("/emp/home/")
45
46 def update_emp(request, emp_id):
47 emp = Emp.objects.get(pk = emp_id)
48 return render(request, "emp/update-emp.html",{'emp':emp})
49
50 def do_update_emp(request,emp_id):
51 if request.method=='POST':
52 emp_name=request.POST.get("emp_name")
53 emp_id_temp=request.POST.get("emp_id")
54 emp_phone=request.POST.get("emp_phone")
55 emp_address=request.POST.get("emp_address")
56 emp_working=request.POST.get("emp_working")
57 emp_department=request.POST.get("emp_department")
58
59 e=Emp.objects.get(pk=emp_id)
60 e.name=emp_name
61 e.emp_id=emp_id_temp
62 e.phone=emp_phone
63 e.address=emp_address
64 e.department=emp_department
65 if emp_working is None:
66 e.working=False
67 else:
68 e.working=True
69
70 e.save()
71 return redirect("/emp/home/")
72
73 def testimonials(request):
74 testi=Testimonial.objects.all()
75
76 return render(request, "emp/testimonials.html",{
77 'testi':testi
78 })
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS D:\Django\ Django DY\MyFirstApp>

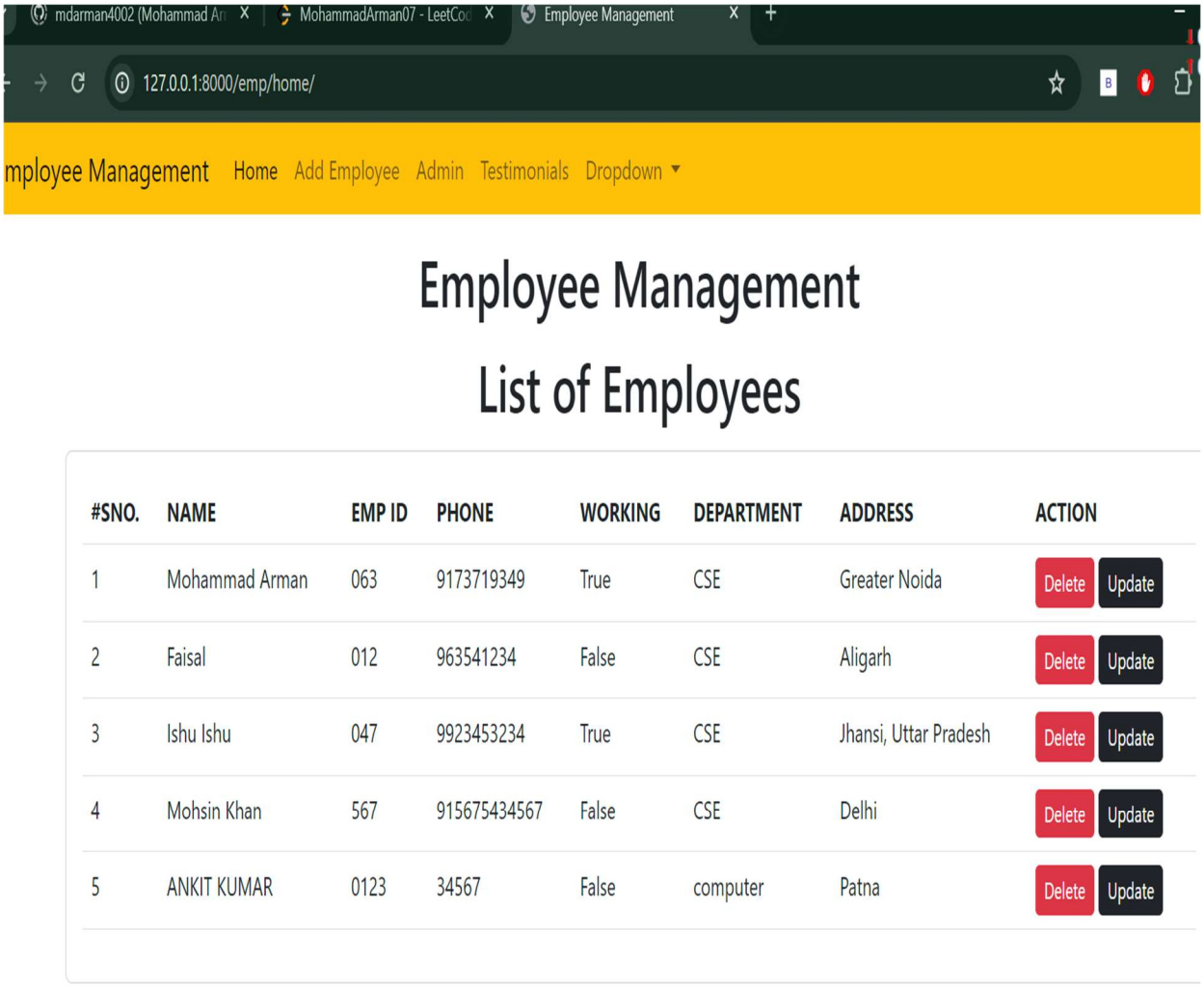
Urls.py


EXPLORER
OPEN EDITORS
manage.py
models.py emp
models.py website
home.html templates\emp
views.py emp
urls.py emp
settings.py MyFirstApp
icon-addlink.svg static\a...
icon-alert.svg static\adm...
collapse.js static\admin\js
LICENSE-SELECT2.md st...
navbar.html templates\e...
add-emp.html templates...
update-emp.html templa...
MYFIRSTAPP
emp
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
media
ome.html
views.py emp
urls.py emp
settings.py
icon-addlink.svg
emp > urls.py > ...
1
2 from django.contrib import admin
3 from django.urls import path,include
4 from .views import *
5 urlpatterns = [
6 path('home/', emp_home),
7 path('add-emp/', add_emp),
8 path("delete-emp/<int:emp_id>", delete_emp),
9 path("update-emp/<int:emp_id>", update_emp),
10 path("do-update-emp/<int:emp_id>", do_update_emp),
11 path("testimonials/",testimonials),
12]
13

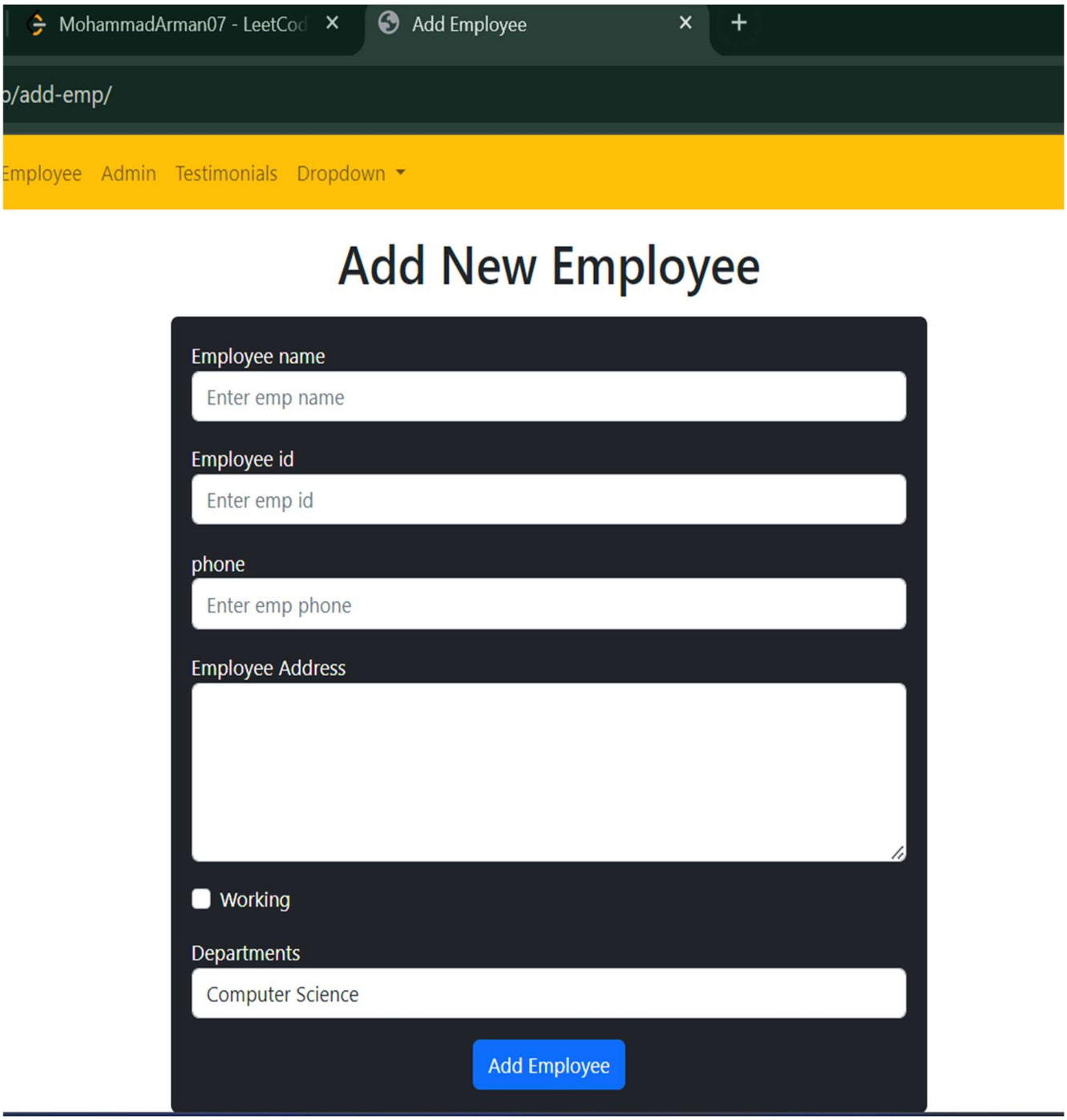
Models.py


EXPLORER
OPEN EDITORS
manage.py
models.py emp
models.py website
home.html templates\emp
views.py emp
urls.py emp
settings.py MyFirstApp
icon-addlink.svg static\a...
icon-alert.svg static\adm...
collapse.js static\admin\js
LICENSE-SELECT2.md st...
MYFIRSTAPP
emp
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
media
MyFirstApp
__pycache__
emp
media
MyFirstApp
static
templates
OUTLINE
models.py emp
models.py website
home.html
views.py emp
emp > models.py > Testimonial > __str__
1 from django.db import models
2
3 # Create your models here.
4 class Emp(models.Model):
5 name = models.CharField(max_length=200)
6 emp_id = models.CharField(max_length=200)
7 phone=models.CharField(max_length=10)
8 address = models.CharField(max_length=150)
9 working = models.BooleanField(default=True)
10 department = models.CharField(max_length=10)
11
12 def __str__(self):
13 return self.name
14
15
16
17 class Testimonial(models.Model):
18 name=models.CharField(max_length=200)
19 testimonial=models.TextField()
20 picture=models.ImageField(upload_to="testimonials/")
21 rating=models.IntegerField(max_length=1)
22
23 def __str__(self):
24 return self.testimonial
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS D:\Django_Django_DY\MyFirstApp>
PS D:\Django_Django_DY\MyFirstApp>

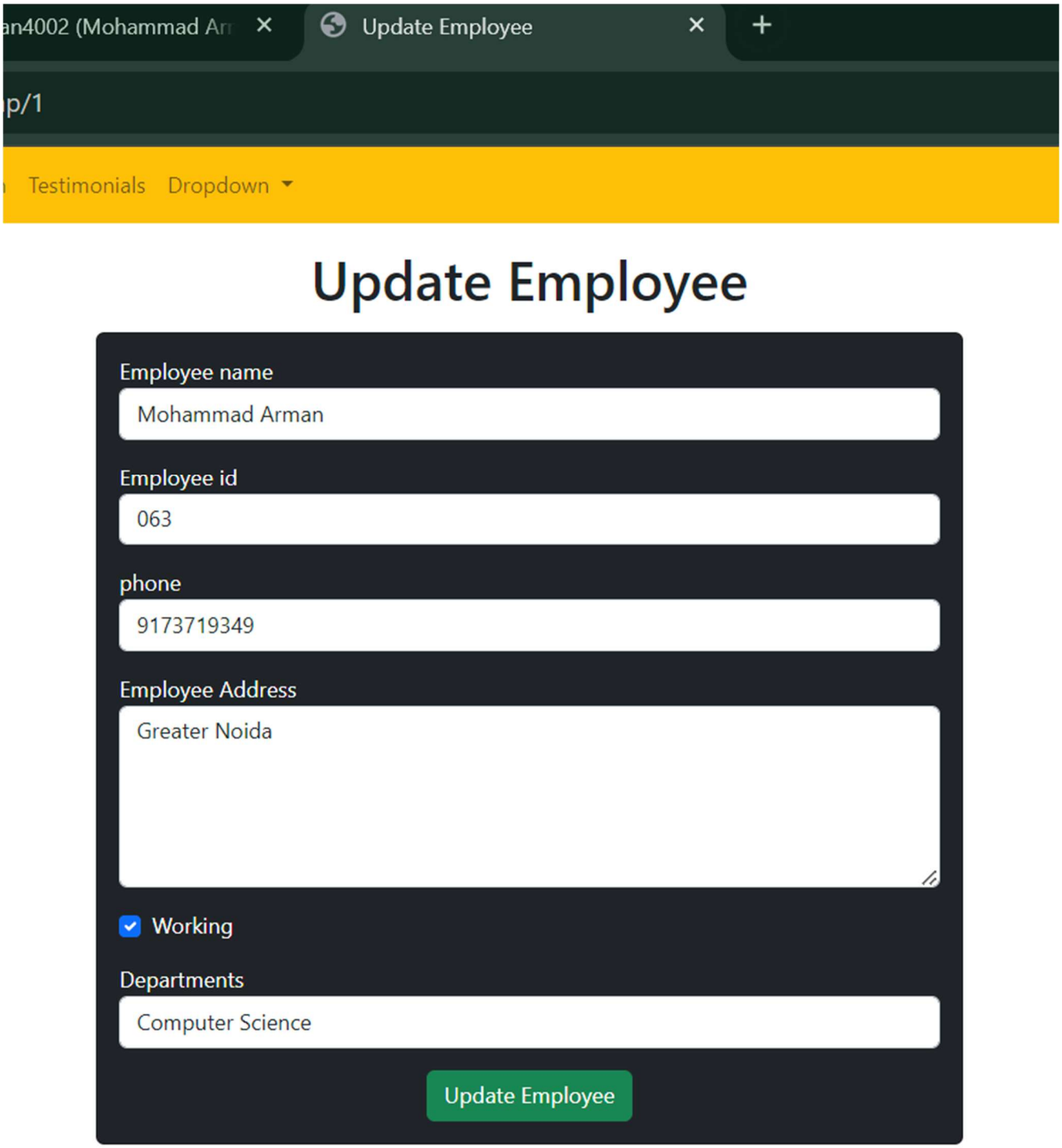
Home page


mdarman4002 (Mohammad Ar
MohammadArman07 - LeetCod
Employee Management
127.0.0.1:8000/emp/home/
mployee Management
Home
Add Employee
Admin
Testimonials
Dropdown
Employee Management
List of Employees
#SNO.
NAME
EMP ID
PHONE
WORKING
DEPARTMENT
ADDRESS
ACTION
1
Mohammad Arman
063
9173719349
True
CSE
Greater Noida
Delete
Update
2
Faisal
012
963541234
False
CSE
Aligarh
Delete
Update
3
Ishu Ishu
047
9923453234
True
CSE
Jhansi, Uttar Pradesh
Delete
Update
4
Mohsin Khan
567
915675434567
False
CSE
Delhi
Delete
Update
5
ANKIT KUMAR
0123
34567
False
computer
Patna
Delete
Update

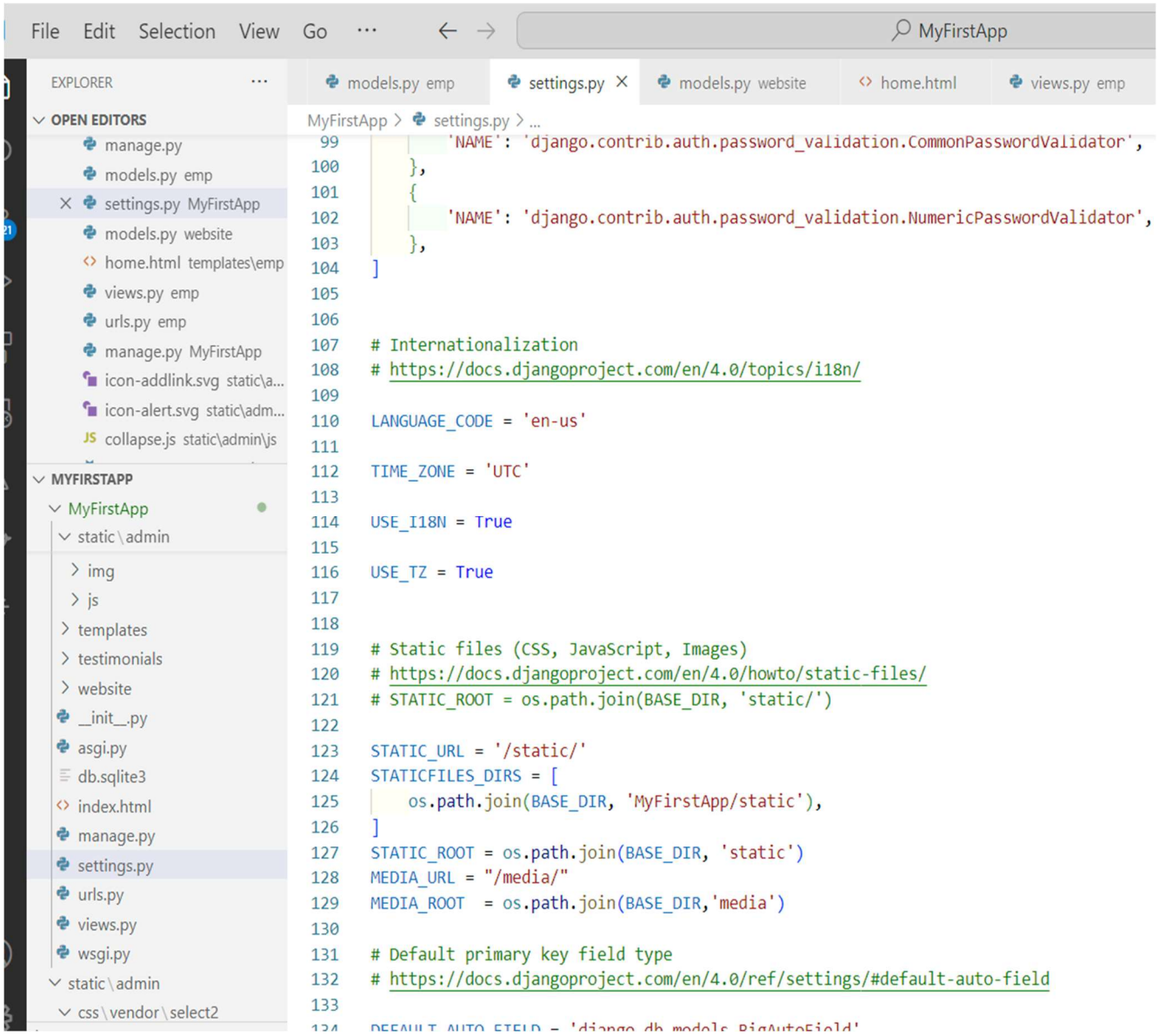
Add employee page



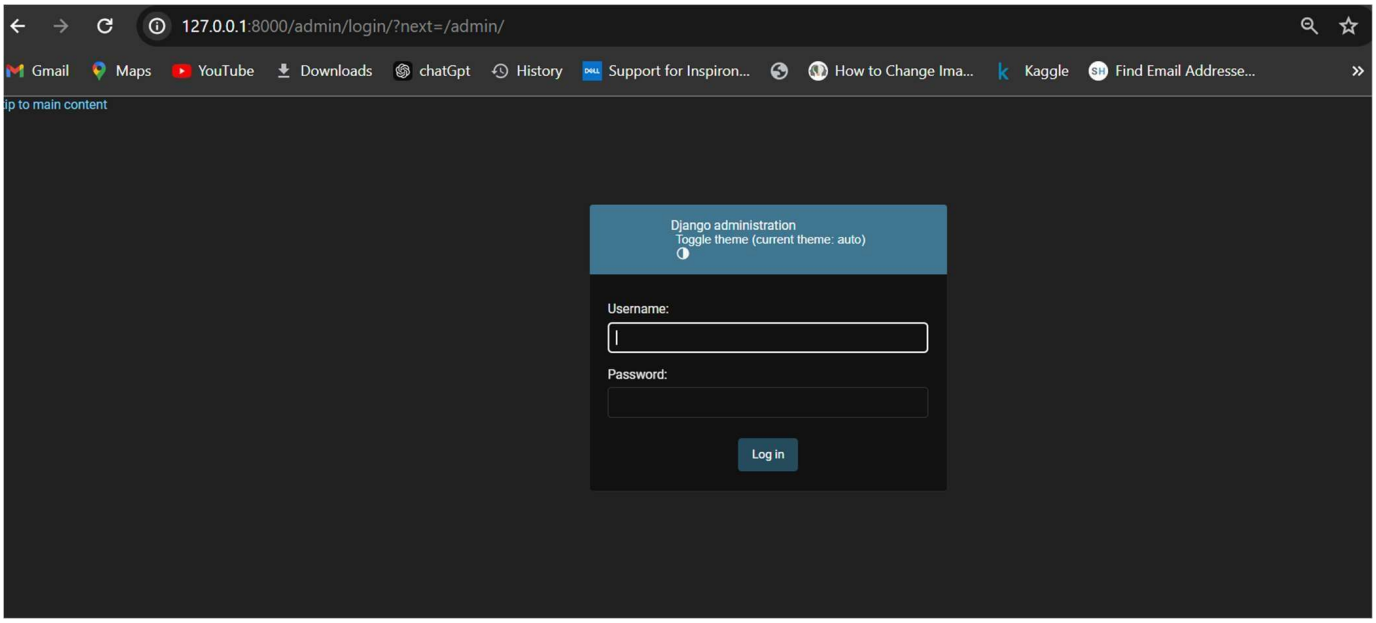
Update employee page

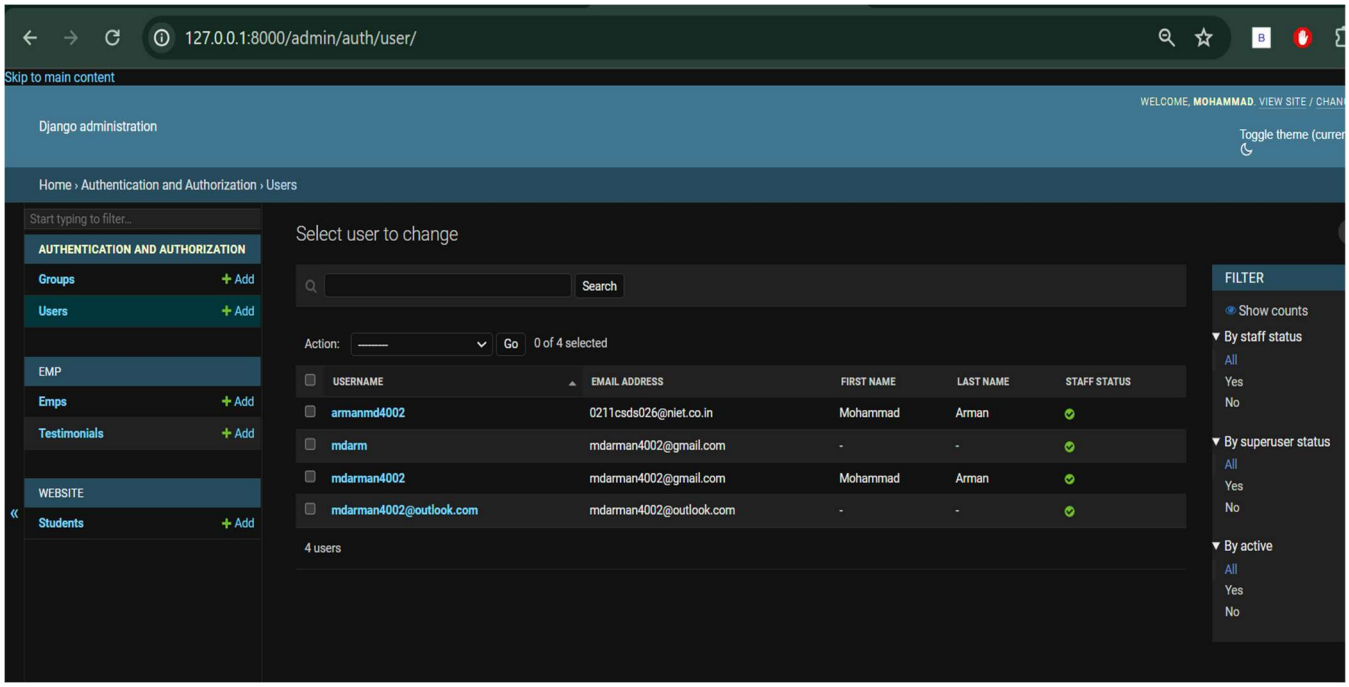


setting.py


File Edit Selection View Go ···
MyFirstApp
EXPLORER
models.py emp
settings.py
models.py website
home.html
views.py emp
OPEN EDITORS
manage.py
models.py emp
settings.py MyFirstApp
models.py website
home.html templates\emp
views.py emp
urls.py emp
manage.py MyFirstApp
icon-addlink.svg static\a...
icon-alert.svg static\adm...
collapse.js static\admin\js
MYFIRSTAPP
MyFirstApp
static\admin
img
js
templates
testimonials
website
__init__.py
asgi.py
db.sqlite3
index.html
manage.py
settings.py
urls.py
views.py
wsgi.py
static\admin
css\vendor\select2
MyFirstApp > settings.py > ...
99 'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
100 },
101 {
102 'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
103 },
104]
105
106
107 # Internationalization
108 # https://docs.djangoproject.com/en/4.0/topics/i18n/
109
110 LANGUAGE_CODE = 'en-us'
111
112 TIME_ZONE = 'UTC'
113
114 USE_I18N = True
115
116 USE_TZ = True
117
118
119 # Static files (CSS, JavaScript, Images)
120 # https://docs.djangoproject.com/en/4.0/howto/static-files/
121 # STATIC_ROOT = os.path.join(BASE_DIR, 'static/')
122
123 STATIC_URL = '/static/'
124 STATICFILES_DIRS = [
125 os.path.join(BASE_DIR, 'MyFirstApp/static'),
126]
127 STATIC_ROOT = os.path.join(BASE_DIR, 'static')
128 MEDIA_URL = "/media/"
129 MEDIA_ROOT = os.path.join(BASE_DIR,'media')
130
131 # Default primary key field type
132 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
133

Admin page


127.0.0.1:8000/admin/login/?next=/admin/
Gmail
Maps
YouTube
Downloads
chatGpt
History
Support for Inspiron...
How to Change Ima...
Kaggle
Find Email Addresse...
ip to main content
Django administration
Toggle theme (current theme: auto)
Username:
Password:
Log in


127.0.0.1:8000/admin/auth/user/
Skip to main content
Django administration
WELCOME, MOHAMMAD. VIEW SITE / CHANG
Toggle theme (curren
Home › Authentication and Authorization › Users
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
EMP
Emps + Add
Testimonials + Add
WEBSITE
Students + Add
Select user to change
Search
Action: ——— Go 0 of 4 selected
USERNAME
EMAIL ADDRESS
FIRST NAME
LAST NAME
STAFF STATUS
armanmd4002
0211csds026@niet.co.in
Mohammad
Arman
mdarm
mdarman4002@gmail.com
mdarman4002
mdarman4002@gmail.com
Mohammad
Arman
mdarman4002@outlook.com
mdarman4002@outlook.com
4 users
FILTER
Show counts
By staff status
All
Yes
No
By superuser status
All
Yes
No
By active
All
Yes
No

1. Conclusion

The conclusion should summarize the overall project, its objectives, and how successfully they were achieved. Highlight the importance of the system and its impact on solving the stated problem.

"The Employee Management System provides a robust solution for managing employee records, enhancing efficiency, and reducing manual errors. By leveraging Python, Django, and SQLite, the system ensures scalability, reliability, and user-friendly operation. This project has successfully met its objectives by automating administrative tasks, allowing for better resource management, and improving overall transparency in operations. The system also demonstrates the effective integration of web technologies with database management, making it an invaluable tool for organizational success."

2. Future Scope

Discuss how the system can be improved or extended in the future. This shows your vision and potential for growth.

- **Advanced Analytics:** Incorporate features such as employee performance reports, leave trend analysis, and department-specific insights to support data-driven decisions.
- **Payroll Integration:** Link the system with payroll software to calculate salaries based on attendance and leaves.
- **Mobile Application:** Develop an Android or iOS application for easier access to the system by employees and managers.
- **Cloud Integration:** Deploy the system on a cloud platform for greater accessibility and scalability.
- **Role-Based Access Control:** Implement a more secure system with hierarchical access levels for Admin, Managers, and Employees.

3. Challenges Faced

Mention any difficulties you encountered during the project and how you resolved them. This reflects your problem-solving skills and perseverance.

- **Database Integration:** Initially, connecting Django to SQLite was challenging due to configuration errors. However, after exploring Django's ORM and debugging the settings, the issue was resolved.
- **UI/UX Design:** Designing a user-friendly interface required iterative testing and incorporating feedback from potential users to make the system intuitive.
- **Authentication Issues:** Ensuring secure login mechanisms for Admin, Managers, and Employees involved implementing Django's built-in authentication features, which required learning about user session handling.
- **Testing and Debugging:** Identifying and fixing bugs, especially in leave management and salary calculations, required rigorous testing and refining the code.

4. Learnings

Outline the knowledge and skills gained through this project. This demonstrates how the project contributed to your technical and professional growth.

- Gained in-depth knowledge of Django, including URL routing, views, models, and templates.
- Understood database design and implemented relationships between tables using Django ORM.
- Enhanced front-end development skills with HTML, CSS, and responsive design techniques.
- Improved debugging and testing practices to ensure system stability and functionality.
- Learned about secure authentication mechanisms and data validation in web applications.
- Gained experience in deploying a Django application and integrating it with a database.

5. Acknowledgments

Recognize and appreciate the people who guided or supported you during the project. This reflects your gratitude and professionalism.

- "I would like to express my sincere gratitude to my mentor **Mr. Sovers Singh Bisht**, for their invaluable guidance and encouragement throughout this project. Their insights and expertise have been instrumental in completing the Employee Management System. Finally, I am grateful to my classmates and friends for their collaboration and moral support during the project's development."

6. References

List all resources, websites, or materials you referred to during the development of your project. This ensures credibility and gives credit to the sources.

Django Documentation: https://docs.djangoproject.com
Python SQLite Guide: https://docs.python.org/3/library/sqlite3.html
Bootstrap Documentation: https://getbootstrap.com
Python tutorial: https://www.geeksforgeeks.org/python-programming-language-tutorial/
W3Schools CSS Tutorials: https://www.w3schools.com/css