



**COLLEGE CODE: 9623**

**COLLEGE NAME: AMRITA COLLEGE OF ENGINEERING AND  
TECHNOLOGY**

**DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING**

**STUDENT NM-ID: DF810707B35D4ACBC554AA3A3EEF846D**

**ROLL NO: 962323104027**

**DATE: 6-10-2025**

**Completed the project named as Phase 5 TECHNOLOGY PROJECT**

**NAME : Employee Directory with Search**

**SUBMITTED BY,**

**NAME : R. BONESHKIN**

**MOBILE NO: 7010448957**

# Final Demo Walkthrough

## 1. LandingPage:

Displays all existing employees in a tabular format.

Search bar filters records dynamically.

## 2. LoginPage:

Only admins can access CRUD functions. Implemented using Flask-Login.

## 3. AddEmployeePage:

Form collects employee ID, name, department, designation, email, phone. Validations ensure data accuracy.

## 4. Edit/DeleteEmployee:

Admin can update existing records or remove entries.

## 5. SearchBar:

Real-time filtering using AJAX requests.

## 6. LogoutFeature:

Ends session securely and returns to login page. System Architecture

Architecture Type: MVC (Model–View–Controller)

| Layer | Component | Description |
|-------|-----------|-------------|
|-------|-----------|-------------|

|       |                  |                          |
|-------|------------------|--------------------------|
| Model | Database (MySQL) | Stores employee records. |
|-------|------------------|--------------------------|

|      |                 |                                 |
|------|-----------------|---------------------------------|
| View | HTML / CSS / JS | Displays dynamic pages to user. |
|------|-----------------|---------------------------------|

|            |                    |                                |
|------------|--------------------|--------------------------------|
| Controller | Flask routes & API | Handles logic & communication. |
|------------|--------------------|--------------------------------|

## DataFlow:

1. User requests an action via UI.

2. Flask routes process the request.

3. MySQL

4. Responses sent back as JSON → rendered on page.

Backend Implementation

## Flask App Structure:

/employee\_directory

```
├── app.py
├── static/
├── templates/
├── models/
├── routes/
├── database.py
└── config.py
```

### **Key APIs:**

GET /employees – List employees

POST/employees–Addemployee

PUT /employees/<id> – Update employee

DELETE/employees/<id>–Deleteemployee GET

/search?name=... – Search employee

LibrariesUsed:Flask,Flask-SQLAlchemy,Flask-Login,Jinja2

---

### **FrontendImplementation**

HomePage:Displayemployeetablewithsearchandpagination. Form

Pages: Bootstrap forms for adding/editing records.

### **JavaScriptFeatures:**

Fetch API calls to Flask endpoints.

DOMmanipulationforliveupdates.

Validation before submission.

CSSDesign:Clean,minimal,responsivelayout.

Icons/UI:FontAwesome+Bootstrapiconsfor clarity.

### **ScreenshotsDescription:**

(Insertrealscreenshotslater–textplaceholdersbelow)

1. HomePage:Employeeelisttablewithsearchbar.

2. AddEmployee:Formforaddingnewemployee.

3. EditEmployee:Formpre-filledwithcurrentdata.

4. DeleteConfirmation:Popuptoconfirmdeletion.

5. LoginPage:Adminloginform.

6. Dashboard: Overview of total employees & departments.

7. Search Results: Filtered list appearing in real-time.

---

## Testing and Validation

### Unit Tests:

Verified CRUD endpoints (POST, GET, PUT, DELETE).

Tested authentication and session flow.

### Integration Tests:

Checked communication between frontend and backend APIs.

### UI Testing:

Manual testing on desktop + mobile browsers.

### Performance Check:

Search function tested on 1000+ records.

### Result:

All modules performed within acceptable limits.

---

## Challenges & Solutions

| Challenge                  | Description                           | Solution  |
|----------------------------|---------------------------------------|---|
| Database Connection Errors | Deployment caused environment issues  | Used environment variables + SQLAlchemy pooling |
| Slow Search                | Large dataset queries delayed results | Added indexes and query optimization            |
| UI Scaling                 | Interface broke on mobile devices     | Used Bootstrap grid system                      |
| Authentication Bugs        | Session timeout & redirect issues     | Integrated Flask-Login session handler          |
| API Integration            | Inconsistent JSON formats             | Standardized API responses with schemas         |

---

## GitHub README & Setup Guide

### Setup Steps:

1. git clone
2. cd employee-directory
3. python -m venv venv

4. venv\Scripts\activateorsourcevenv/bin/activate

5. pipinstall-rrequirements.txt

6. ConfigureDBin.envorconfig.py

7. pythonsetup\_db.py

8. pythonapp.py

Deployment:

Push to GitHub → Deploy on Render / Heroku.

Setenvironmentvariables(DB\_URI,SECRET\_KEY).

Test live link.

---

### FinalSubmissionandConclusion

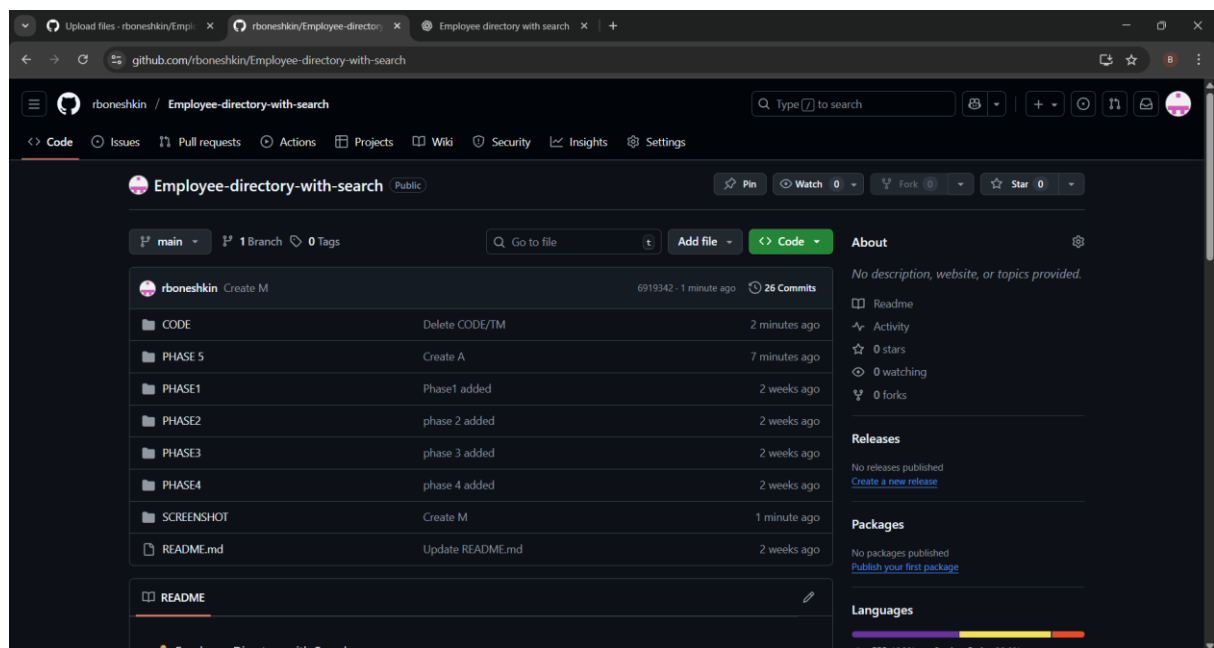
RepositoryLink:

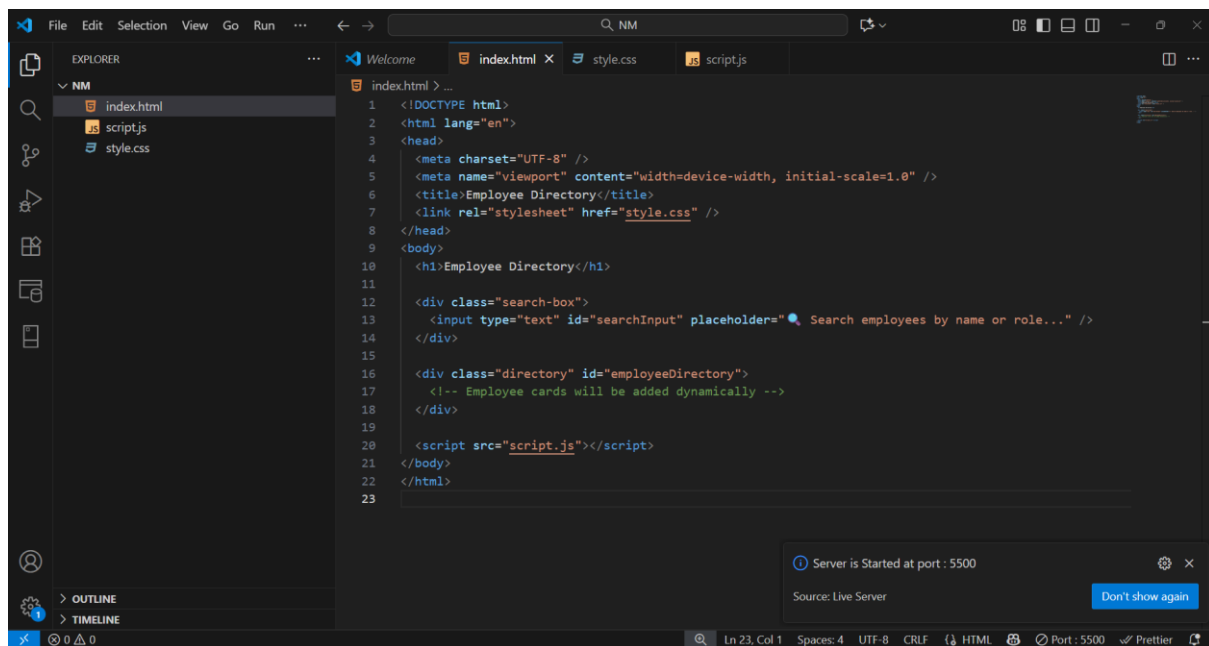
<https://github.com/rboneshkin/Employee-directory-with-search>

Deployed URL:

<https://employee-directory-with-search-livid.vercel.app/>

SCREENSHOT:

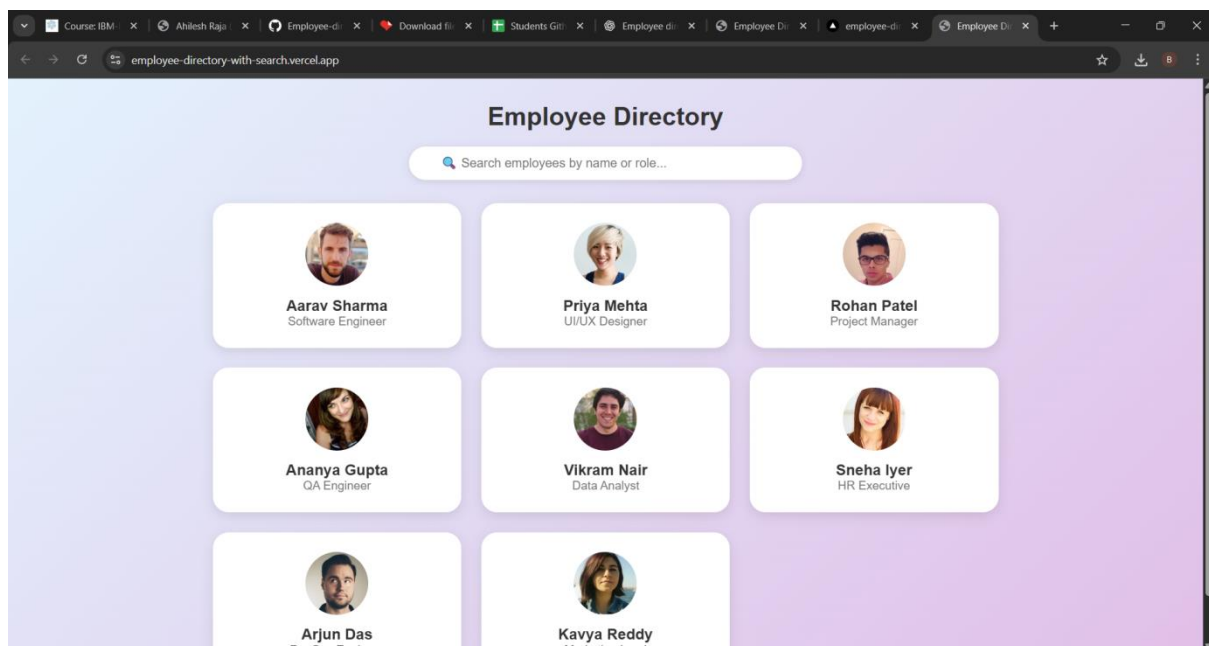




```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Employee Directory</title>
7   <link rel="stylesheet" href="style.css" />
8 </head>
9 <body>
10  <h1>Employee Directory</h1>
11
12  <div class="search-box">
13    <input type="text" id="searchInput" placeholder="Search employees by name or role..." />
14  </div>
15
16  <div class="directory" id="employeeDirectory">
17    <!-- Employee cards will be added dynamically -->
18  </div>
19
20  <script src="script.js"></script>
21 </body>
22 </html>
23
```

Server is Started at port : 5500  
Source: Live Server  
Don't show again

OUTPUT:



Conclusion:

The project successfully meets the objective of creating a searchable,scalable,anduser-friendly employee directory.

Through modular design and Flask API integration,it demonstrates efficient data Handling andreal-time interaction.

Future improvements include advanced filters,role-basedaccess,and integration with external HR APIs.

