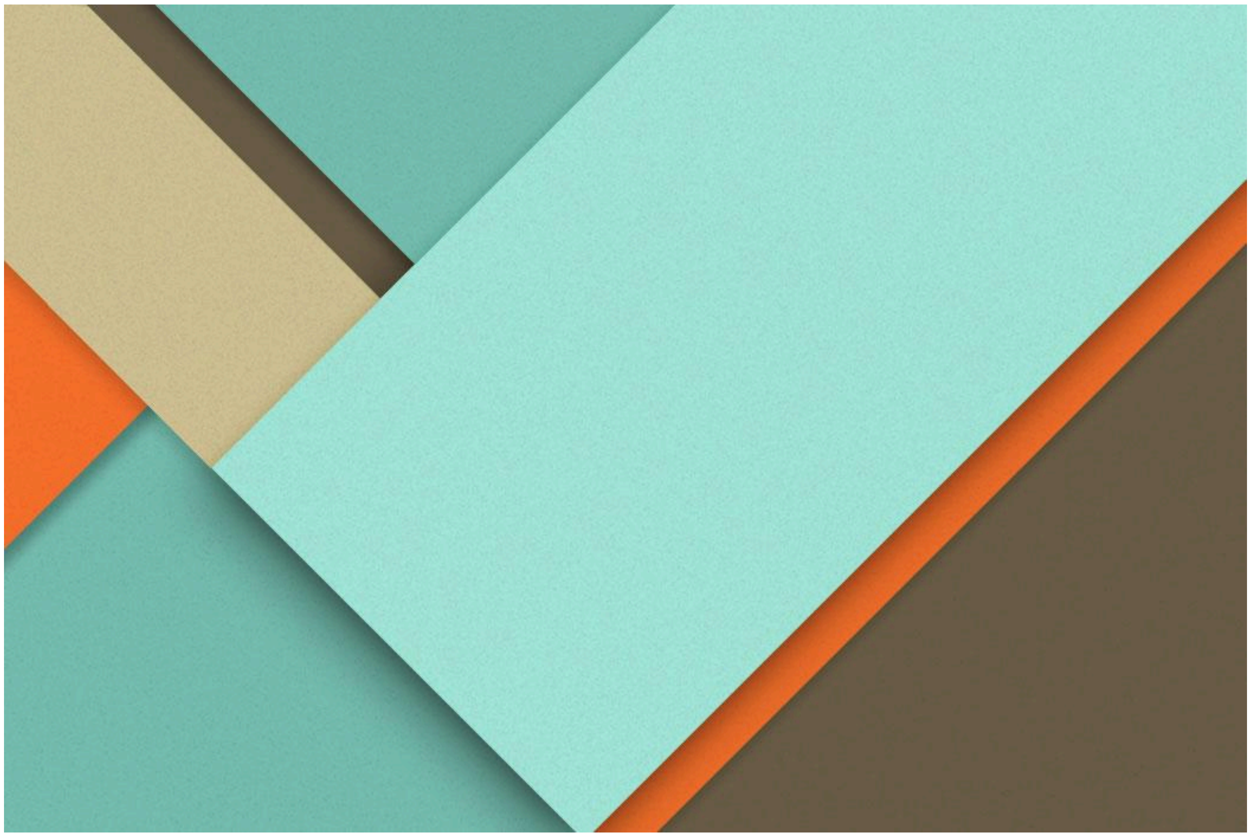


EEMS

10.21.2025

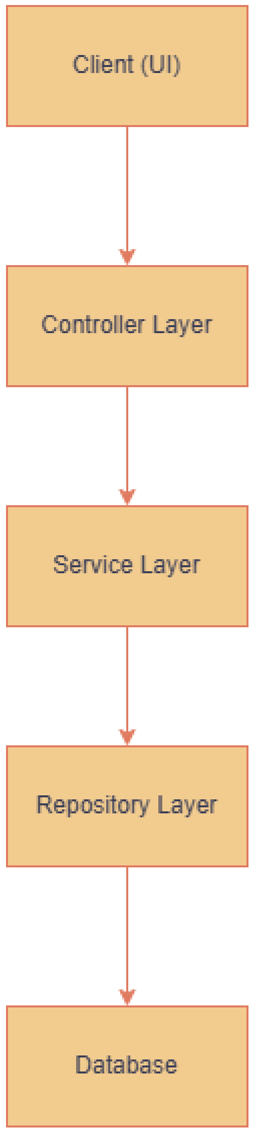
John Okon Ansa
61955
Group 11
jansa@miu.edu

Overview

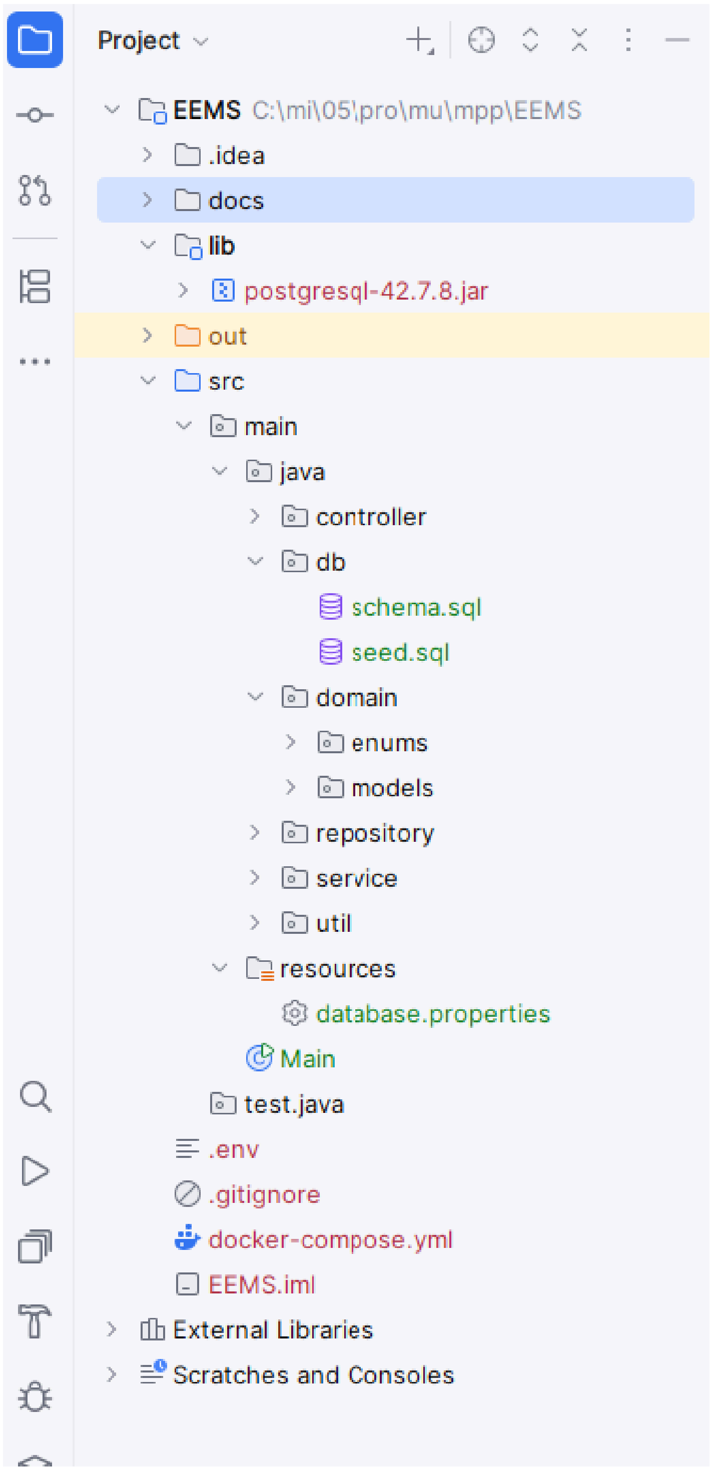
This project is an implementation of a non-framework-dependent N-Tier Java application using pure JDBC for data persistence.

Project Structure

Architectural Diagram



Folder Structure



```
EEMS C:\mi\05\pro\mu\mpp\EEMS
  .idea
  docs
  lib
    postgresql-42.7.8.jar
  out
  src
    main
      java
        controller
        db
          schema.sql
          seed.sql
        domain
          enums
          models
        repository
        service
        util
      resources
        database.properties
      Main
    test.java
  .env
  .gitignore
  docker-compose.yml
  EEMS.iml
External Libraries
Scratches and Consoles
```

Required Deliverables

I. Domain Class Definition

A list of all necessary classes, including attributes, data types, and brief justifications for each class.

Classes

1. **Department**
    a. **Purpose**: Represents an organizational unit responsible for hosting and managing projects to which employees belong.
    b. **Attributes**:
        i. *id (int)*: Unique identifier for the department (Primary Key).
        ii. *name (String)*: Name of the department
        iii. *location (String)*: Location of the department
        iv. *annualBudget (double)*: Annual budget of the department
    c. **Justification**: Needed to manage organizational structure, budgeting, and grouping of employees/projects.
2. **Employee**
    a. **Purpose**: Represents a company staff member (the workforce).
    b. **Attributes**:
        i. *id (int)*: Unique identifier for the employee (Primary Key).
        ii. *fullName (String)*: Full name of the employee
        iii. *title (String)*: Job title of the employee
        iv. *hireDate (Date)*: Hire date of the employee
        v. *salary (double)*: Salary of the employee
        vi. *departmentId (int)*: Foreign key referencing the Department class
    c. **Justification**: Core entity to represent human resources and link to projects for HR cost calculations and transfers.
3. **Project**
    a. **Purpose**: Represents an operational task or initiative within the company.
    b. **Attributes**:
        i. *id (int)*: Unique identifier for the project (Primary Key).
        ii. *name (String)*: Name of the project
        iii. *description (String)*: Description of the project
        iv. *startDate (Date)*: Start date of the project
        v. *endDate (Date)*: End date of the project

vi. *budget (double)*: Budget of the project
vii. *status (enum): Status of the project (e.g., Active, Completed)*

c. **Justification**: Central to company operations and used in all major business logic tasks (cost calculation, reporting, client associations).

4. **Client**
    a. **Purpose**: Represents external organizations that partner with or sponsor projects.
    b. **Attributes**:
        i. *id (int)*: Unique identifier for the client (Primary Key)
        ii. *name (String)*: Name of the client
        iii. *industry (String)*: Industry of the client
        iv. *primaryContactName (String)*: Name of the primary contact person
        v. *primaryContactPhone (String)*: Phone number of the primary contact person
        vi. *primaryContactEmail (String)*: Email of the primary contact person
    c. **Justification**: Supports tracking of client–project relationships, essential for business development and reporting (e.g., finding clients by project deadlines).
5. **EmployeeProject (Junction Table)**
    a. **Purpose**: Tracks the many-to-many relationship between employees and projects and includes additional information about work allocation.
    b. **Attributes**:
        i. *employeeId (int)*: Foreign key referencing the Employee class
        ii. *projectId (int)*: Foreign key referencing the Project class
        iii. *percentageTimeAllocation(double)*: Percentage of time allocated to the project
    c. **Justification**: Allows tracking how employees contribute to projects and forms the basis for the calculateProjectHRCost(int projectId) method.
6. **ClientProject (Junction Table)**
    a. **Purpose**: Tracks many-to-many relationships between projects and clients.
    b. **Attributes**:
        i. *clientId (int)*: Foreign key referencing the Client class
        ii. *projectId (int)*: Foreign key referencing the Project class
    c. **Justification**: Allows linking multiple clients to one project and vice versa for business reports and client engagement tracking.
7. **DepartmentProject (Junction Table)**

   a. **Purpose**: Tracks many-to-many relationships between projects and clients.
   b. **Attributes**:
      i. *departmentId (int)*: Foreign key referencing the Department class
      ii. *projectId (int)*: Foreign key referencing the Project class
   c. **Justification**: Allows linking multiple clients to one project and vice versa for business reports and client engagement tracking.

8.

Datatype

- int: For IDs (primary/foreign keys)
- String: For names, descriptions, contact info
- double: For monetary values (salary, budget)
- LocalDate: For dates (hire date, project dates)
- enum: Could be used for status fields

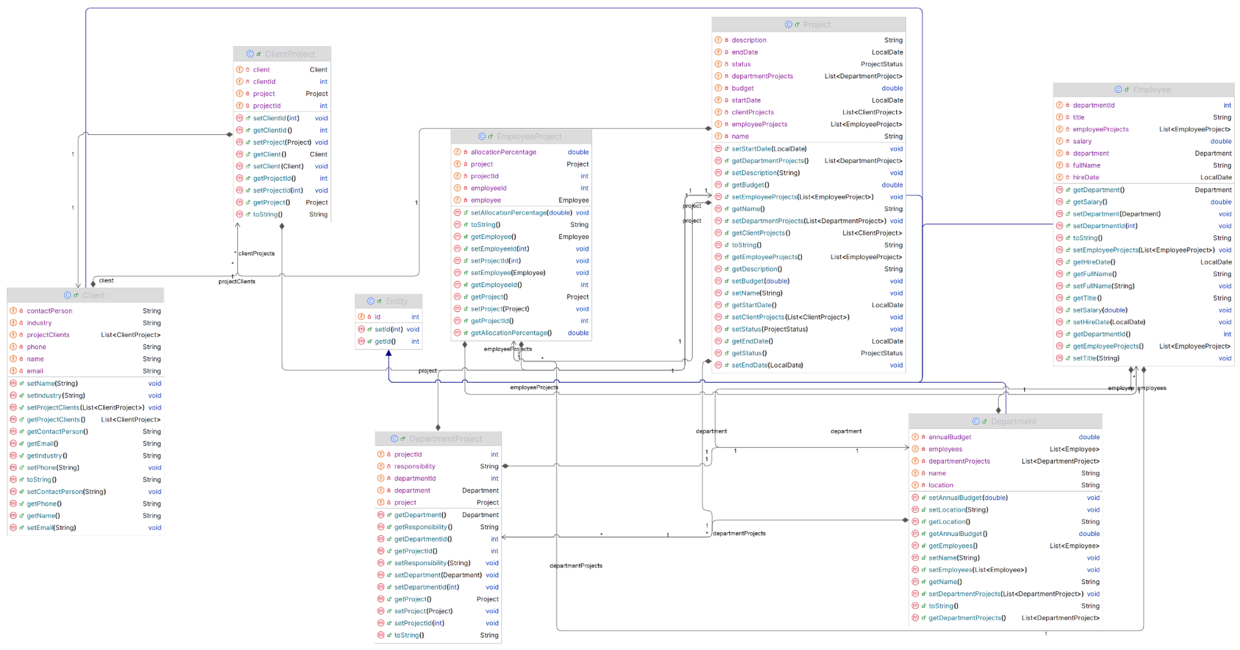
Relationships

The multiplicities can be represented as follows:

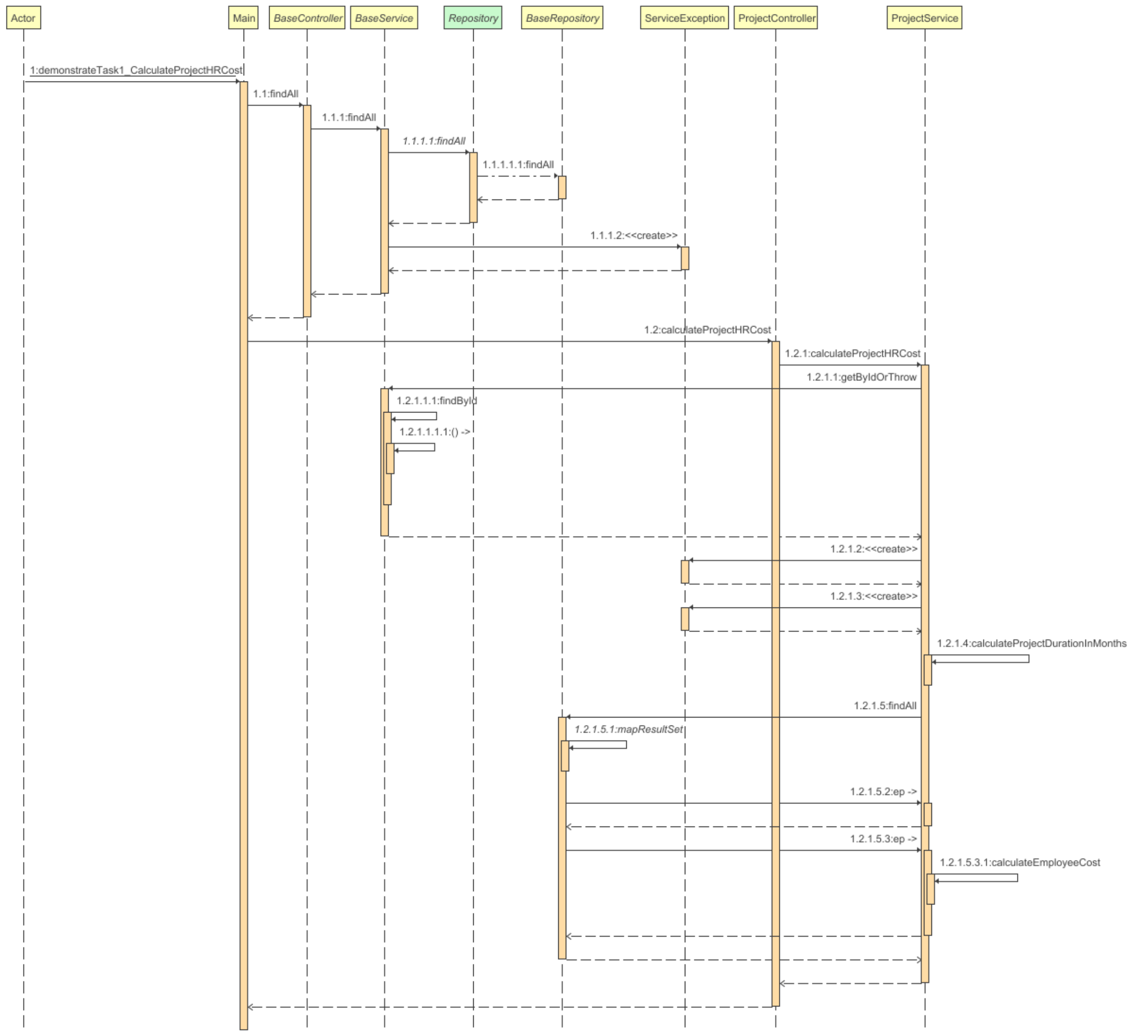
- Department 1:N Employee
- Employee N:M Project (via Employee_Project junction table)
- Department N:M Project (via Department_Project junction table)
- Project N:M Client (via Client_Project junction table)

- **Department - Employee:**
   a. A department can have multiple employees (1:N).
   b. An employee belongs to one department (N:1).
- **Employee - Project:**
   a. An employee can be assigned to multiple projects (N:M).
   b. A project can have multiple employees (N:M).
- **Department - Project:**
   a. A department can manage or host multiple projects (N:M).
   b. A project can be managed or hosted by multiple departments (N:M).
- **Project - Client:**
   a. A project can be tied to multiple clients (N:M).
   b. A client can sponsor or be associated with multiple projects (N:M).

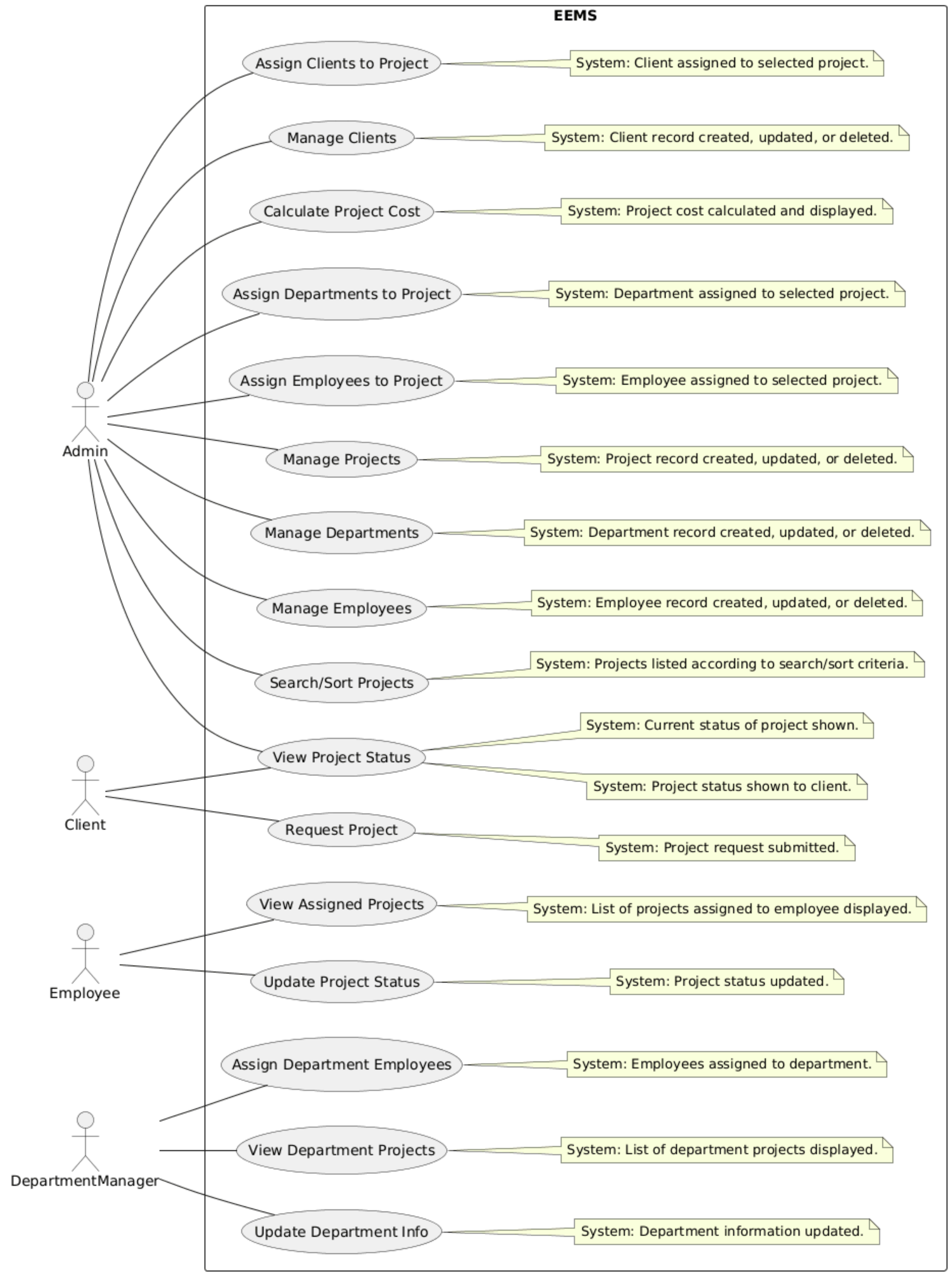
II. Class Diagram



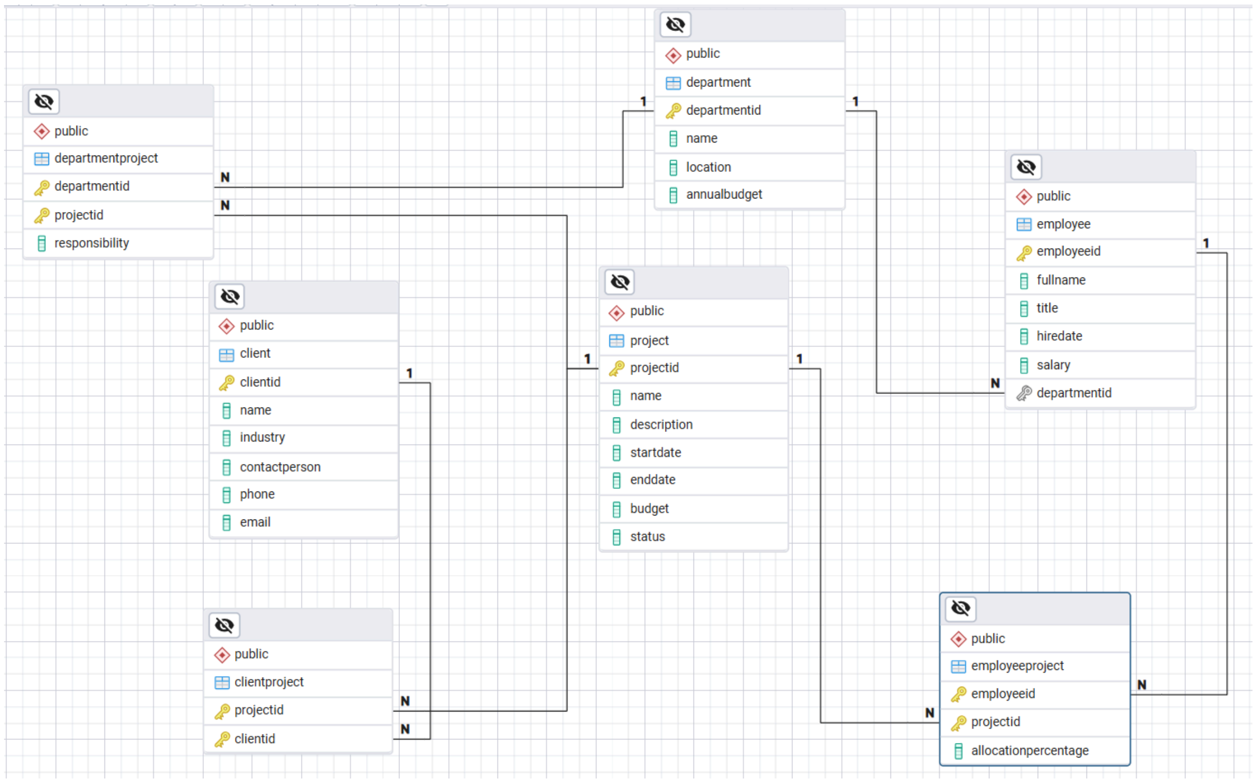
III. Sequence Diagram



IV. Use Case Diagram


EEMS
Assign Clients to Project
System: Client assigned to selected project.
Manage Clients
System: Client record created, updated, or deleted.
Calculate Project Cost
System: Project cost calculated and displayed.
Assign Departments to Project
System: Department assigned to selected project.
Assign Employees to Project
System: Employee assigned to selected project.
Manage Projects
System: Project record created, updated, or deleted.
Manage Departments
System: Department record created, updated, or deleted.
Manage Employees
System: Employee record created, updated, or deleted.
Search/Sort Projects
System: Projects listed according to search/sort criteria.
View Project Status
System: Current status of project shown.
System: Project status shown to client.
Request Project
System: Project request submitted.
View Assigned Projects
System: List of projects assigned to employee displayed.
Update Project Status
System: Project status updated.
Assign Department Employees
System: Employees assigned to department.
View Department Projects
System: List of department projects displayed.
Update Department Info
System: Department information updated.
Admin
Client
Employee
DepartmentManager

V. Database Schema Diagram


public
department
departmentid
name
location
annualbudget
public
departmentproject
departmentid
projectid
responsibility
public
employee
employeeid
fullname
title
hiredate
salary
departmentid
public
client
clientid
name
industry
contactperson
phone
email
public
project
projectid
name
description
startdate
enddate
budget
status
public
clientproject
projectid
clientid
public
employeeproject
employeeid
projectid
allocationpercentage
1
N