Employee Management System

PROJECT REPORT
OF MINI PROJECT

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

SUBMITTED BY
Mohammad Arman (2101331540063)

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: 21st November 2024

Name of the Student
Mohammad Arman

The Mini project viva-voce examination of Mohammad Arman (2101331540063) of B.TECH Data Science has beenheld on 21st 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

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 while providing 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 PostgreSQL for 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 outdated systems 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 organized way. 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 efficient handling 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 and easy 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 platform for efficiently managing employee attendance, leave requests, and approvals, improving workflow and decision-making.

- **Implement Secure Access Control**: To ensure data security by implementing role-based access control, allowing administrators to manage sensitive information while providing employees 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 as it 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 and user-friendly interface for both administrators and employees.

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

- **Bootstrap:** Used for creating a responsive design, ensuring the application works seamlessly across 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 with other systems.

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

Software Requirement Specifications (SRS)

Introduction

- The Employee Management System is a web-based application developed using Django. It is designed 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 salary records.

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 users and 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:

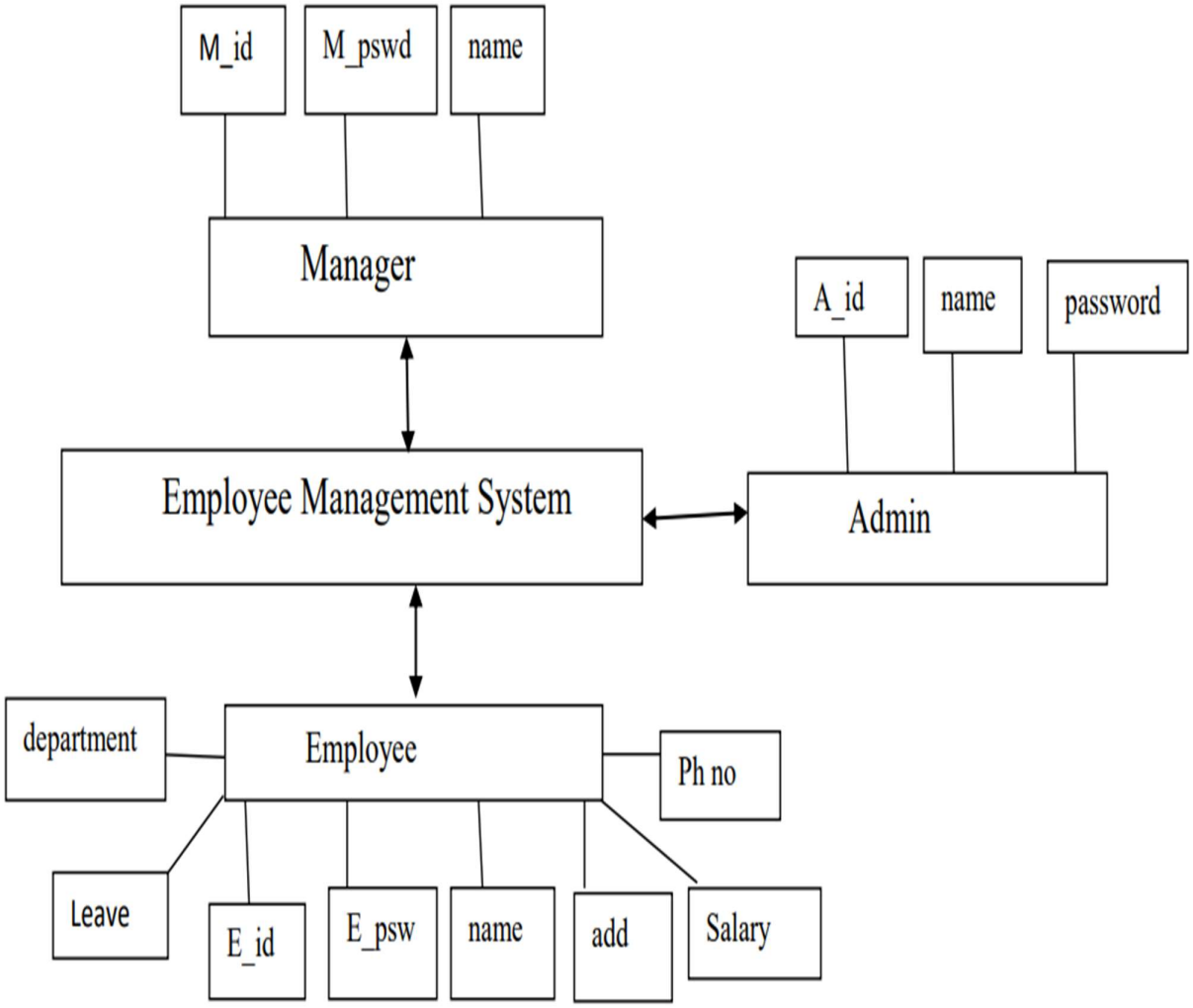
- The code should follow Django best practices, with proper documentation for easy future updates.
- Modularity in design to allow the addition of new features with minimal changes to existing functionality.

7. Compatibility:

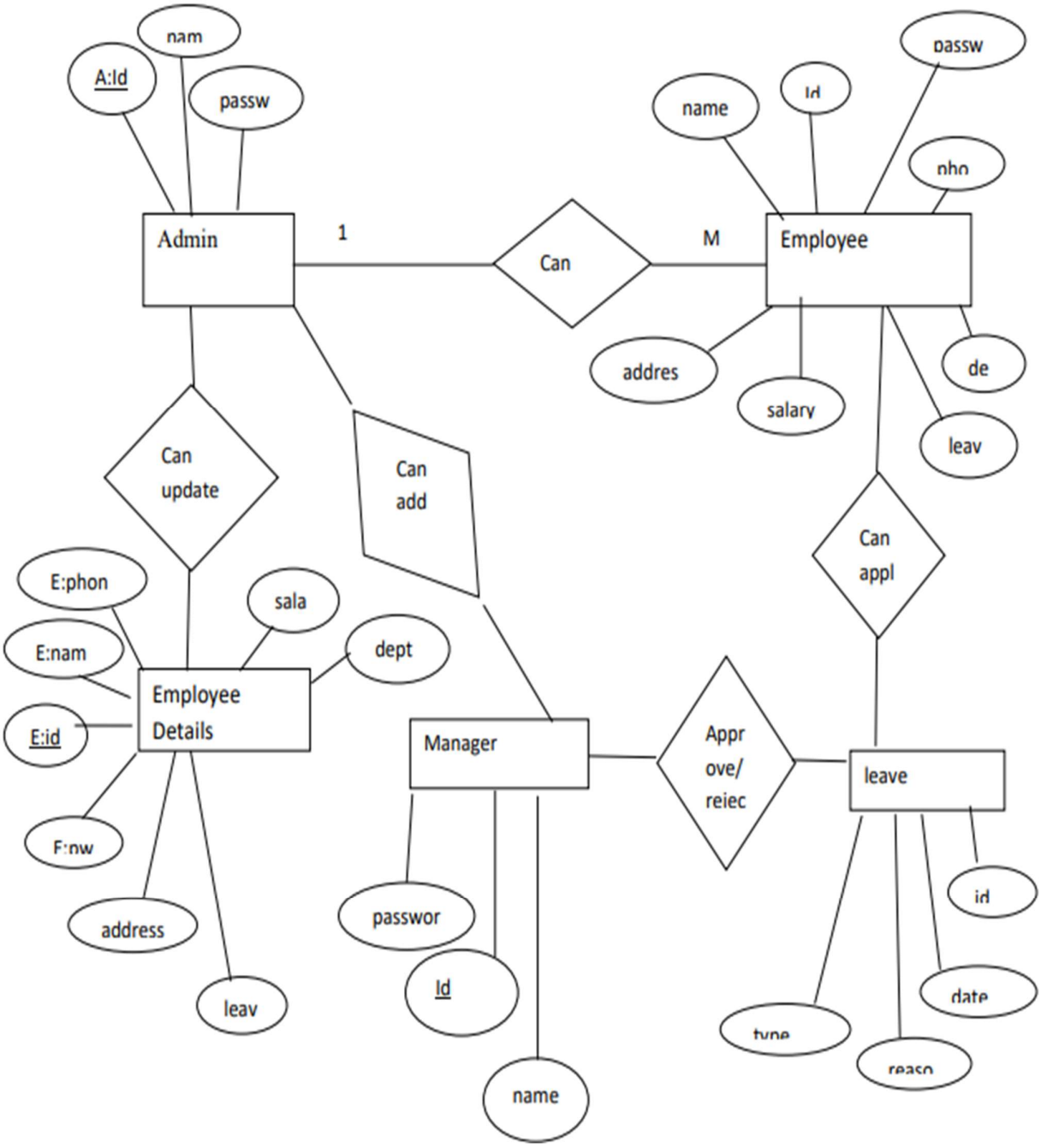
- The system should work across different web browsers (Chrome, Firefox, Edge) and be responsive on various devices (desktop, tablet, mobile).

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



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”

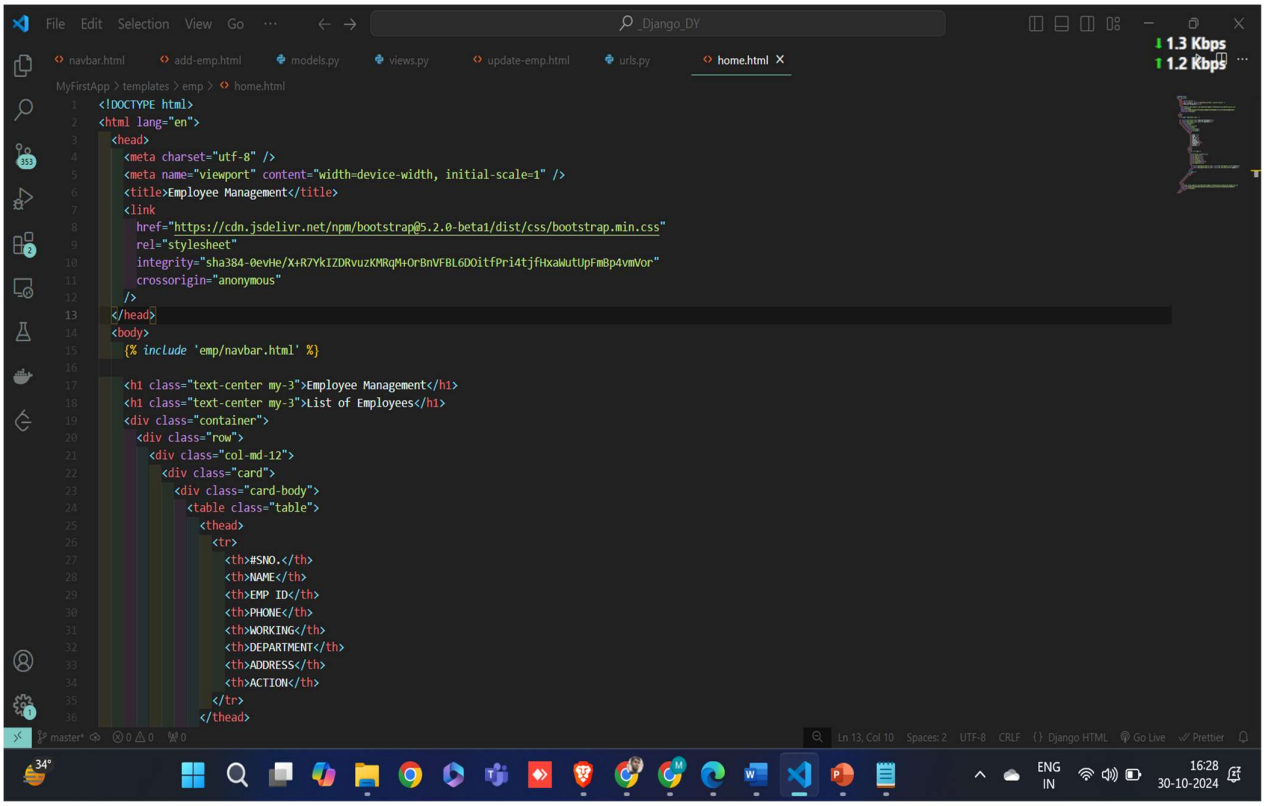
End if End

SYSTEM IMPLEMENTATION

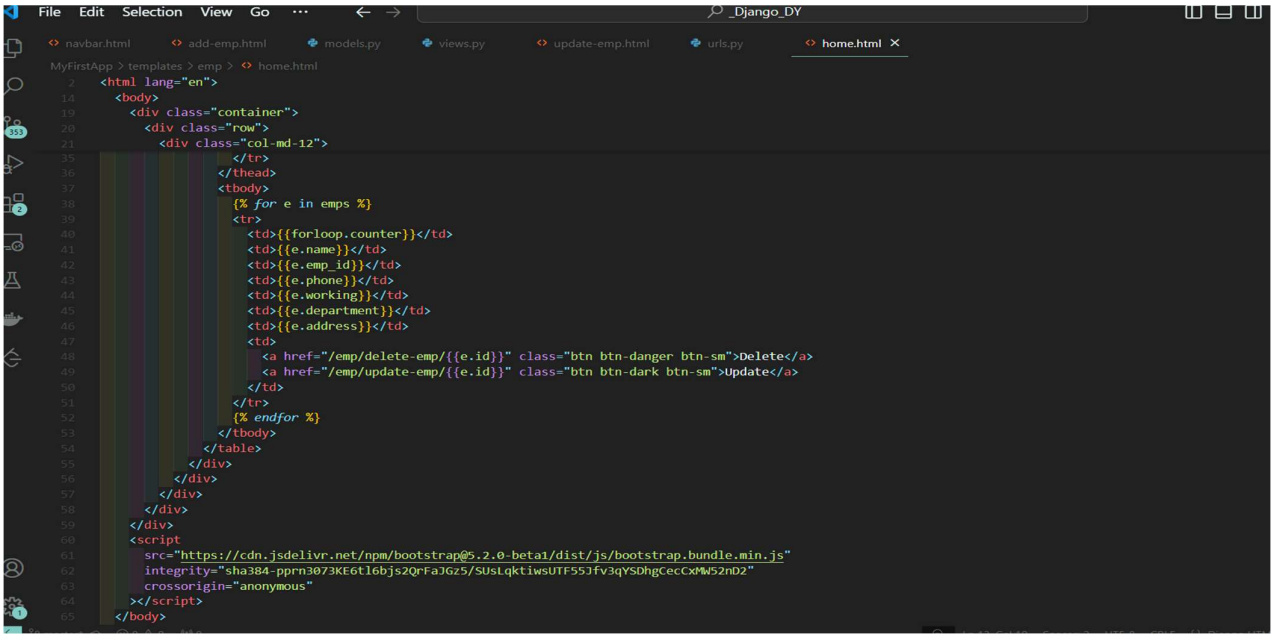
HOMEPAGE FOR EMPLOYEE MANAGEMENT SYSTEM

Coding:

home.html

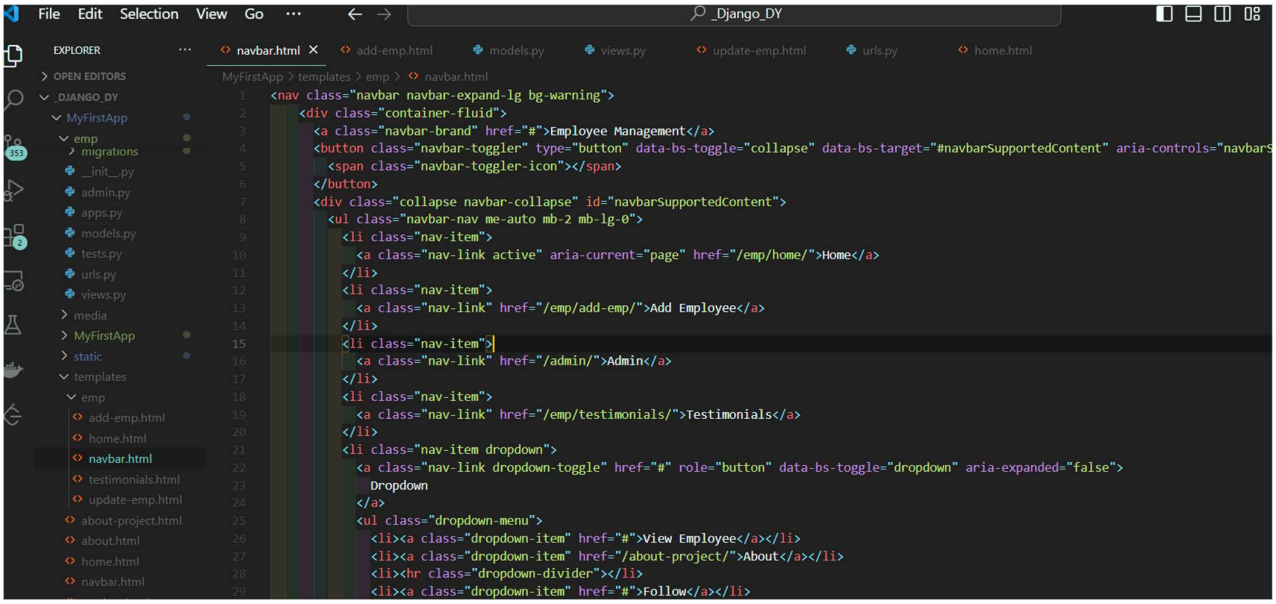


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Employee Management</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"
      rel="stylesheet"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

    <h1 class="text-center my-3">Employee Management</h1>
    <h1 class="text-center my-3">List of Employees</h1>
    <div class="container">
      <div class="row">
        <div class="col-md-12">
          <div class="card">
            <div class="card-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>
```



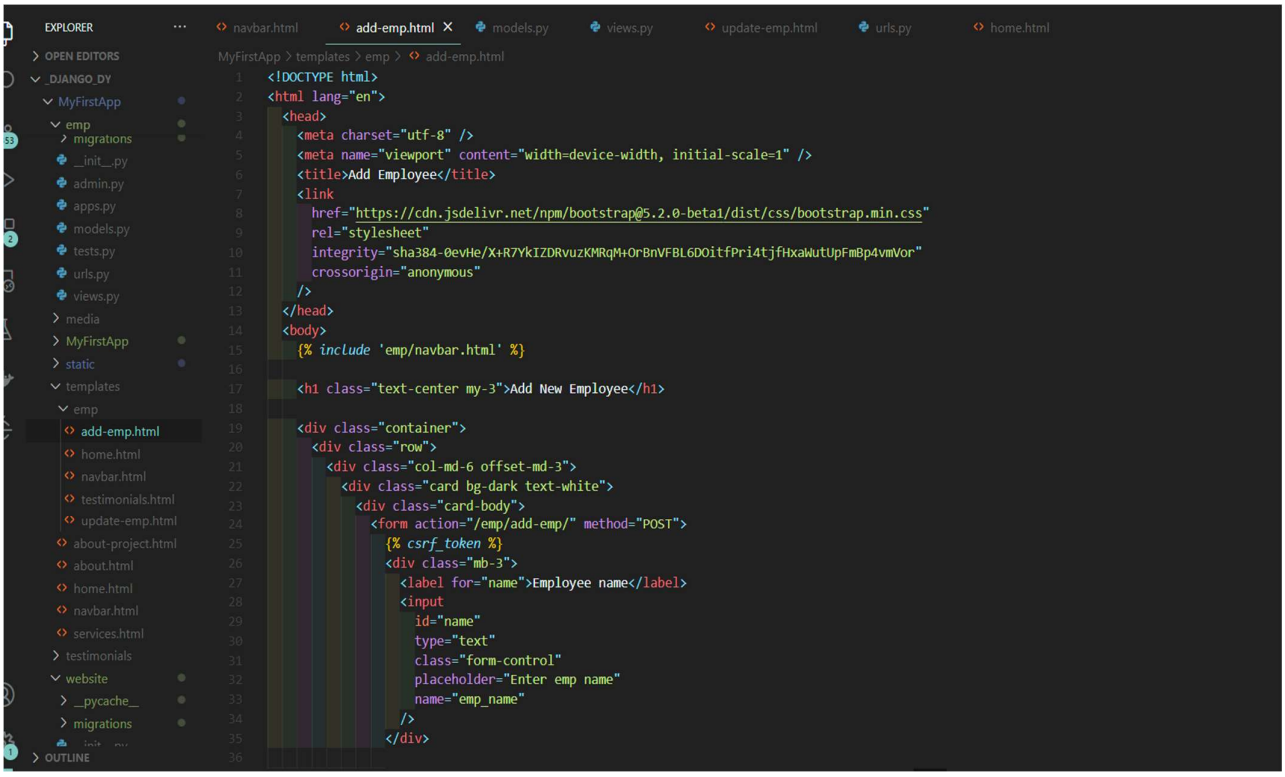
```
                <tbody>
                  {% for e in emps %}
                  <tr>
                    <td>{{forloop.counter}}</td>
                    <td>{{e.name}}</td>
                    <td>{{e.emp_id}}</td>
                    <td>{{e.phone}}</td>
                    <td>{{e.working}}</td>
                    <td>{{e.department}}</td>
                    <td>{{e.address}}</td>
                    <td>
                      <a href="/emp/delete-emp/{{e.id}}" class="btn btn-danger btn-sm">Delete</a>
                      <a href="/emp/update-emp/{{e.id}}" class="btn btn-dark btn-sm">Update</a>
                    </td>
                  </tr>
                  {% endfor %}
                </tbody>
              </table>
            </div>
          </div>
        </div>
      </div>
    </div>
    <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>
```

Navbnar.html



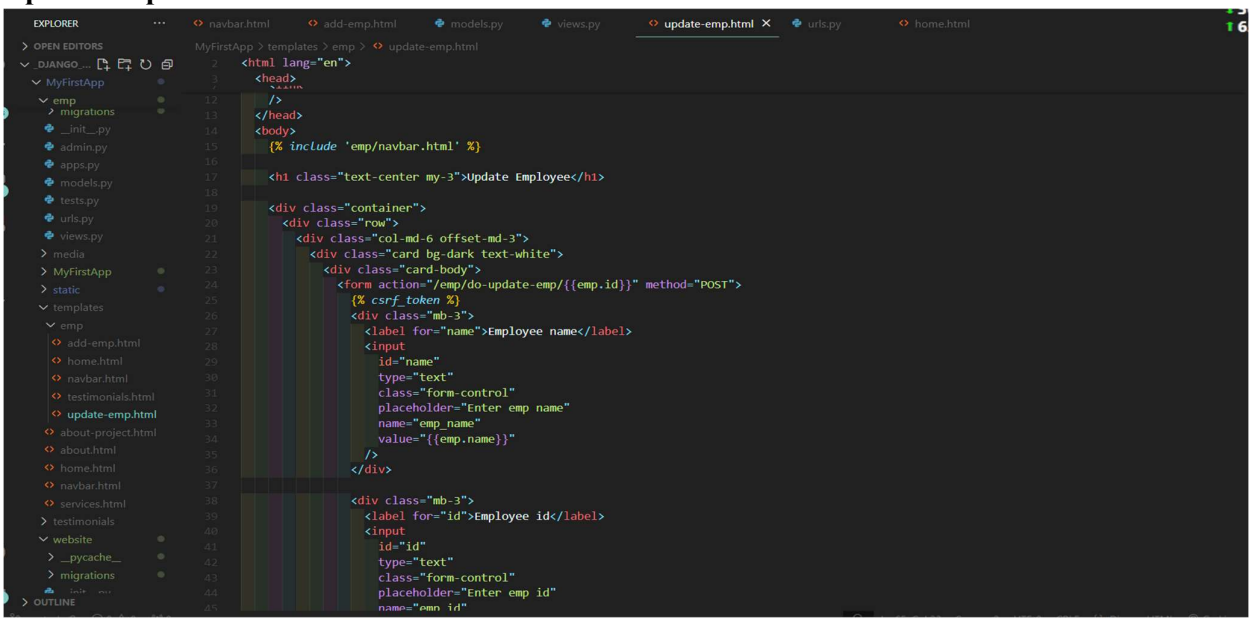
```
<nav class="navbar navbar-expand-lg bg-warning">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Employee Management</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarS
      <span class="navbar-toggler-icon"></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>
```

Add-emp.html

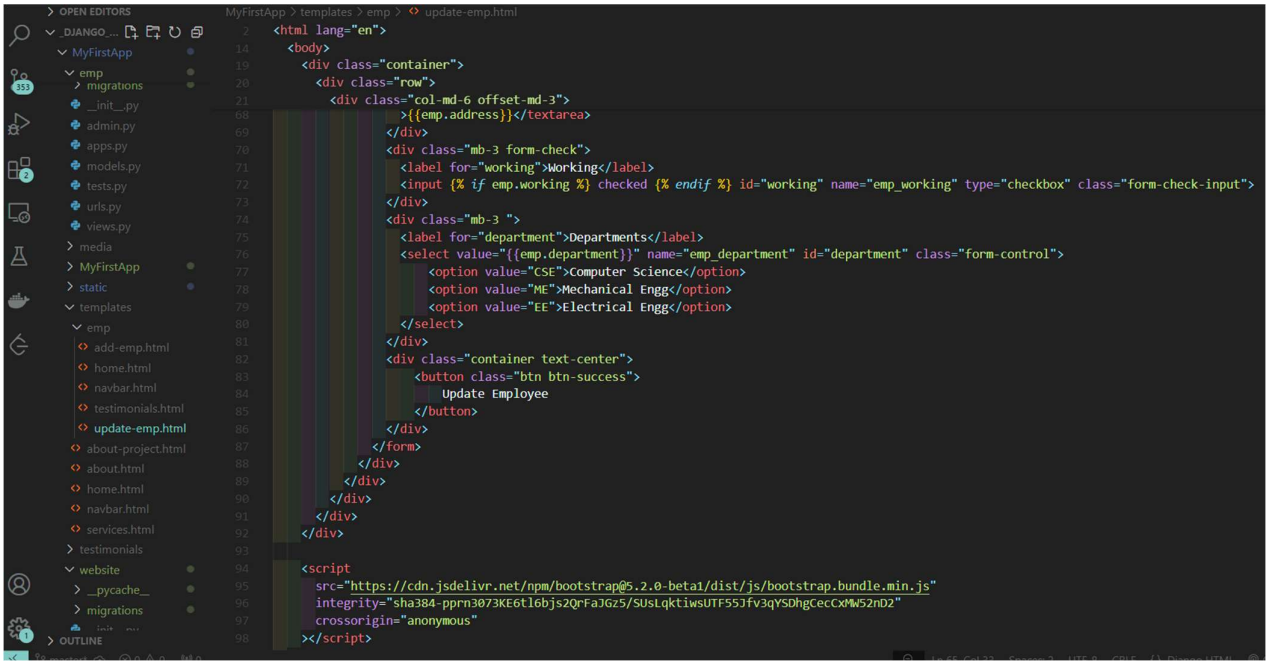


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Add Employee</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"
      rel="stylesheet"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

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

    <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/add-emp/" 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"
                  />
                </div>
```

Update-emp.html

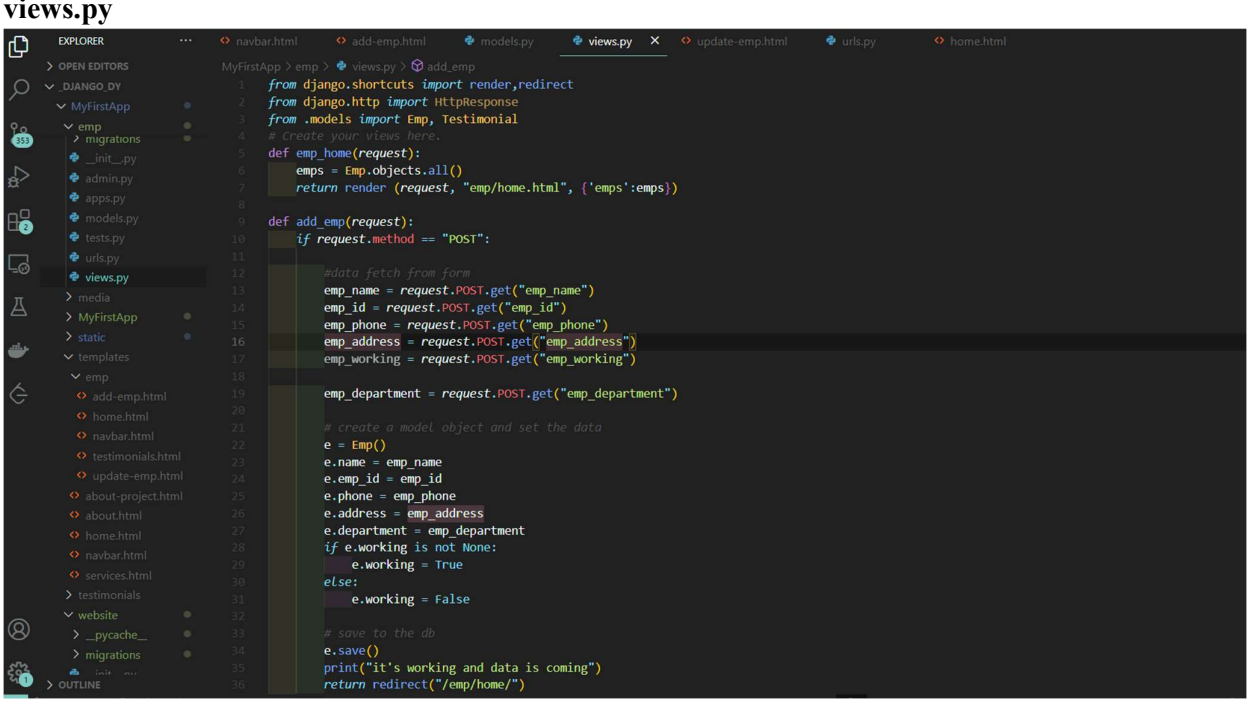


```
<html lang="en">
  <head>
    />
  </head>
  <body>
    {% include 'emp/navbar.html' %}

    <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
                    id="id"
                    type="text"
                    class="form-control"
                    placeholder="Enter emp id"
                    name="emp_id"
```

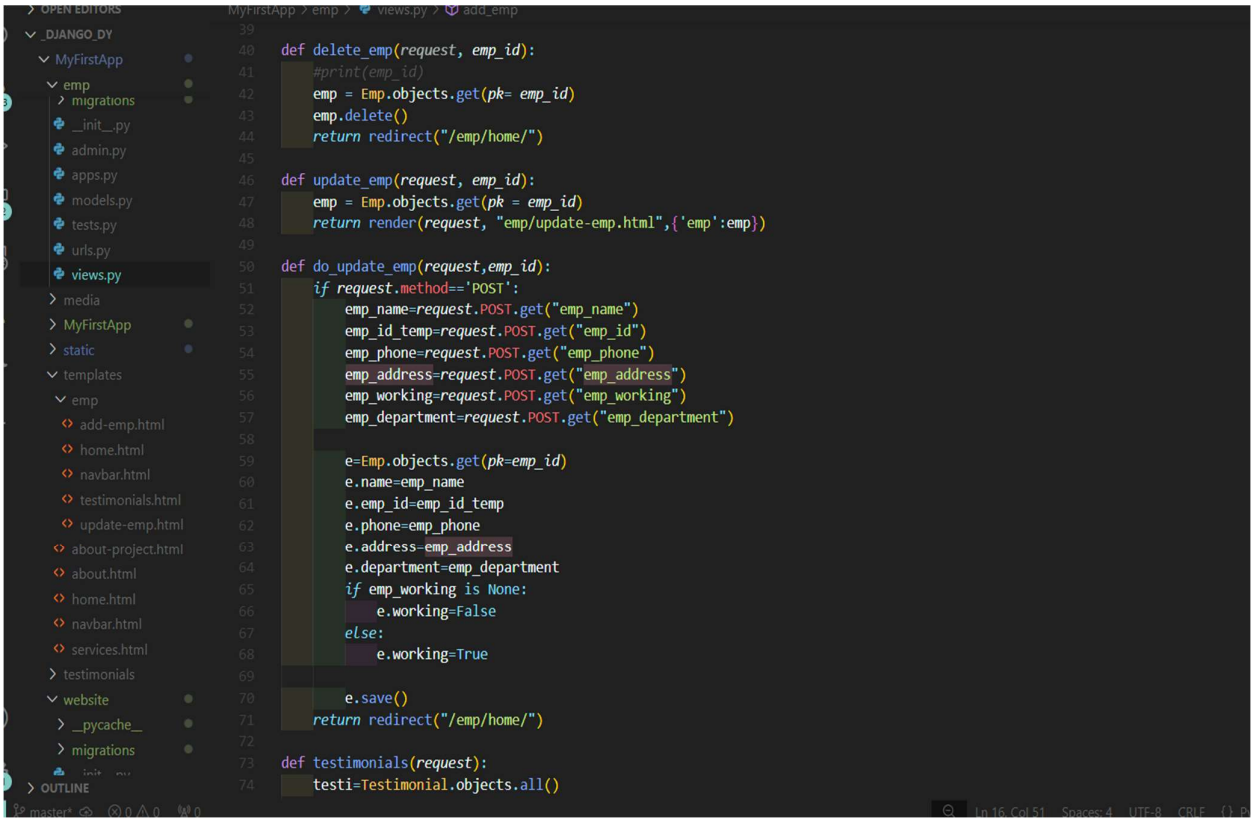


```
<html lang="en">
  <body>
    <div class="container">
      <div class="row">
        <div class="col-md-6 offset-md-3">
                    >{{emp.address}}</textarea>
                  </div>
                  <div class="mb-3 form-check">
                    <label for="working">Working</label>
                    <input {% if emp.working %} checked {% endif %} id="working" name="emp_working" type="checkbox" class="form-check-input">
                  </div>
                  <div class="mb-3 ">
                    <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>
                  </div>
                  <div class="container text-center">
                      <button class="btn btn-success">
                          Update Employee
                      </button>
                  </div>
                </form>
              </div>
            </div>
          </div>
        </div>
      </div>

    <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>
```

views.py

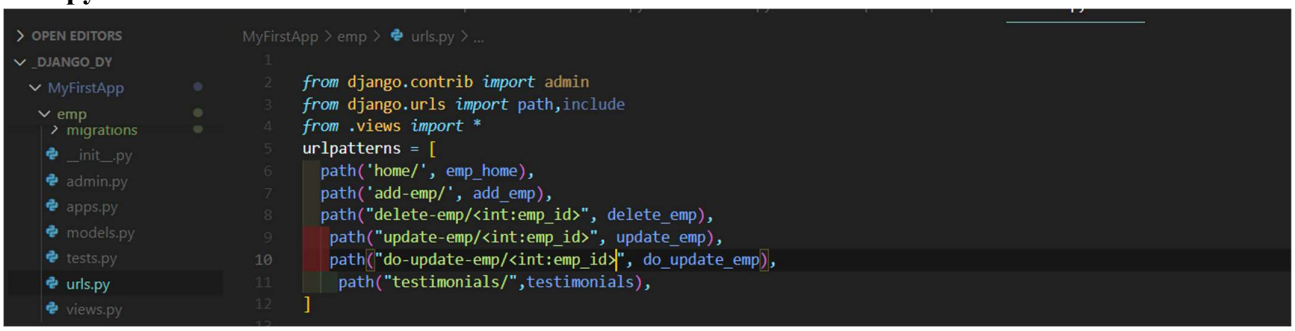


```python
from django.shortcuts import render,redirect
from django.http import HttpResponse
from .models import Emp, Testimonial
# Create your views here.
def emp_home(request):
    emps = Emp.objects.all()
    return render (request, "emp/home.html", {'emps':emps})

def add_emp(request):
    if request.method == "POST":

        #data fetch from form
        emp_name = request.POST.get("emp_name")
        emp_id = request.POST.get("emp_id")
        emp_phone = request.POST.get("emp_phone")
        emp_address = request.POST.get("emp_address")
        emp_working = request.POST.get("emp_working")

        emp_department = request.POST.get("emp_department")

        # create a model object and set the data
        e = Emp()
        e.name = emp_name
        e.emp_id = emp_id
        e.phone = emp_phone
        e.address = emp_address
        e.department = emp_department
        if e.working is not None:
            e.working = True
        else:
            e.working = False

        # save to the db
        e.save()
        print("it's working and data is coming")
        return redirect("/emp/home/")
```

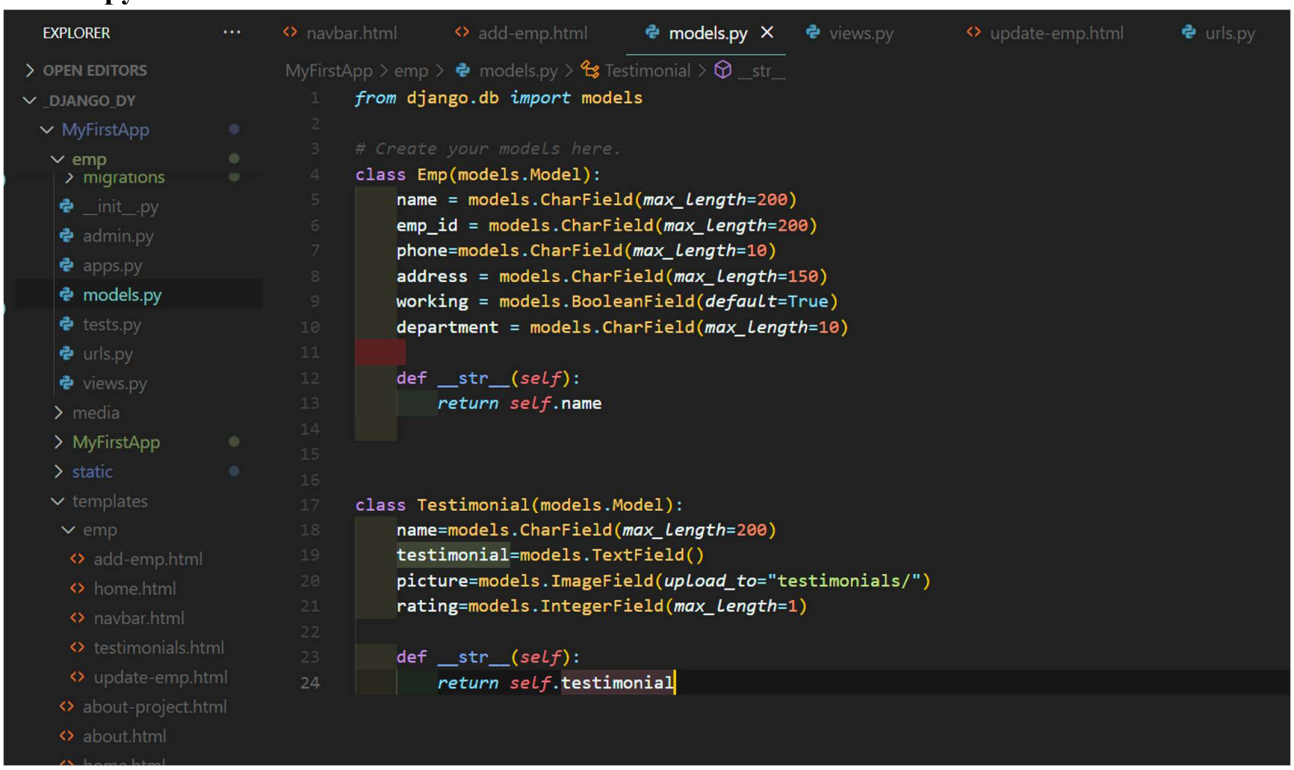


```python

def delete_emp(request, emp_id):
    #print(emp_id)
    emp = Emp.objects.get(pk= emp_id)
    emp.delete()
    return redirect("/emp/home/")

def update_emp(request, emp_id):
    emp = Emp.objects.get(pk = emp_id)
    return render(request, "emp/update-emp.html",{'emp':emp})

def do_update_emp(request,emp_id):
    if request.method=='POST':
        emp_name=request.POST.get("emp_name")
        emp_id_temp=request.POST.get("emp_id")
        emp_phone=request.POST.get("emp_phone")
        emp_address=request.POST.get("emp_address")
        emp_working=request.POST.get("emp_working")
        emp_department=request.POST.get("emp_department")

        e=Emp.objects.get(pk=emp_id)
        e.name=emp_name
        e.emp_id=emp_id_temp
        e.phone=emp_phone
        e.address=emp_address
        e.department=emp_department
        if emp_working is None:
            e.working=False
        else:
            e.working=True

        e.save()
    return redirect("/emp/home/")

def testimonials(request):
    testi=Testimonial.objects.all()
```

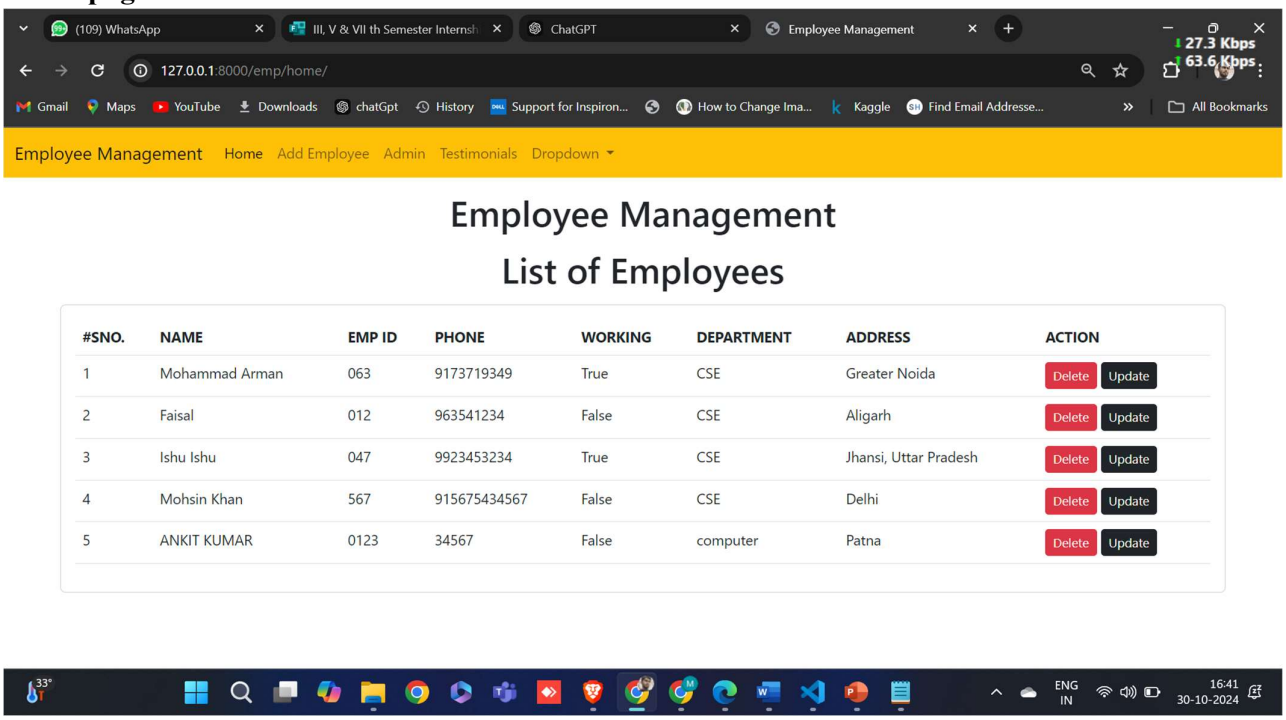
Urls.py



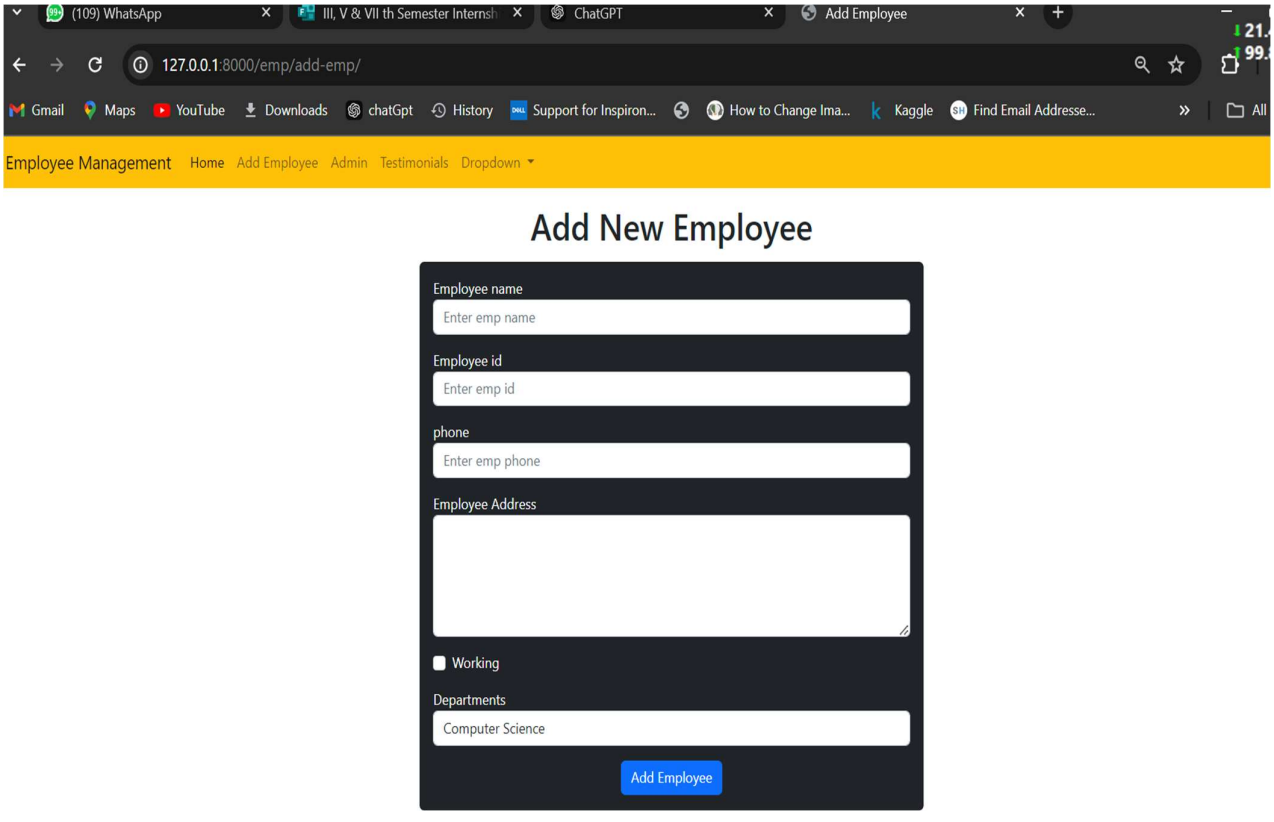
Models.py



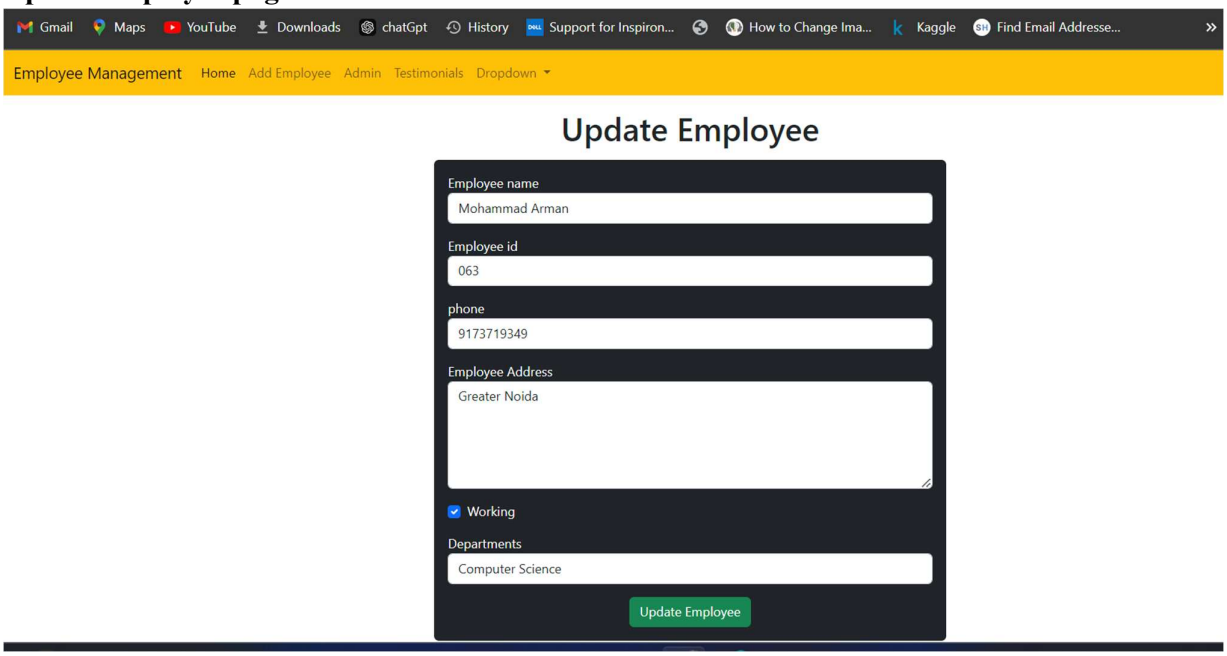
Home page



Add employee page



Update employee page



Admin page