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. Landing Page:

Displays all existing employees in a tabular format.

Search bar filters records dynamically.

2. Login Page:

Only admins can access CRUD functions.

Implemented using Flask-Login.

3. Add Employee Page:

Form collects employee ID, name, department, designation, email, phone.

Validations ensure data accuracy.

4. Edit / Delete Employee:

Admin can update existing records or remove entries.

5. Search Bar:

Real-time filtering using AJAX requests.

6. Logout Feature:

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.

Data Flow:

1. User requests an action via UI.

2. Flask routes process the request.

3. MySQL q

4. Response 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 – Add employee

PUT /employees/<id> – Update employee

DELETE /employees/<id> – Delete employee

GET /search?name=... – Search employee

Libraries Used: Flask, Flask-SQLAlchemy, Flask-Login, Jinja2

Frontend Implementation

Home Page: Displays employee table with search and pagination.

Form Pages: Bootstrap forms for adding/editing records.

JavaScript Features:

Fetch API calls to Flask endpoints.

DOM manipulation for live updates.

Validation before submission.

CSS Design: Clean, minimal, responsive layout.

Icons / UI: Font Awesome + Bootstrap icons for clarity.

Screenshots Description:

(Insert real screenshots later – text placeholders below)

1. Home Page: Employee list table with search bar.

2. Add Employee: Form for adding new employee.

3. Edit Employee: Form pre-filled with current data.

4. Delete Confirmation: Popup to confirm deletion.

5. Login Page: Admin login form.

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 https://github.com/username/employee-directory

2. cd employee-directory

3. python -m venv venv

4. venv\Scripts\activate or source venv/bin/activate

5. pip install -r requirements.txt

6. Configure DB in .env or config.py

7. python setup_db.py

8. python app.py

Deployment:

Push to GitHub → Deploy on Render / Heroku.

Set environment variables (DB_URI, SECRET_KEY).

Test live link.

Final Submission and Conclusion

Repository Link:

👉 https://github.com/username/employee-directory

Deployed URL:

👉 https://employee-directory-demo.onrender.com

Demo Credentials:

Username: admin

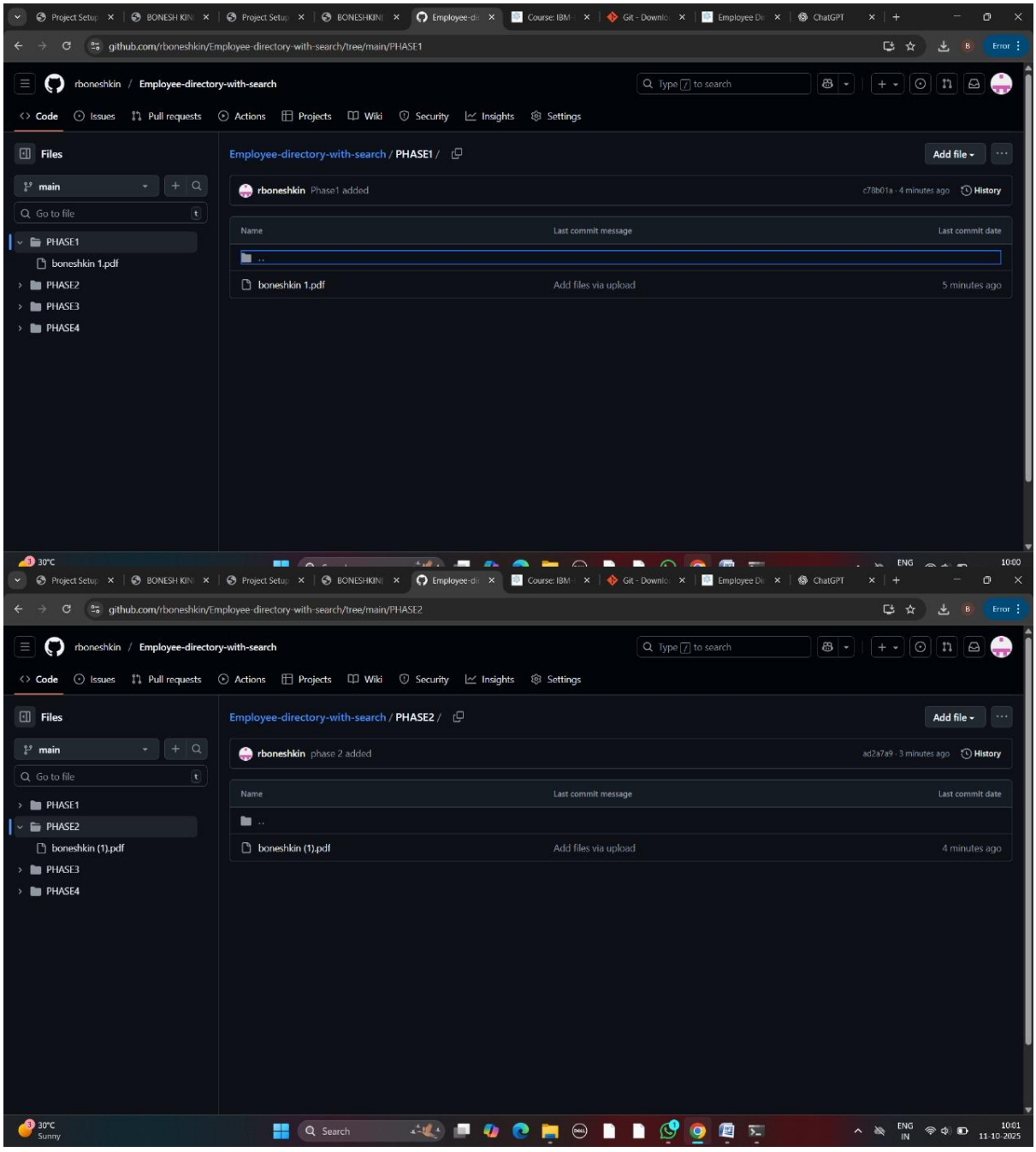
Password: admin123

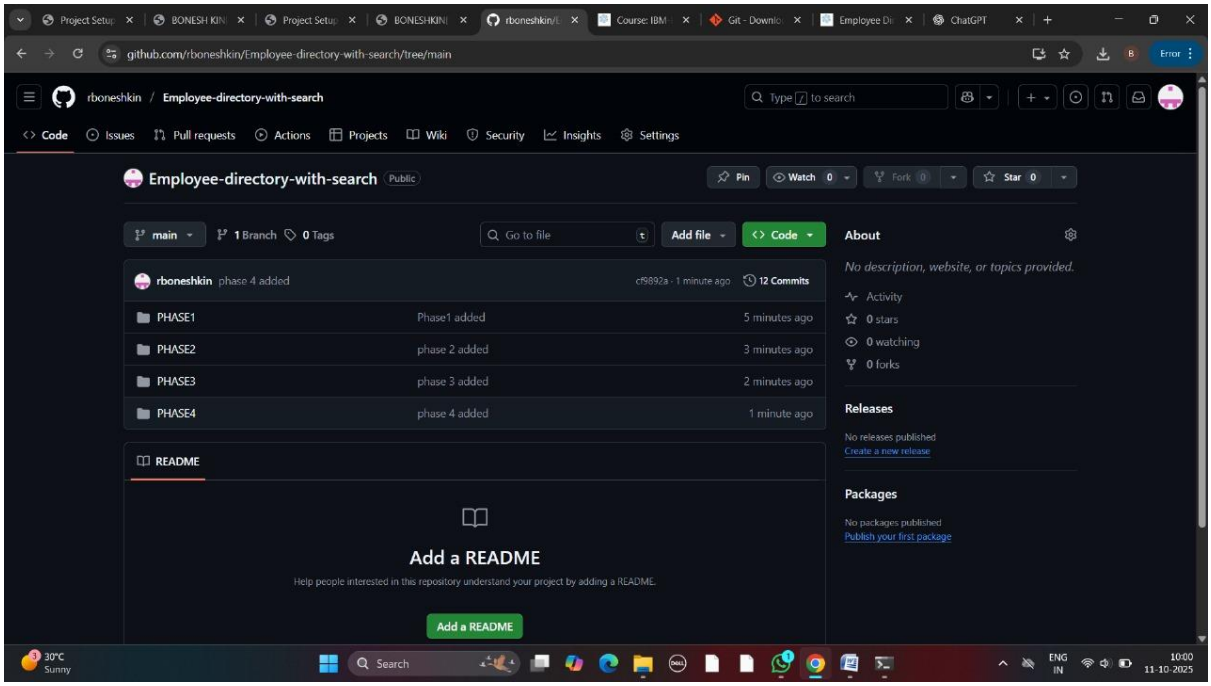
Conclusion:

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

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

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


github.com/rboneshkin/Employee-directory-with-search/tree/main/PHASE1
rboneshkin / Employee-directory-with-search
Code Issues Pull requests Actions Projects Wiki Security Insights Settings
Files
main
Go to file
PHASE1
boneshkin 1.pdf
PHASE2
PHASE3
PHASE4
Employee-directory-with-search / PHASE1 /
Add file
rboneshkin Phase1 added
c78b01a · 4 minutes ago
History
Name
Last commit message
Last commit date
..
boneshkin 1.pdf
Add files via upload
5 minutes ago
github.com/rboneshkin/Employee-directory-with-search/tree/main/PHASE2
rboneshkin / Employee-directory-with-search
Code Issues Pull requests Actions Projects Wiki Security Insights Settings
Files
main
Go to file
PHASE1
PHASE2
boneshkin (1).pdf
PHASE3
PHASE4
Employee-directory-with-search / PHASE2 /
Add file
rboneshkin phase 2 added
ad2a7a9 · 3 minutes ago
History
Name
Last commit message
Last commit date
..
boneshkin (1).pdf
Add files via upload
4 minutes ago
30°C
Sunny
Search
ENG
IN
10:01
11-10-2025



GITHUB:https://github.com/username/employee-directory