EMPLOYEE, MANAGER & PROJECT MANAGEMENT SYSTEM

ISHEET HARISH SHETTY

G01384428

Contents

INTRODUCTION

In the fast-paced and dynamic landscape of contemporary workplaces, effective management of human resources and projects is crucial for organizational success. Introducing the Employee, Manager, and Project Management System—a comprehensive solution designed to streamline and enhance the efficiency of workforce management and project execution.
Essentially the three main components will have a unique role which comes together or an effective management of the entire system. The key role of each component are as follows:

1. Employee Data Management:

This system provides a centralized hub for managing employee data. It will contain data mainly related to the personal information of the employee like ID, name, position, and the department in which the employee is assigned or works in.

2. Manager Data Management:

The Manager module offers insights into team dynamics, and the information related to the head or the manager of each department or the manager assigned to a particular project.
It will mainly contain information like the name of the manager, the managerial role/position like head / tech coordinator / Dean of the department which they are assigned and the department the which they are managing.

3. Project Data Management:

The Project Component consists of the data related to projects which are present in the company like the name of the project, the status of the project, the department the project belongs to, the employe who are assigned and working for the project and finally the manager ID who is managing the project at various levels.

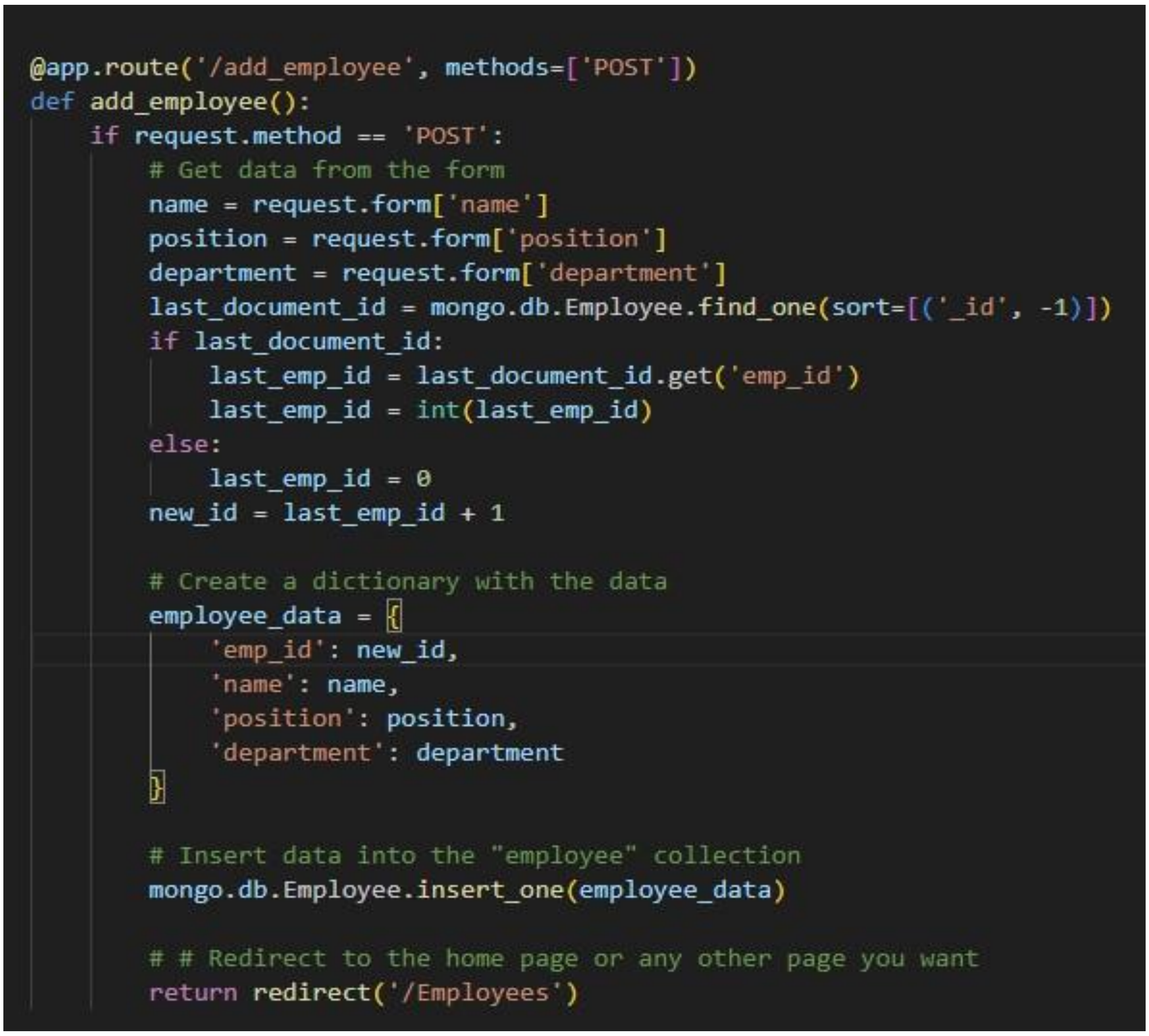
With the Employee, Manager, and Project Management System, organizations can harness the power of integration to create a cohesive and efficient work environment. This system not only simplifies daily operations but also empowers stakeholders at all levels to contribute to the overall success of the organization.

DATA

The system will have three main collections: Employee, Manager and Project. The data structure for the Employee, Manager, and Project Management System would involve organized collections and attributes for each component

Employee

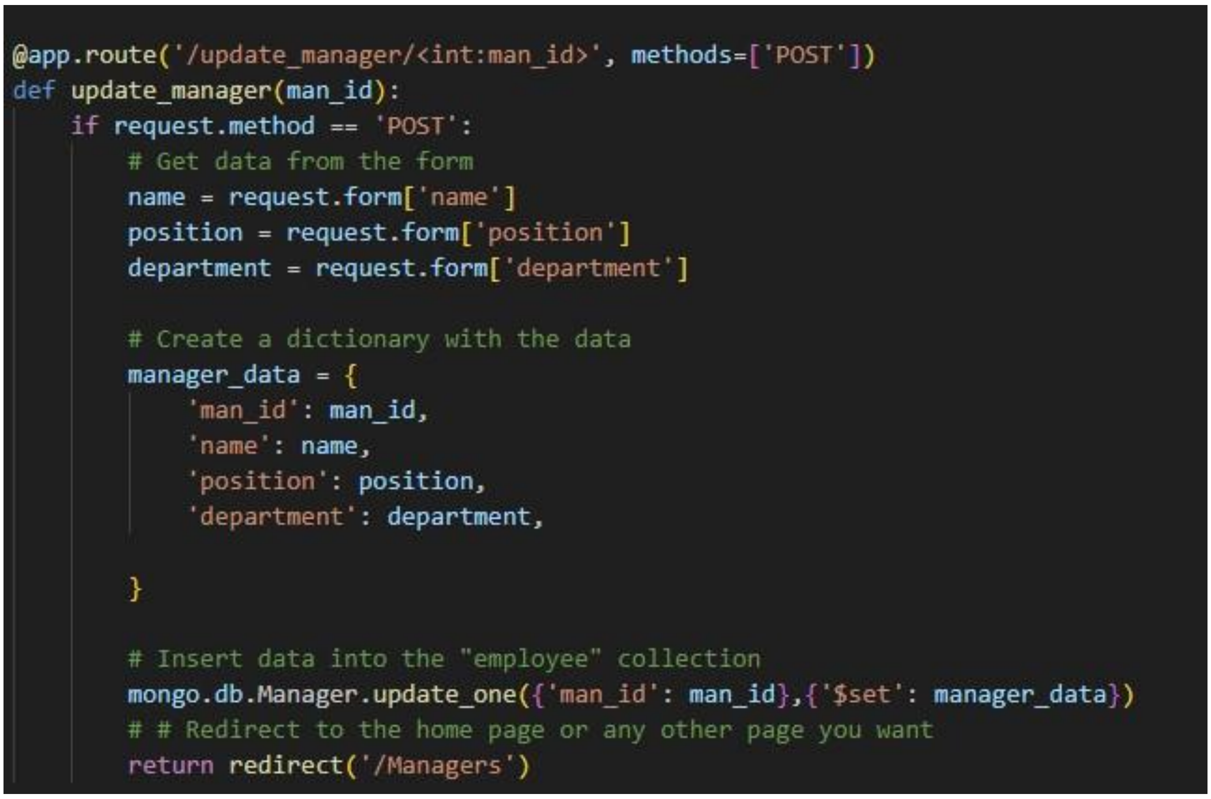
- emp_id: An integer to uniquely identify each employee.
- name: String which contains the name of the employee.
- position: String which indicates the position the Employee is working in.
- department: String which indicates the department in which the employee works.



```
@app.route('/add_employee', methods=['POST'])
def add_employee():
    if request.method == 'POST':
        # Get data from the form
        name = request.form['name']
        position = request.form['position']
        department = request.form['department']
        last_document_id = mongo.db.Employee.find_one(sort=[('_id', -1)])
        if last_document_id:
            last_emp_id = last_document_id.get('emp_id')
            last_emp_id = int(last_emp_id)
        else:
            last_emp_id = 0
        new_id = last_emp_id + 1

        # Create a dictionary with the data
        employee_data = {
            'emp_id': new_id,
            'name': name,
            'position': position,
            'department': department
        }

        # Insert data into the "employee" collection
        mongo.db.Employee.insert_one(employee_data)

        # # Redirect to the home page or any other page you want
        return redirect('/Employees')
```

Manager

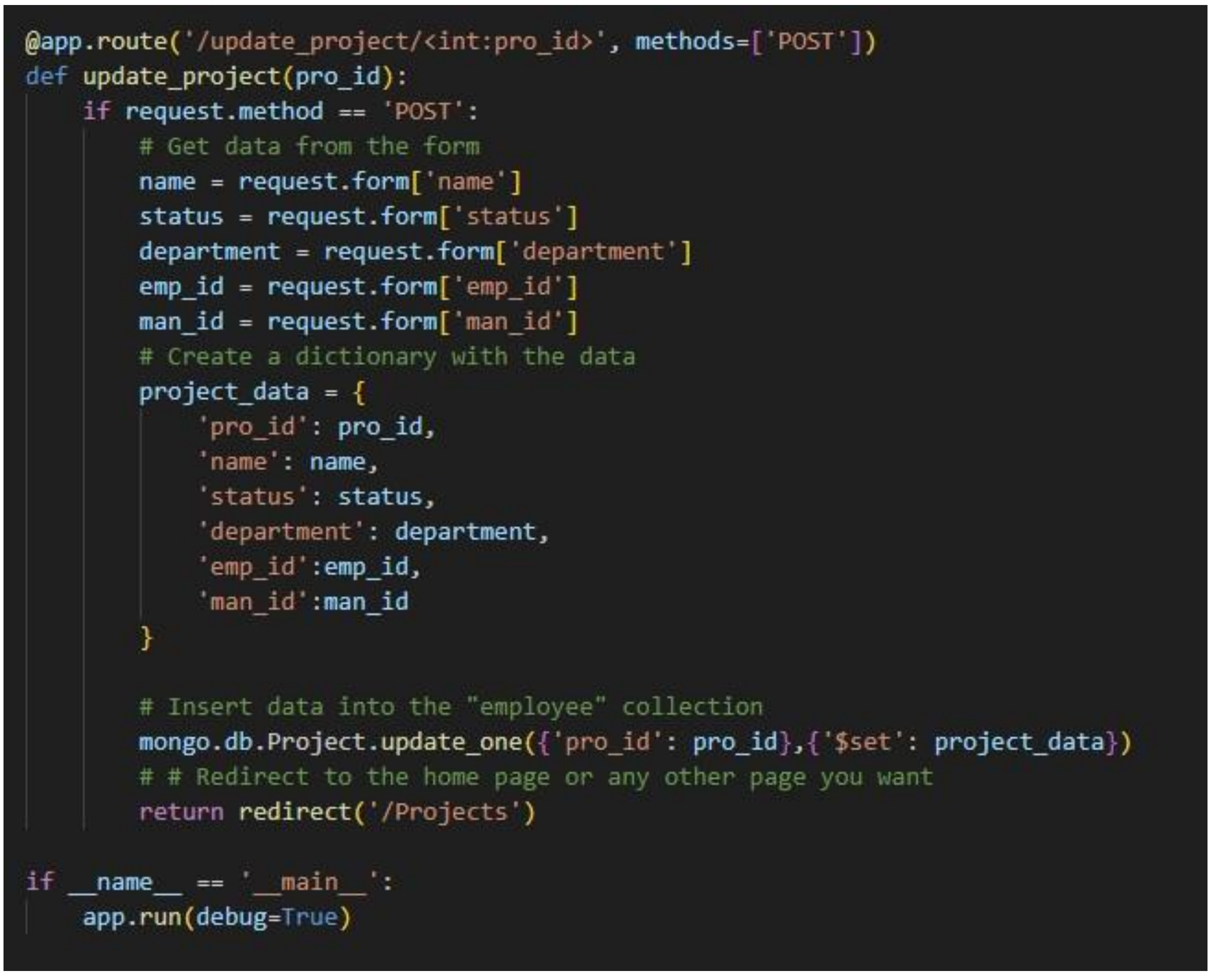
- man_id: An integer to uniquely identify each manager.
- name: String which contains the name of the manager.
- position: String which indicates the position the Manager is working in.
- department: String which indicates the department in which the manager works.



```python
@app.route('/update_manager/<int:man_id>', methods=['POST'])
def update_manager(man_id):
    if request.method == 'POST':
        # Get data from the form
        name = request.form['name']
        position = request.form['position']
        department = request.form['department']

        # Create a dictionary with the data
        manager_data = {
            'man_id': man_id,
            'name': name,
            'position': position,
            'department': department,

        }

        # Insert data into the "employee" collection
        mongo.db.Manager.update_one({'man_id': man_id},{'$set': manager_data})
        # # Redirect to the home page or any other page you want
        return redirect('/Managers')
```

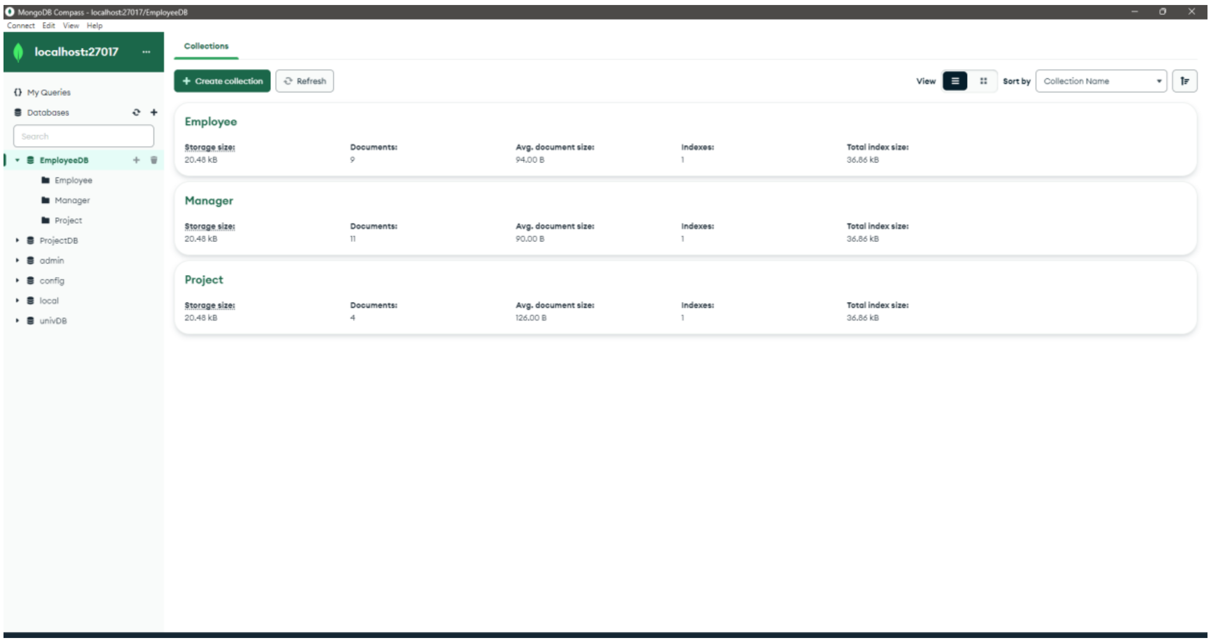
Project

- pro_id: An integer to uniquely identify each project.
- name: String which contains the name of the project.
- status: String which indicates the status of the project like running, closed.
- department: String which indicates the department the project belongs to.
- emp_id : indicating the employee who is working on the project
- pro_id : indicating the manager who is in charge of the project



```python
@app.route('/update_project/<int:pro_id>', methods=['POST'])
def update_project(pro_id):
    if request.method == 'POST':
        # Get data from the form
        name = request.form['name']
        status = request.form['status']
        department = request.form['department']
        emp_id = request.form['emp_id']
        man_id = request.form['man_id']
        # Create a dictionary with the data
        project_data = {
            'pro_id': pro_id,
            'name': name,
            'status': status,
            'department': department,
            'emp_id':emp_id,
            'man_id':man_id
        }

        # Insert data into the "employee" collection
        mongo.db.Project.update_one({'pro_id': pro_id},{'$set': project_data})
        # # Redirect to the home page or any other page you want
        return redirect('/Projects')

if __name__ == '__main__':
    app.run(debug=True)
```

Screenshot of database collection in MongoDB compass:



MongoDB Compass - localhost:27017/EmployeeDB

Employee

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	9	94.00 B	1	36.86 kB

Manager

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	11	90.00 B	1	36.86 kB

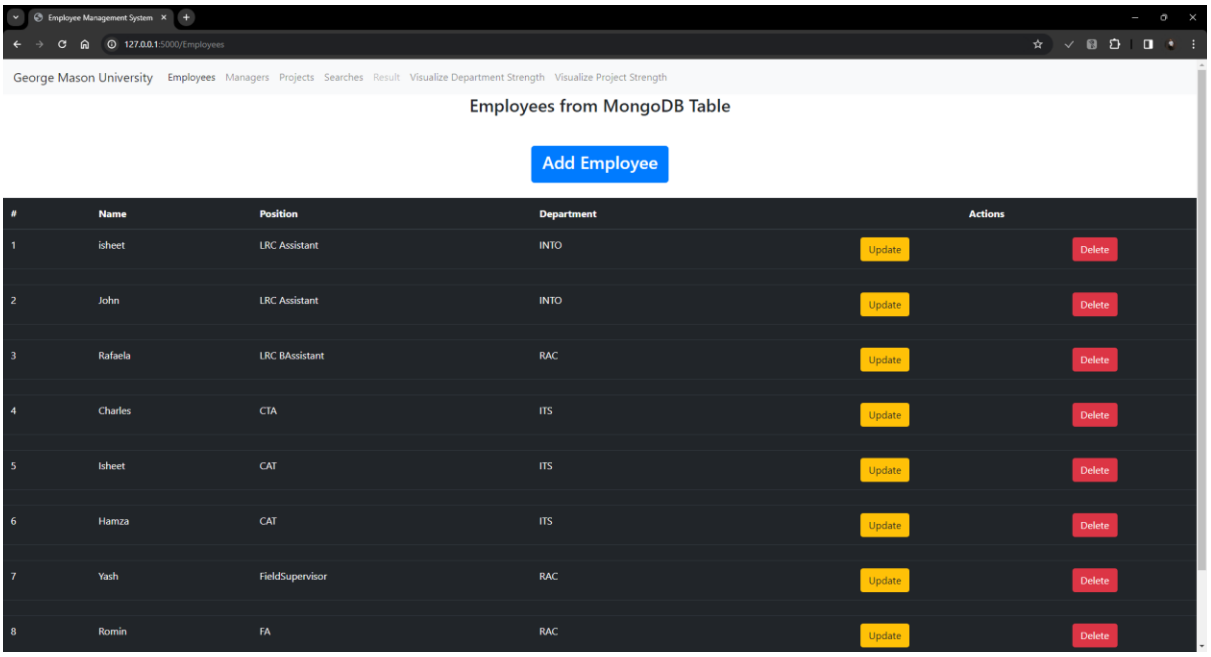
Project

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	4	126.00 B	1	36.86 kB

CRUD FUNCTIONALITIES

Employee:

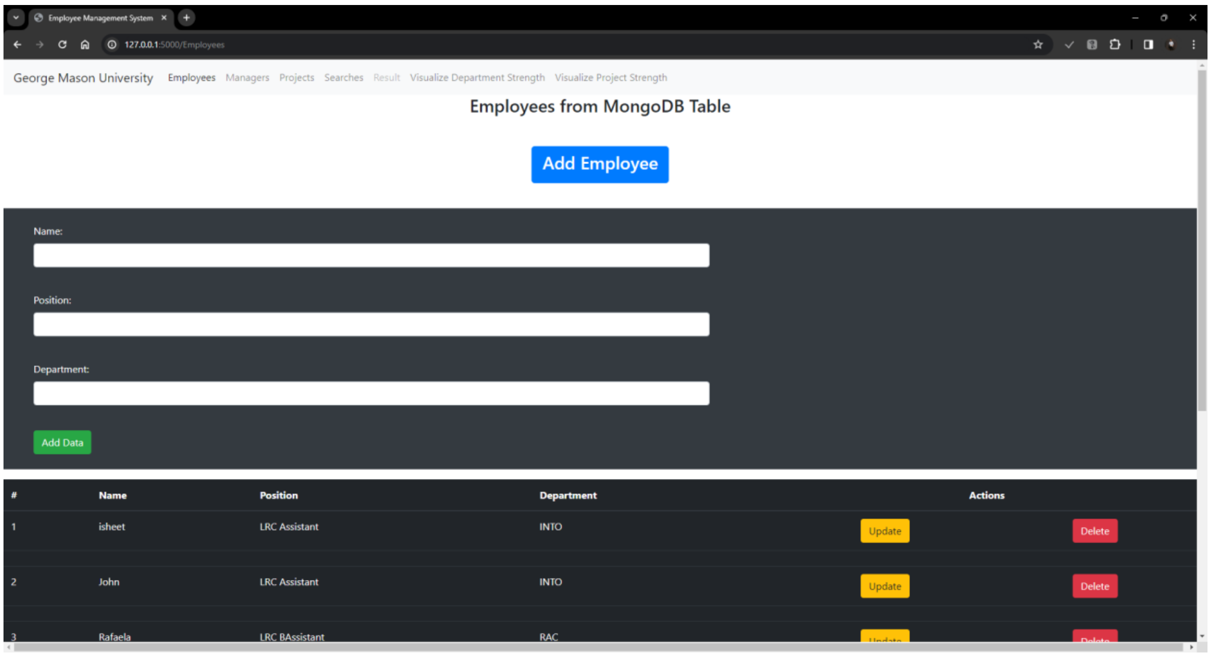
View Employee:



Employees from MongoDB Table

#	Name	Position	Department	Actions	
1	isheet	LRC Assistant	INTO	Update	Delete
2	John	LRC Assistant	INTO	Update	Delete
3	Rafaela	LRC BAssistant	RAC	Update	Delete
4	Charles	CTA	ITS	Update	Delete
5	Isheet	CAT	ITS	Update	Delete
6	Hamza	CAT	ITS	Update	Delete
7	Yash	FieldSupervisor	RAC	Update	Delete
8	Romin	FA	RAC	Update	Delete

Add Employee:



Employees from MongoDB Table

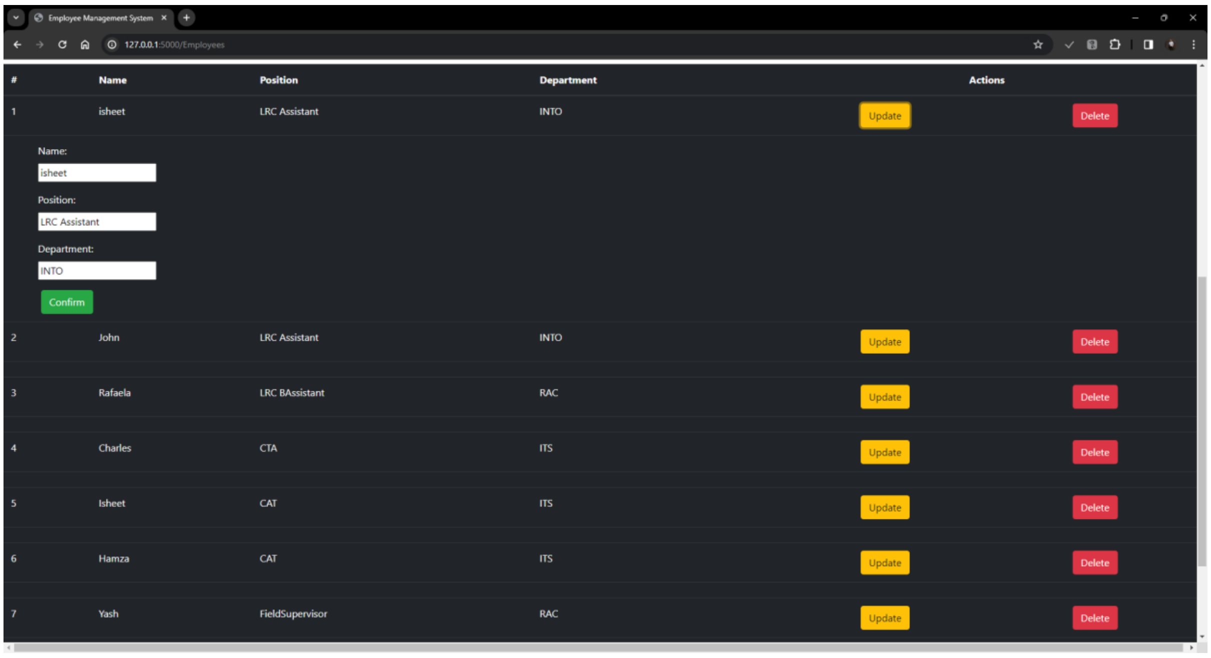
Name:

Position:

Department:

#	Name	Position	Department	Actions	
1	isheet	LRC Assistant	INTO	Update	Delete
2	John	LRC Assistant	INTO	Update	Delete
3	Rafaela	LRC BAssistant	RAC		

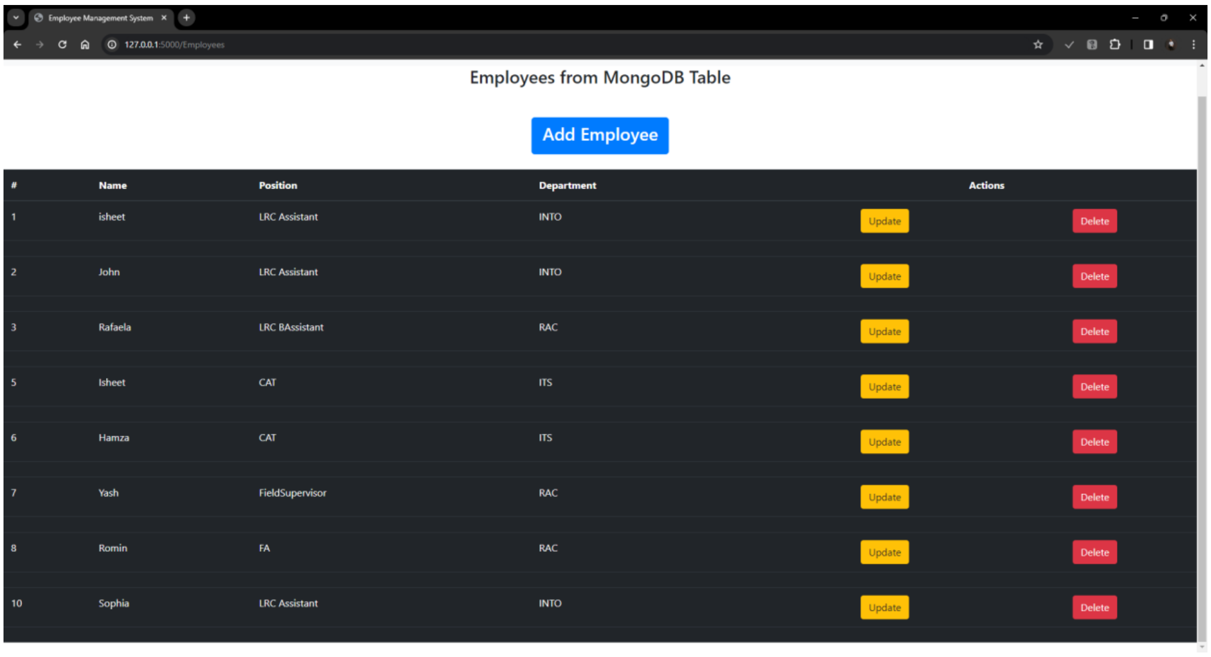
Update Employee:



#	Name	Position	Department	Actions	
1	isheet	LRC Assistant	INTO	Update	Delete
2	John	LRC Assistant	INTO	Update	Delete
3	Rafaela	LRC BAssistant	RAC	Update	Delete
4	Charles	CTA	ITS	Update	Delete
5	Isheet	CAT	ITS	Update	Delete
6	Hamza	CAT	ITS	Update	Delete
7	Yash	FieldSupervisor	RAC	Update	Delete

Name: isheet
Position: LRC Assistant
Department: INTO
Confirm

Delete Employee:



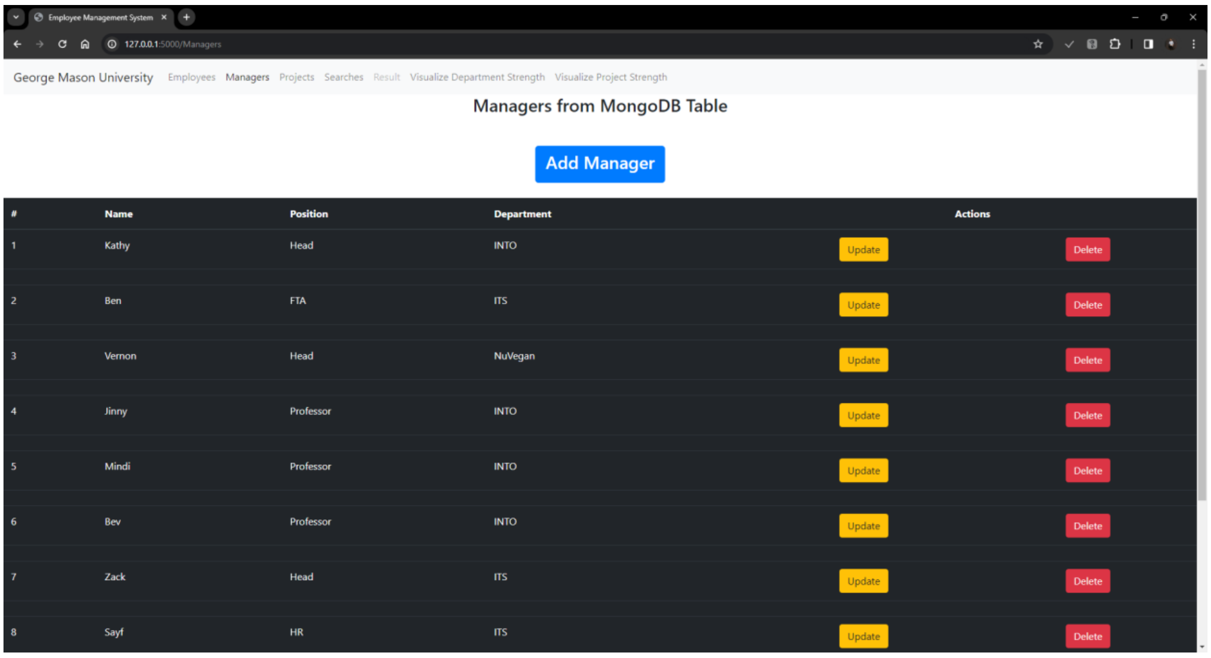
Employees from MongoDB Table

Add Employee

#	Name	Position	Department	Actions	
1	isheet	LRC Assistant	INTO	Update	Delete
2	John	LRC Assistant	INTO	Update	Delete
3	Rafaela	LRC BAssistant	RAC	Update	Delete
5	Isheet	CAT	ITS	Update	Delete
6	Hamza	CAT	ITS	Update	Delete
7	Yash	FieldSupervisor	RAC	Update	Delete
8	Romin	FA	RAC	Update	Delete
10	Sophia	LRC Assistant	INTO	Update	Delete

Manager:

View Manager:


Employee Management System

127.0.0.1:5000/Managers

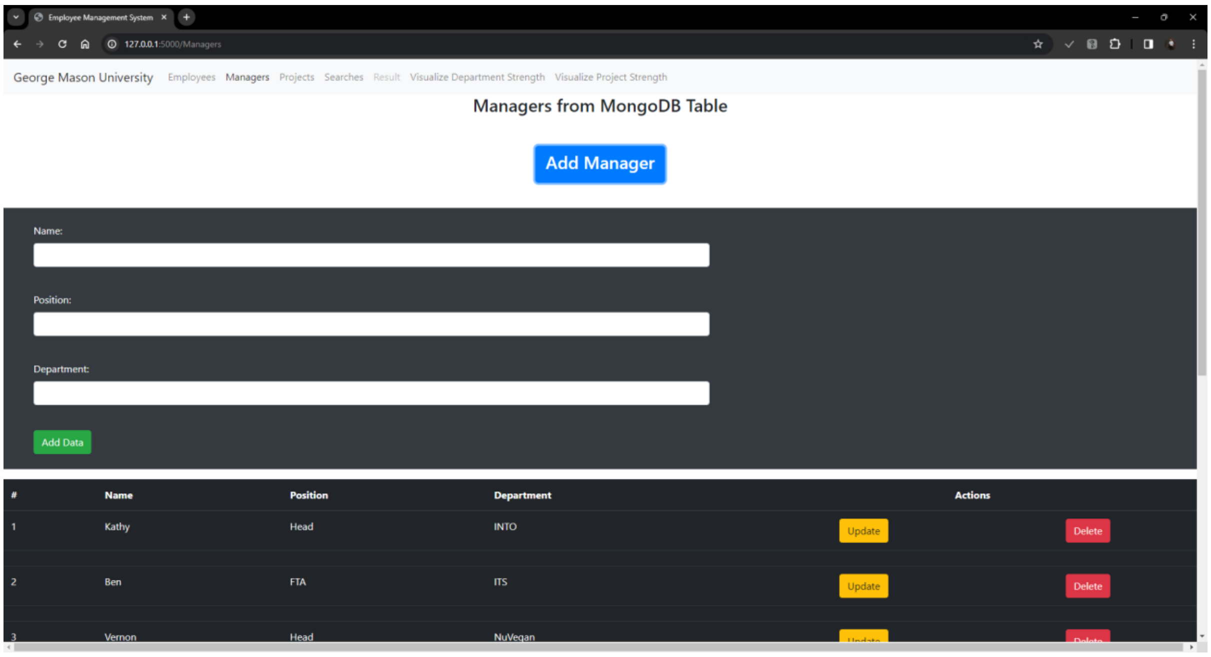
George Mason University Employees Managers Projects Searches Result Visualize Department Strength Visualize Project Strength

Managers from MongoDB Table

Add Manager

#	Name	Position	Department	Actions	
1	Kathy	Head	INTO	Update	Delete
2	Ben	FTA	ITS	Update	Delete
3	Vernon	Head	NuVegan	Update	Delete
4	Jinny	Professor	INTO	Update	Delete
5	Mindi	Professor	INTO	Update	Delete
6	Bev	Professor	INTO	Update	Delete
7	Zack	Head	ITS	Update	Delete
8	Sayf	HR	ITS	Update	Delete

Add Manager


Employee Management System

127.0.0.1:5000/Managers

George Mason University Employees Managers Projects Searches Result Visualize Department Strength Visualize Project Strength

Managers from MongoDB Table

Add Manager

Name:

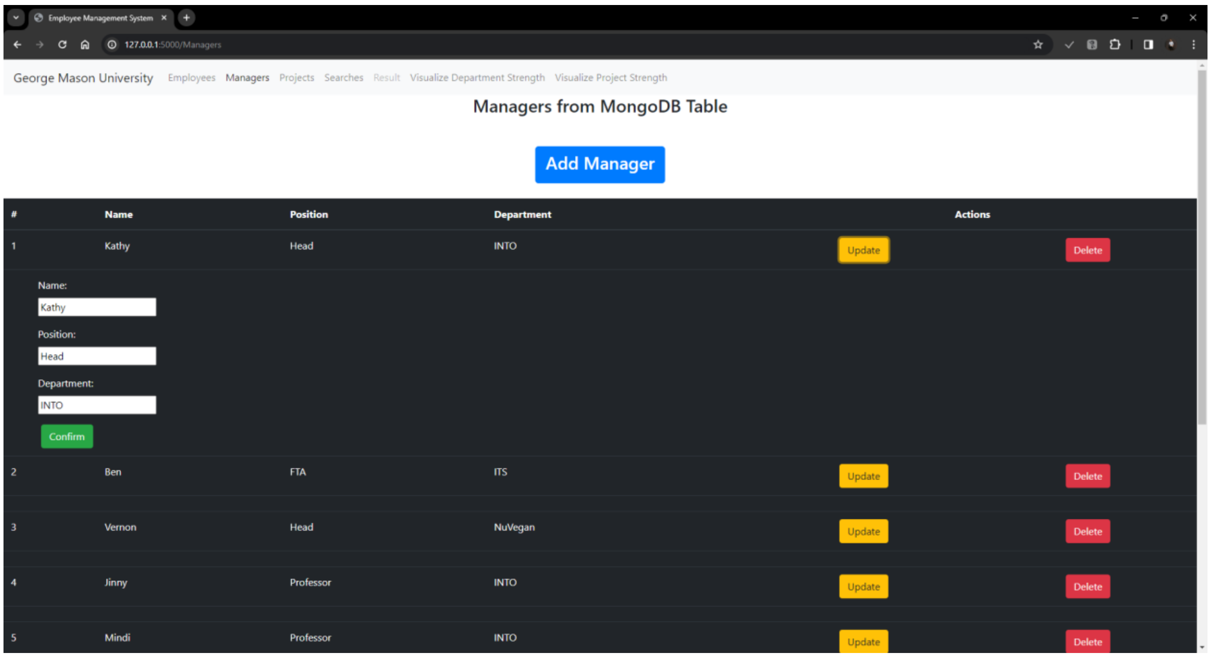
Position:

Department:

Add Data

#	Name	Position	Department	Actions	
1	Kathy	Head	INTO	Update	Delete
2	Ben	FTA	ITS	Update	Delete
3	Vernon	Head	NuVegan		

Update Manager



Employee Management System

127.0.0.1:5000/Managers

George Mason University Employees Managers Projects Searches Result Visualize Department Strength Visualize Project Strength

Managers from MongoDB Table

Add Manager

#	Name	Position	Department	Actions	
1	Kathy	Head	INTO	Update	Delete

Name: Kathy

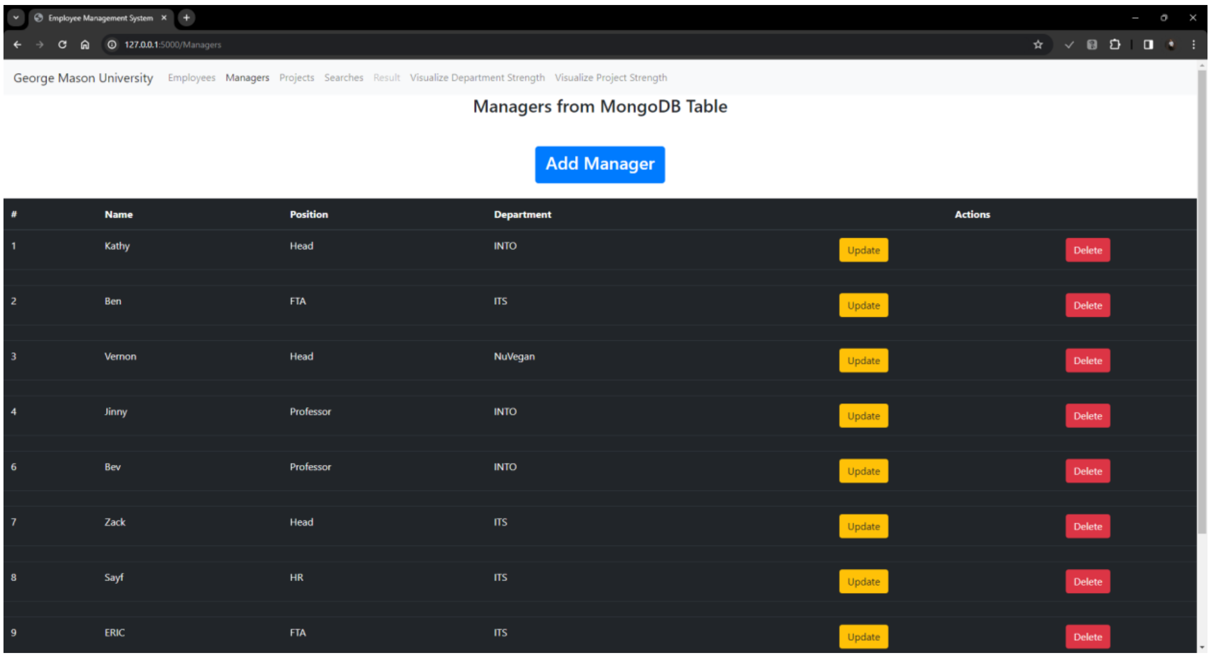
Position: Head

Department: INTO

Confirm

#	Name	Position	Department	Actions	
2	Ben	FTA	ITS	Update	Delete
3	Vernon	Head	NuVegan	Update	Delete
4	Jinny	Professor	INTO	Update	Delete
5	Mindi	Professor	INTO	Update	Delete

Delete Manager



Employee Management System

127.0.0.1:5000/Managers

George Mason University Employees Managers Projects Searches Result Visualize Department Strength Visualize Project Strength

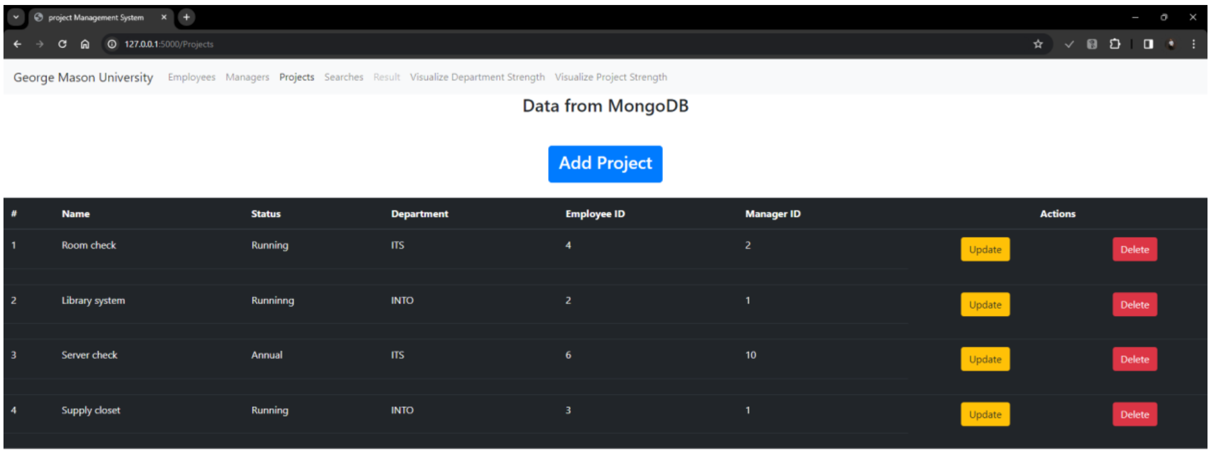
Managers from MongoDB Table

Add Manager

#	Name	Position	Department	Actions	
1	Kathy	Head	INTO	Update	Delete
2	Ben	FTA	ITS	Update	Delete
3	Vernon	Head	NuVegan	Update	Delete
4	Jinny	Professor	INTO	Update	Delete
6	Bev	Professor	INTO	Update	Delete
7	Zack	Head	ITS	Update	Delete
8	Sayf	HR	ITS	Update	Delete
9	ERIC	FTA	ITS	Update	Delete

Project:

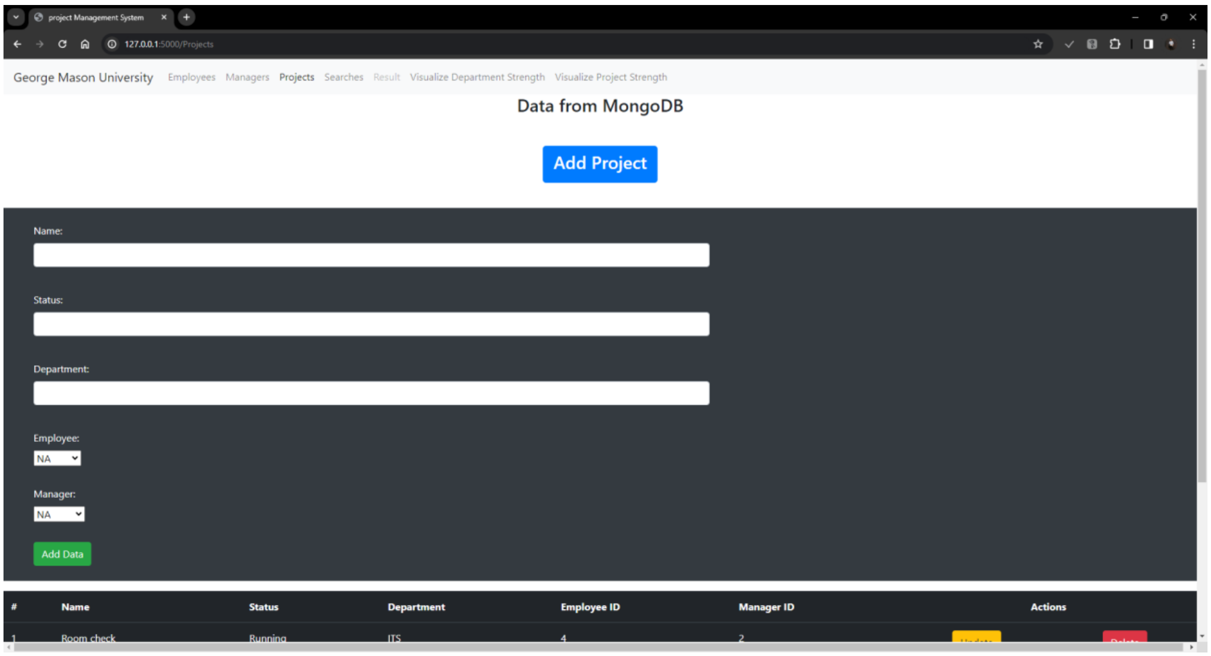
View Project:



Data from MongoDB

#	Name	Status	Department	Employee ID	Manager ID	Actions	
1	Room check	Running	ITS	4	2	Update	Delete
2	Library system	Runninng	INTO	2	1	Update	Delete
3	Server check	Annual	ITS	6	10	Update	Delete
4	Supply closet	Running	INTO	3	1	Update	Delete

Add Manager



Data from MongoDB

Name:

Status:

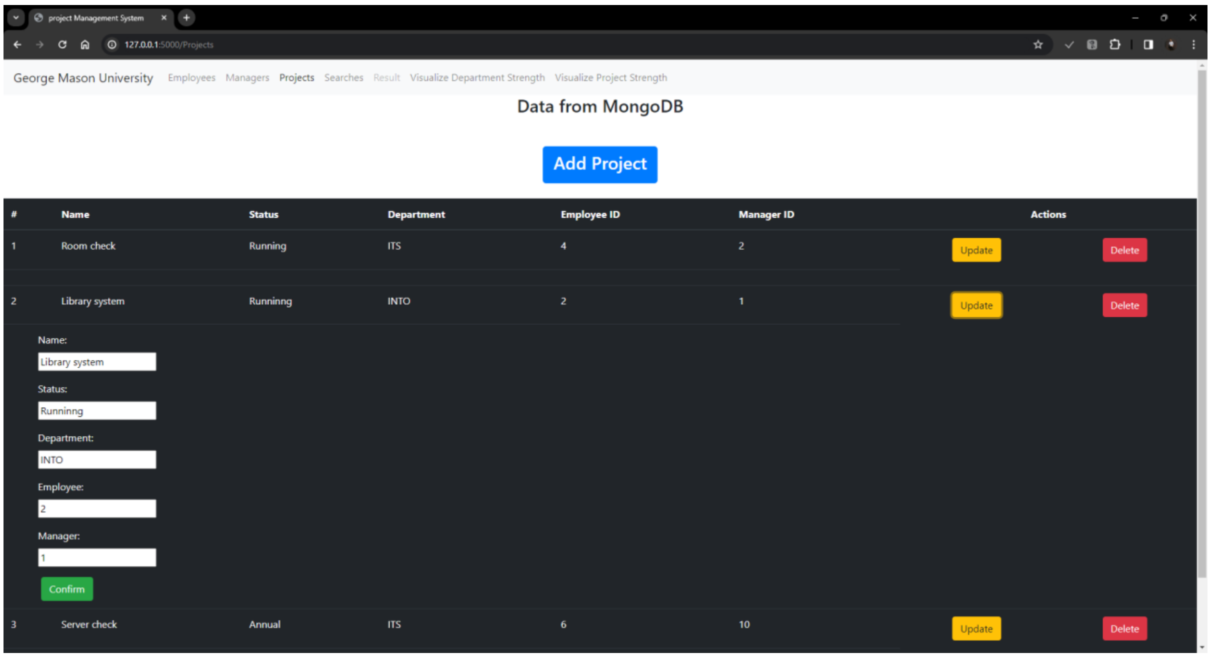
Department:

Employee: NA

Manager: NA

#	Name	Status	Department	Employee ID	Manager ID	Actions
1	Room check	Running	ITS	4	2	

Update Manager



Data from MongoDB

#	Name	Status	Department	Employee ID	Manager ID	Actions	
1	Room check	Running	ITS	4	2	Update	Delete
2	Library system	Runninng	INTO	2	1	Update	Delete
3	Server check	Annual	ITS	6	10	Update	Delete

Name: Library system

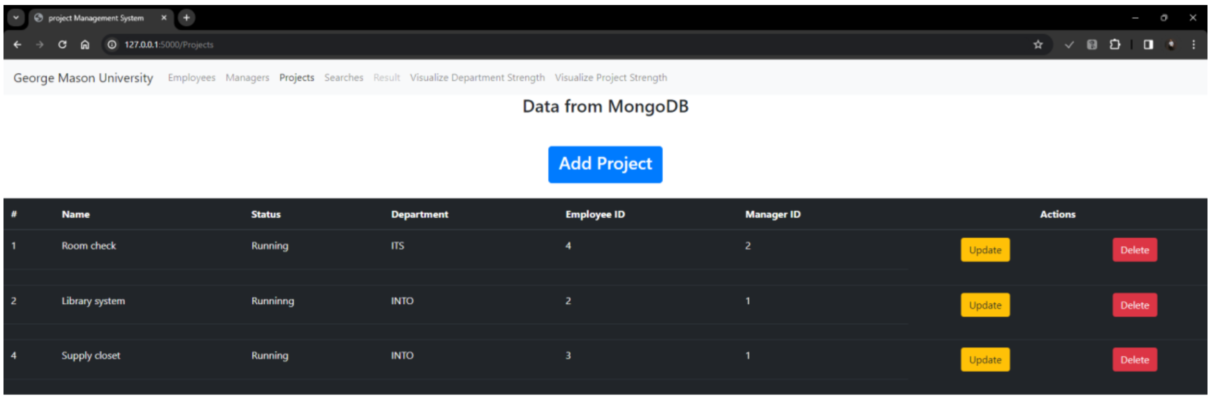
Status: Runninng

Department: INTO

Employee: 2

Manager: 1

Delete Manager



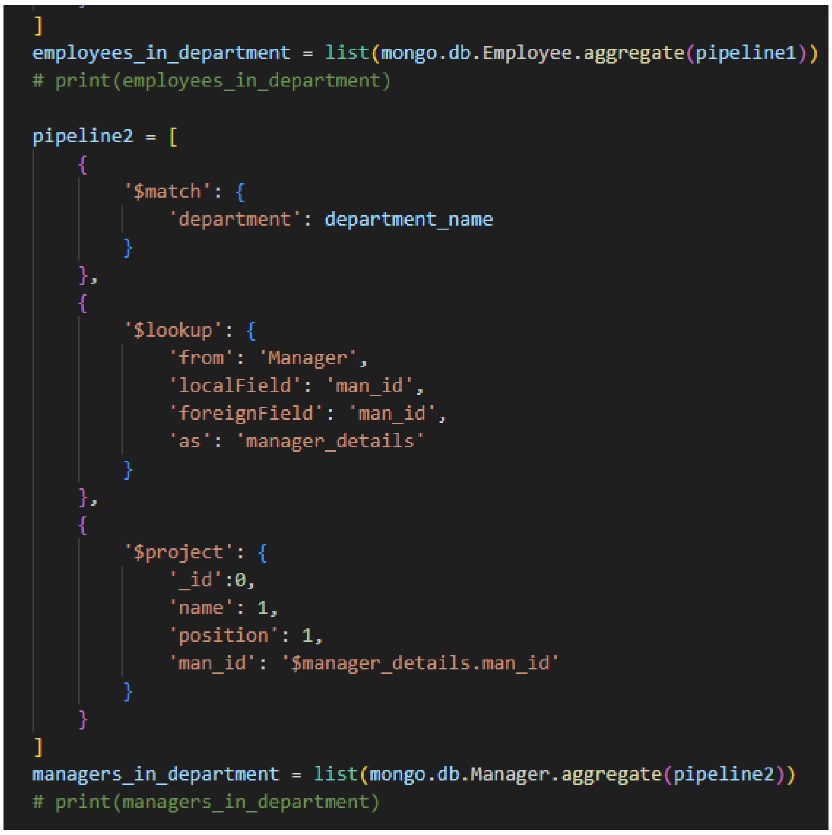
Data from MongoDB

#	Name	Status	Department	Employee ID	Manager ID	Actions	
1	Room check	Running	ITS	4	2	Update	Delete
2	Library system	Runninng	INTO	2	1	Update	Delete
4	Supply closet	Running	INTO	3	1	Update	Delete

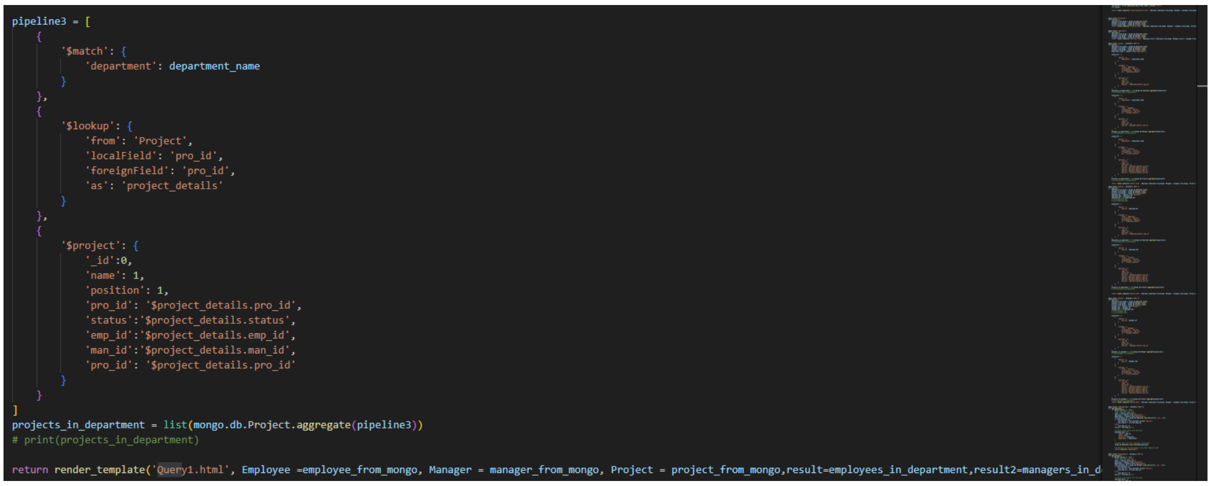
Complex Queries

Query 1

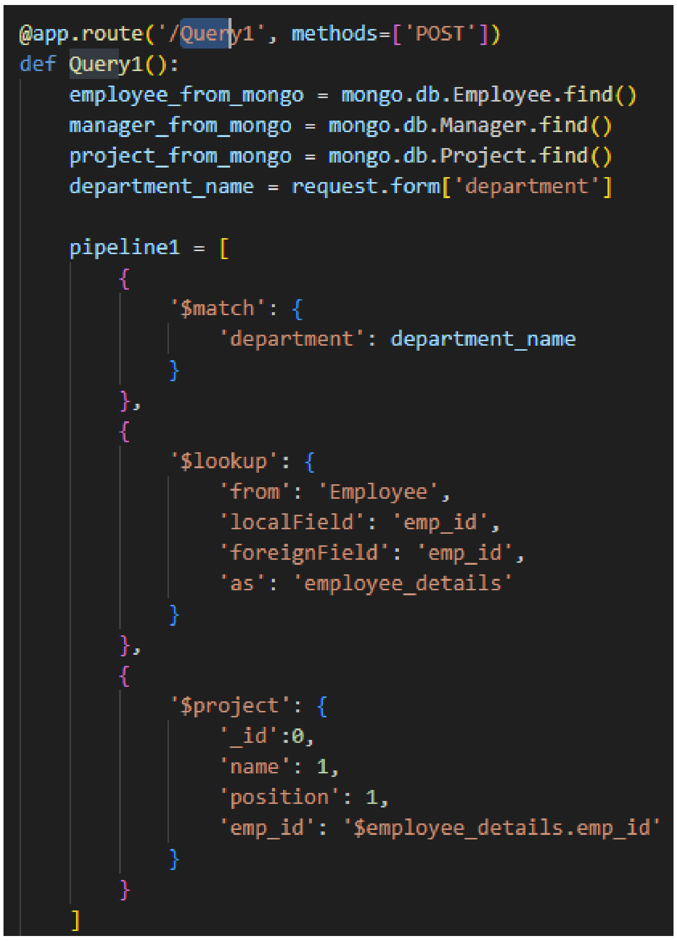
List all the Employees Managers and Projects by a particular department.



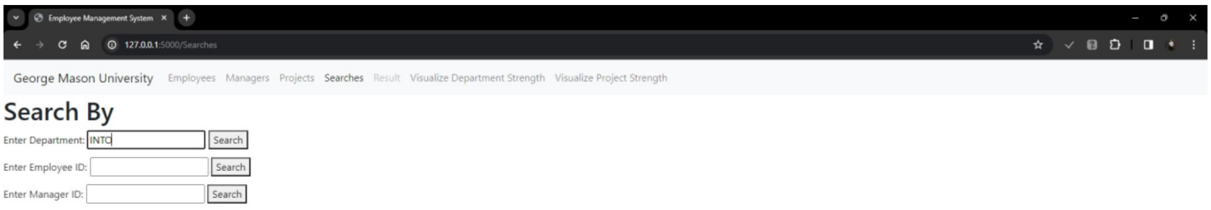
```
]
employees_in_department = list(mongo.db.Employee.aggregate(pipeline1))
# print(employees_in_department)

pipeline2 = [
    {
        '$match': {
            'department': department_name
        }
    },
    {
        '$lookup': {
            'from': 'Manager',
            'localField': 'man_id',
            'foreignField': 'man_id',
            'as': 'manager_details'
        }
    },
    {
        '$project': {
            '_id':0,
            'name': 1,
            'position': 1,
            'man_id': '$manager_details.man_id'
        }
    }
]
managers_in_department = list(mongo.db.Manager.aggregate(pipeline2))
# print(managers_in_department)
```



```
pipeline3 = [
    {
        '$match': {
            'department': department_name
        }
    },
    {
        '$lookup': {
            'from': 'Project',
            'localField': 'pro_id',
            'foreignField': 'pro_id',
            'as': 'project_details'
        }
    },
    {
        '$project': {
            '_id':0,
            'name': 1,
            'position': 1,
            'pro_id': '$project_details.pro_id',
            'status':'$project_details.status',
            'emp_id':'$project_details.emp_id',
            'man_id':'$project_details.man_id',
            'pro_id': '$project_details.pro_id'
        }
    }
]
projects_in_department = list(mongo.db.Project.aggregate(pipeline3))
# print(projects_in_department)

return render_template('Query1.html', Employee =employee_from_mongo, Manager = manager_from_mongo, Project = project_from_mongo,result=employees_in_department,result2=managers_in_d
```



```python
@app.route('/Query1', methods=['POST'])
def Query1():
    employee_from_mongo = mongo.db.Employee.find()
    manager_from_mongo = mongo.db.Manager.find()
    project_from_mongo = mongo.db.Project.find()
    department_name = request.form['department']

    pipeline1 = [
        {
            '$match': {
                'department': department_name
            }
        },
        {
            '$lookup': {
                'from': 'Employee',
                'localField': 'emp_id',
                'foreignField': 'emp_id',
                'as': 'employee_details'
            }
        },
        {
            '$project': {
                '_id':0,
                'name': 1,
                'position': 1,
                'emp_id': '$employee_details.emp_id'
            }
        }
    ]
```

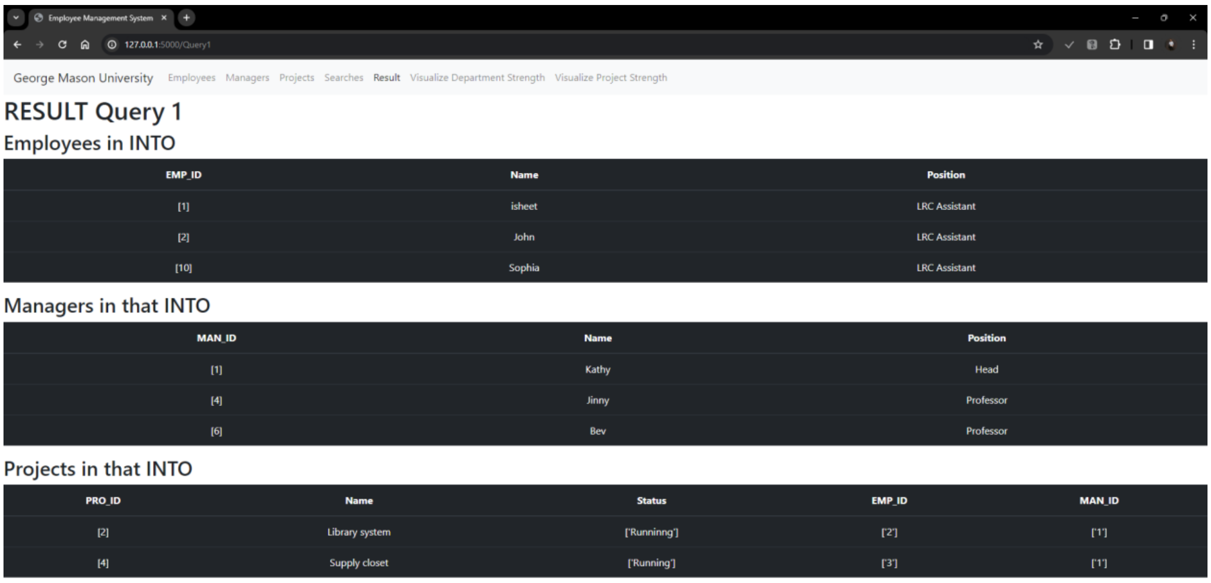


Search By

Enter Department: INTO Search

Enter Employee ID: Search

Enter Manager ID: Search



RESULT Query 1

Employees in INTO

EMP_ID	Name	Position
[1]	isheet	LRC Assistant
[2]	John	LRC Assistant
[10]	Sophia	LRC Assistant

Managers in that INTO

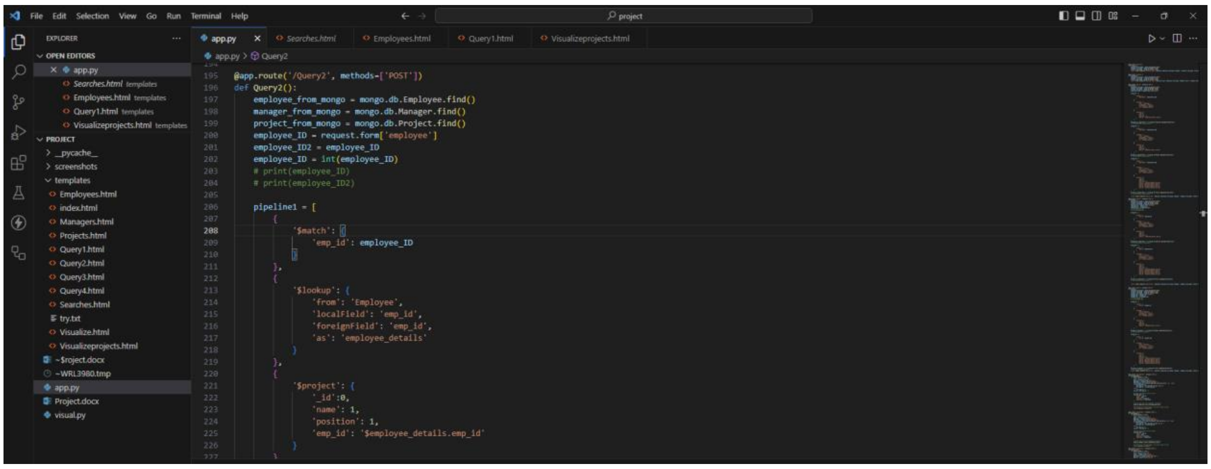
MAN_ID	Name	Position
[1]	Kathy	Head
[4]	Jinny	Professor
[6]	Bev	Professor

Projects in that INTO

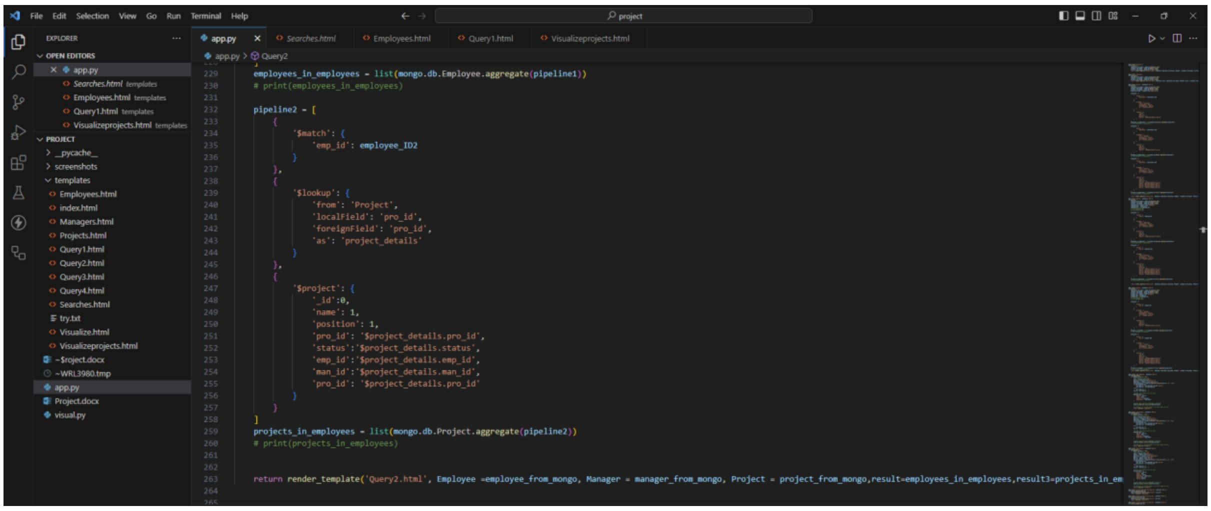
PRO_ID	Name	Status	EMP_ID	MAN_ID
[2]	Library system	['Runninng']	['2']	['1']
[4]	Supply closet	['Running']	['3']	['1']

Query 2

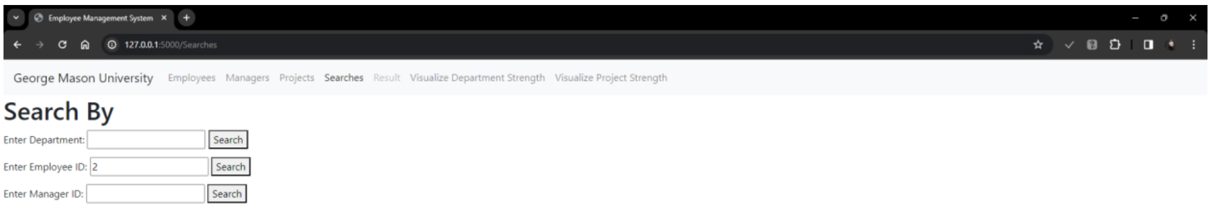
List all the Employees Details and Projects Details for employee with employee ID



```python
@app.route('/Query2', methods=['POST'])
def Query2():
    employee_from_mongo = mongo.db.Employee.find()
    manager_from_mongo = mongo.db.Manager.find()
    project_from_mongo = mongo.db.Project.find()
    employee_ID = request.form['employee']
    employee_ID2 = employee_ID
    employee_ID = int(employee_ID)
    # print(employee_ID)
    # print(employee_ID2)

    pipeline1 = [
        {
            '$match': {
                'emp_id': employee_ID
            }
        },
        {
            '$lookup': {
                'from': 'Employee',
                'localField': 'emp_id',
                'foreignField': 'emp_id',
                'as': 'employee_details'
            }
        },
        {
            '$project': {
                '_id':0,
                'name': 1,
                'position': 1,
                'emp_id': '$employee_details.emp_id'
            }
```



```python
    employees_in_employees = list(mongo.db.Employee.aggregate(pipeline1))
    # print(employees_in_employees)

    pipeline2 = [
        {
            '$match': {
                'emp_id': employee_ID2
            }
        },
        {
            '$lookup': {
                'from': 'Project',
                'localField': 'pro_id',
                'foreignField': 'pro_id',
                'as': 'project_details'
            }
        },
        {
            '$project': {
                '_id':0,
                'name': 1,
                'position': 1,
                'pro_id': '$project_details.pro_id',
                'status':'$project_details.status',
                'emp_id':'$project_details.emp_id',
                'man_id':'$project_details.man_id',
                'pro_id': '$project_details.pro_id'
            }
        }
    ]
    projects_in_employees = list(mongo.db.Project.aggregate(pipeline2))
    # print(projects_in_employees)


    return render_template('Query2.html', Employee =employee_from_mongo, Manager = manager_from_mongo, Project = project_from_mongo,result=employees_in_employees,result3=projects_in_em
```

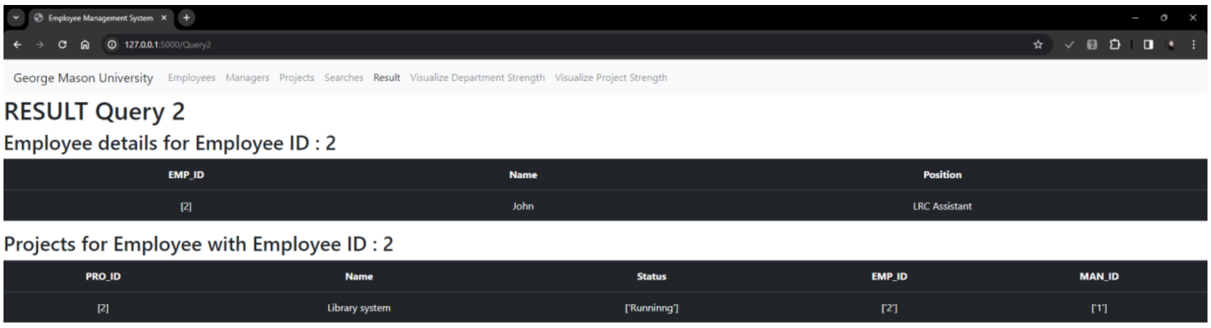


Search By

Enter Department: Search

Enter Employee ID: 2 Search

Enter Manager ID: Search



RESULT Query 2

Employee details for Employee ID : 2

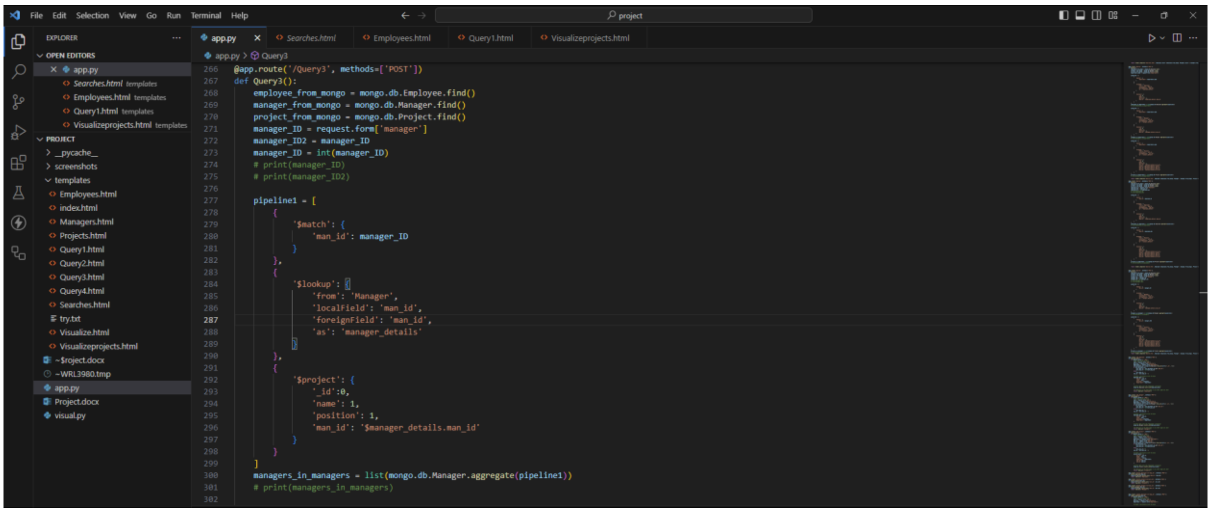
EMP_ID	Name	Position
[2]	John	LRC Assistant

Projects for Employee with Employee ID : 2

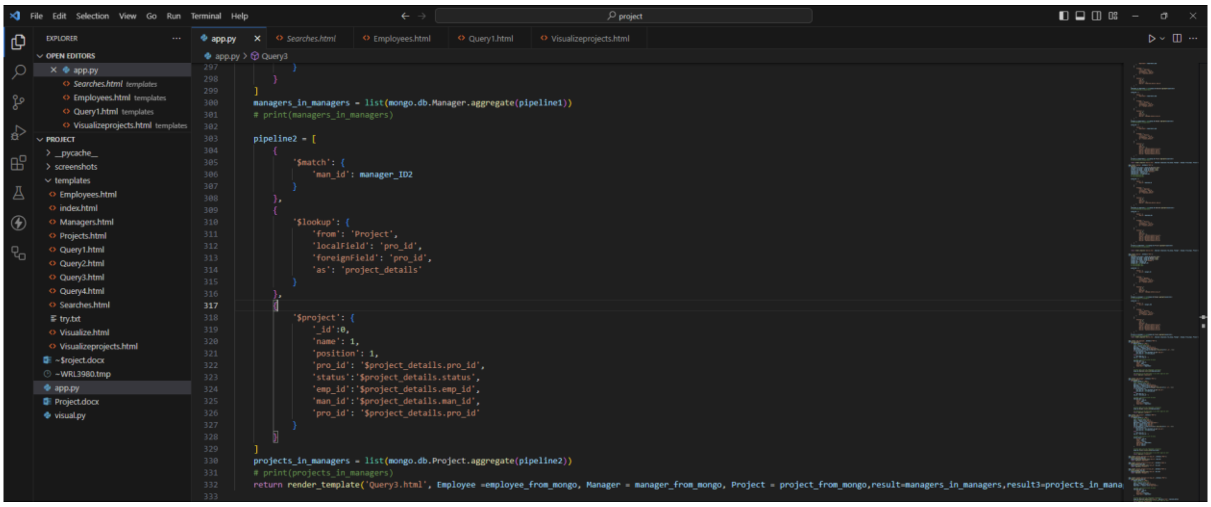
PRO_ID	Name	Status	EMP_ID	MAN_ID
[2]	Library system	['Runninng']	['2']	['1']

Query 3

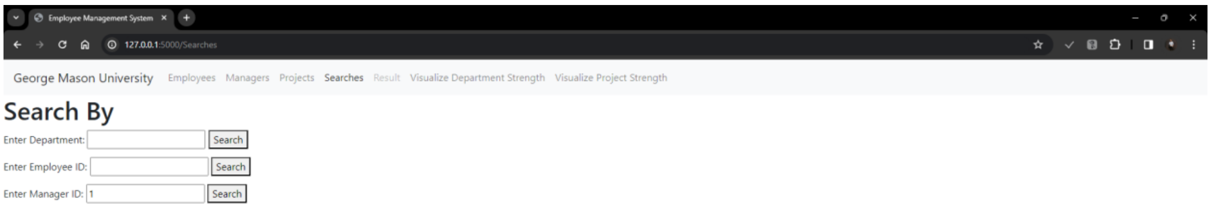
List all the Manager Details and Projects Details for manager with manager ID



```python
@app.route('/Query3', methods=['POST'])
def Query3():
    employee_from_mongo = mongo.db.Employee.find()
    manager_from_mongo = mongo.db.Manager.find()
    project_from_mongo = mongo.db.Project.find()
    manager_ID = request.form['manager']
    manager_ID2 = manager_ID
    manager_ID = int(manager_ID)
    # print(manager_ID)
    # print(manager_ID2)

    pipeline1 = [
        {
            '$match': {
                'man_id': manager_ID
            }
        },
        {
            '$lookup': {
                'from': 'Manager',
                'localField': 'man_id',
                'foreignField': 'man_id',
                'as': 'manager_details'
            }
        },
        {
            '$project': {
                '_id':0,
                'name': 1,
                'position': 1,
                'man_id': '$manager_details.man_id'
            }
        }
    ]
    managers_in_managers = list(mongo.db.Manager.aggregate(pipeline1))
    # print(managers_in_managers)

```

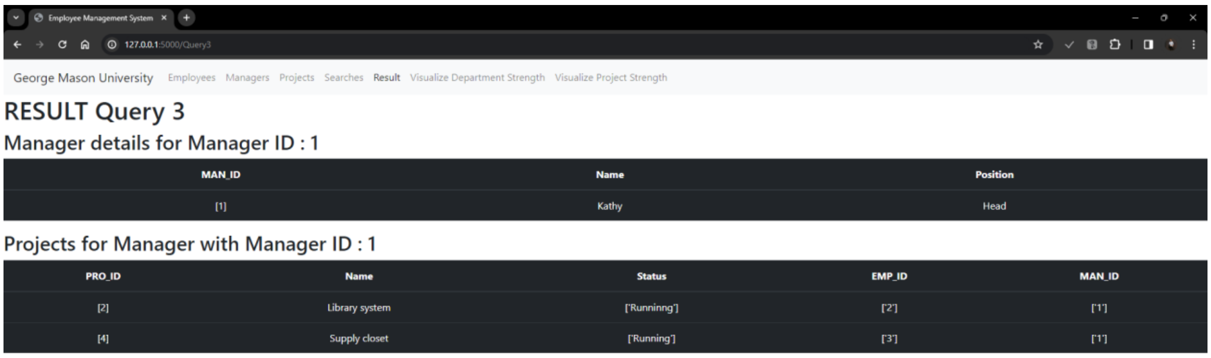


```python
            }
        }
    ]
    managers_in_managers = list(mongo.db.Manager.aggregate(pipeline1))
    # print(managers_in_managers)

    pipeline2 = [
        {
            '$match': {
                'man_id': manager_ID2
            }
        },
        {
            '$lookup': {
                'from': 'Project',
                'localField': 'pro_id',
                'foreignField': 'pro_id',
                'as': 'project_details'
            }
        },
        {
            '$project': {
                '_id':0,
                'name': 1,
                'position': 1,
                'pro_id': '$project_details.pro_id',
                'status':'$project_details.status',
                'emp_id':'$project_details.emp_id',
                'man_id':'$project_details.man_id',
                'pro_id': '$project_details.pro_id'
            }
        }
    ]
    projects_in_managers = list(mongo.db.Project.aggregate(pipeline2))
    # print(projects_in_managers)
    return render_template('Query3.html', Employee =employee_from_mongo, Manager = manager_from_mongo, Project = project_from_mongo,result=managers_in_managers,result3=projects_in_mana

```



Search By

Enter Department:		Search
Enter Employee ID:		Search
Enter Manager ID:	1	Search



RESULT Query 3

Manager details for Manager ID : 1

MAN_ID	Name	Position
[1]	Kathy	Head

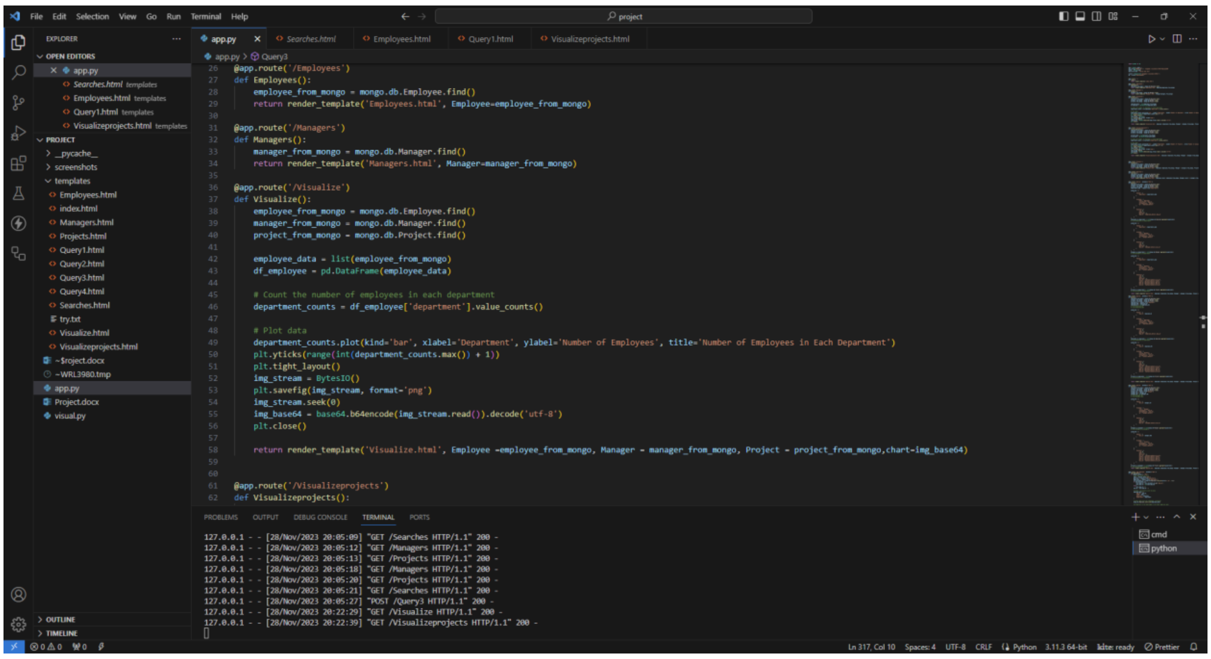
Projects for Manager with Manager ID : 1

PRO_ID	Name	Status	EMP_ID	MAN_ID
[2]	Library system	['Runninng']	['2']	['1']
[4]	Supply closet	['Running']	['3']	['1']

Visualization

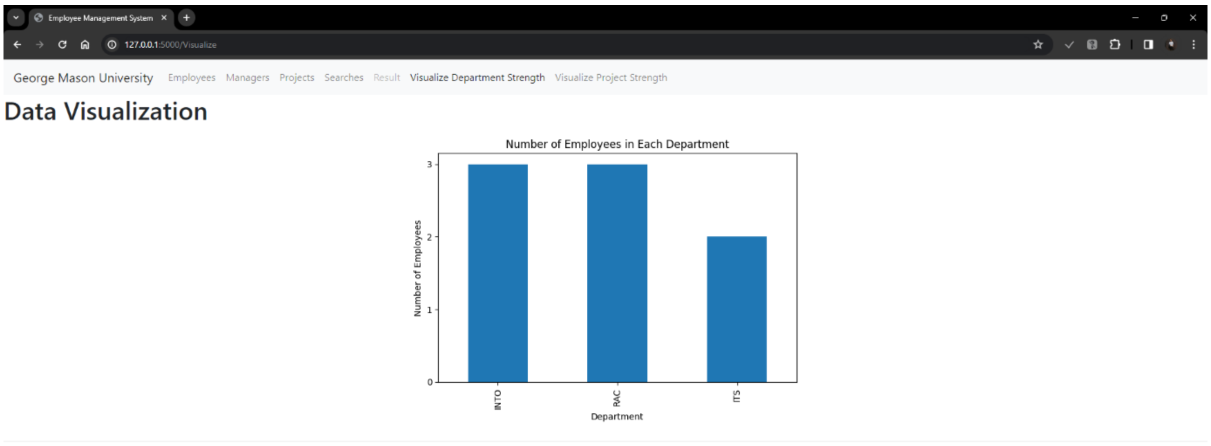
Visualization 1

Visualize the graph for the number of employees vs the department showing us exactly how many employees are present in each department.



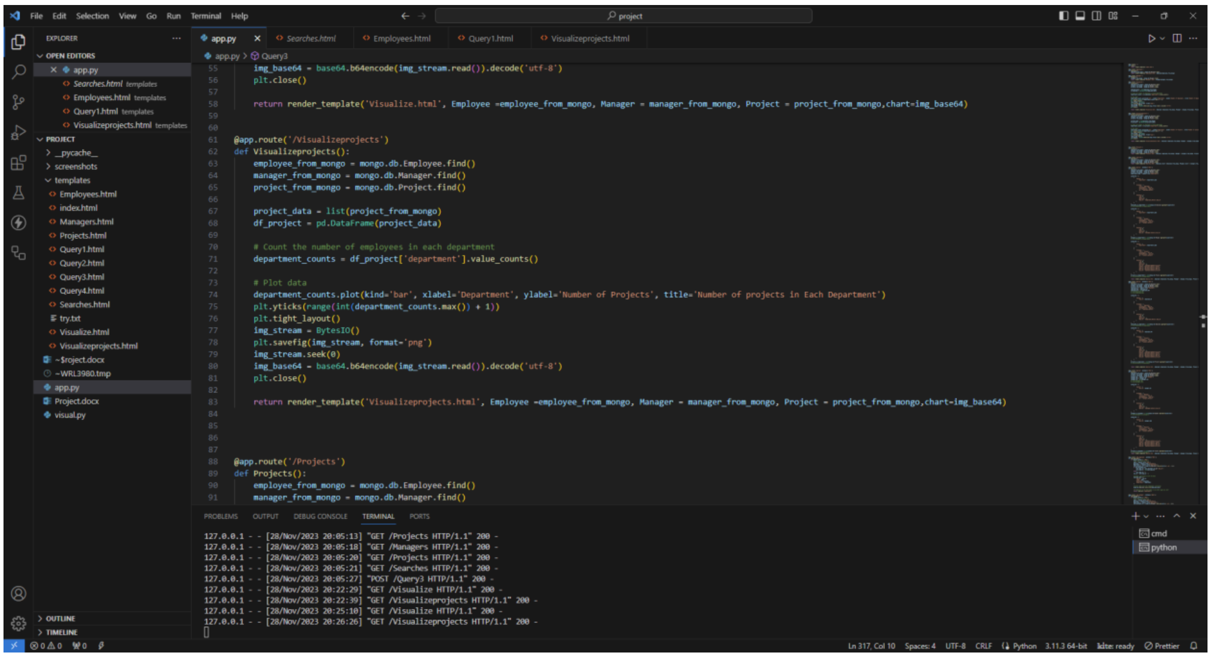
```python
@app.route('/Employees')
def Employees():
    employee_from_mongo = mongo.db.Employee.find()
    return render_template('Employees.html', Employee=employee_from_mongo)

@app.route('/Managers')
def Managers():
    manager_from_mongo = mongo.db.Manager.find()
    return render_template('Managers.html', Manager=manager_from_mongo)

@app.route('/Visualize')
def Visualize():
    employee_from_mongo = mongo.db.Employee.find()
    manager_from_mongo = mongo.db.Manager.find()
    project_from_mongo = mongo.db.Project.find()

    employee_data = list(employee_from_mongo)
    df_employee = pd.DataFrame(employee_data)

    # Count the number of employees in each department
    department_counts = df_employee['department'].value_counts()

    # Plot data
    department_counts.plot(kind='bar', xlabel='Department', ylabel='Number of Employees', title='Number of Employees in Each Department')
    plt.yticks(range(int(department_counts.max()) + 1))
    plt.tight_layout()
    img_stream = BytesIO()
    plt.savefig(img_stream, format='png')
    img_stream.seek(0)
    img_base64 = base64.b64encode(img_stream.read()).decode('utf-8')
    plt.close()

    return render_template('Visualize.html', Employee =employee_from_mongo, Manager = manager_from_mongo, Project = project_from_mongo,chart=img_base64)


@app.route('/Visualizeprojects')
def Visualizeprojects():
```

```
127.0.0.1 - - [28/Nov/2023 20:05:09] "GET /Searches HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:12] "GET /Managers HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:13] "GET /Projects HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:18] "GET /Managers HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:20] "GET /Projects HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:21] "GET /Searches HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:27] "POST /Query3 HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:22:29] "GET /Visualize HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:22:39] "GET /Visualizeprojects HTTP/1.1" 200 -
```



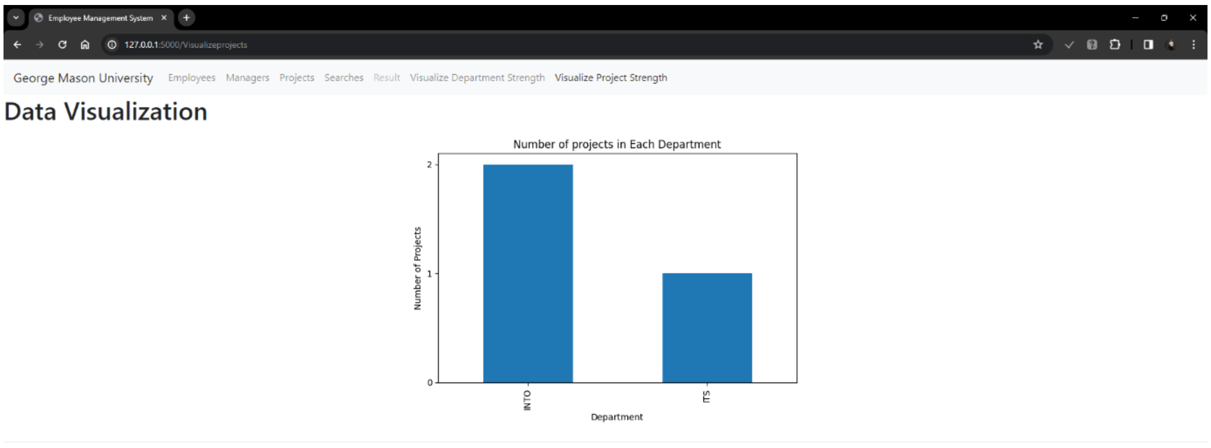
Visualization 2

Visualize the graph for the number of projects vs the department showing us exactly how many projects are present in each department.



```python
    img_base64 = base64.b64encode(img_stream.read()).decode('utf-8')
    plt.close()

    return render_template('Visualize.html', Employee =employee_from_mongo, Manager = manager_from_mongo, Project = project_from_mongo,chart=img_base64)


@app.route('/Visualizeprojects')
def Visualizeprojects():
    employee_from_mongo = mongo.db.Employee.find()
    manager_from_mongo = mongo.db.Manager.find()
    project_from_mongo = mongo.db.Project.find()

    project_data = list(project_from_mongo)
    df_project = pd.DataFrame(project_data)

    # Count the number of employees in each department
    department_counts = df_project['department'].value_counts()

    # Plot data
    department_counts.plot(kind='bar', xlabel='Department', ylabel='Number of Projects', title='Number of projects in Each Department')
    plt.yticks(range(int(department_counts.max()) + 1))
    plt.tight_layout()
    img_stream = BytesIO()
    plt.savefig(img_stream, format='png')
    img_stream.seek(0)
    img_base64 = base64.b64encode(img_stream.read()).decode('utf-8')
    plt.close()

    return render_template('Visualizeprojects.html', Employee =employee_from_mongo, Manager = manager_from_mongo, Project = project_from_mongo,chart=img_base64)




@app.route('/Projects')
def Projects():
    employee_from_mongo = mongo.db.Employee.find()
    manager_from_mongo = mongo.db.Manager.find()
```

```
127.0.0.1 - - [28/Nov/2023 20:05:13] "GET /Projects HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:18] "GET /Managers HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:20] "GET /Projects HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:21] "GET /Searches HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:05:27] "POST /Query3 HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:22:29] "GET /Visualize HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:22:39] "GET /Visualizeprojects HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:25:10] "GET /Visualize HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2023 20:26:26] "GET /Visualizeprojects HTTP/1.1" 200 -
```



Technologies Used

Frontend:

- HTML (Hypertext Markup Language)
- CSS (Cascading Style Sheets)
- JavaScript
- Bootstrap

Backend:

- Python
- Flask

Database:

- MongoDB
- MongoDB Compass

Accomplishments

User Interface Development:

Basic CRUD Operations:

Designed and developed user interfaces for adding, deleting, and updating records for three collections: Employee, Manager and Project

Integrated these UI components with backend APIs to interact with the NoSQL database.

Search Queries UI:

Implemented a user interface for executing complex search queries (query1, query2 and query3) involving data from all three collections.

Provided user-friendly input forms to pass parameters for executing the search queries.

Backend Integration:

Integrated UI components with backend APIs to seamlessly perform add, delete, update, and search operations on MongoDB collections (employee, manager, and project).

MongoDB Queries:

Query1, Query2 and Query3 MongoDB Queries:

Wrote MongoDB queries for query1, query2 and query3 to efficiently retrieve and filter data from the three collections based on the specified parameters.

Results Display:

Displaying Query Results:

Integrated UI components with backend APIs to retrieve and display the results of query1, query2 and query3 in the user interface.

Dashboard Implementation:

Dashboard Design:

Designed and implemented a dashboard that provides a graphical representation of relevant data from the three collections (employee, manager, and project)

Chart Visualization:

Incorporated two sections to display charts that visualize data trends or insights from the NoSQL database.

General Requirements:

Three Collections:

Ensured the project includes at least three collections in the NoSQL database, namely Employee, Manager, and Project.

Basic View, Insert, Update, and Delete:

Implemented basic views for each collection, allowing users to insert, update, and delete records through the user interface.

Search Queries Involving All Collections:

Developed three search queries involving data from all three collections, providing a comprehensive search functionality for users.

Chart Visualization Sections:

Included two sections in the UI to visually represent data through charts, enhancing the user experience and providing insights into the database information.

NoSQL Database Usage:

Utilized a NoSQL database (MongoDB) for storing and managing the data efficiently.

By accomplishing these tasks, the project ensures a robust user interface, seamless interaction with the backend, effective search functionalities, and a visually appealing dashboard for data representation.