PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Object Oriented Analysis and Design using Java (UE20CS352)

Mini Project

TOPIC:

DISASTER RELIEF MANAGEMENT SYSTEM

TEAM MEMBER DETAILS:

NAME	SRN
Abu Bucker Siddique	PES1UG20CS617
Manojkumar Darshankar	PES1UG20CS662
Adarsh K S	PES1UG20CS619
P Chaitanya P S	PES1UG20CS669

PROBLEM STATEMENT:

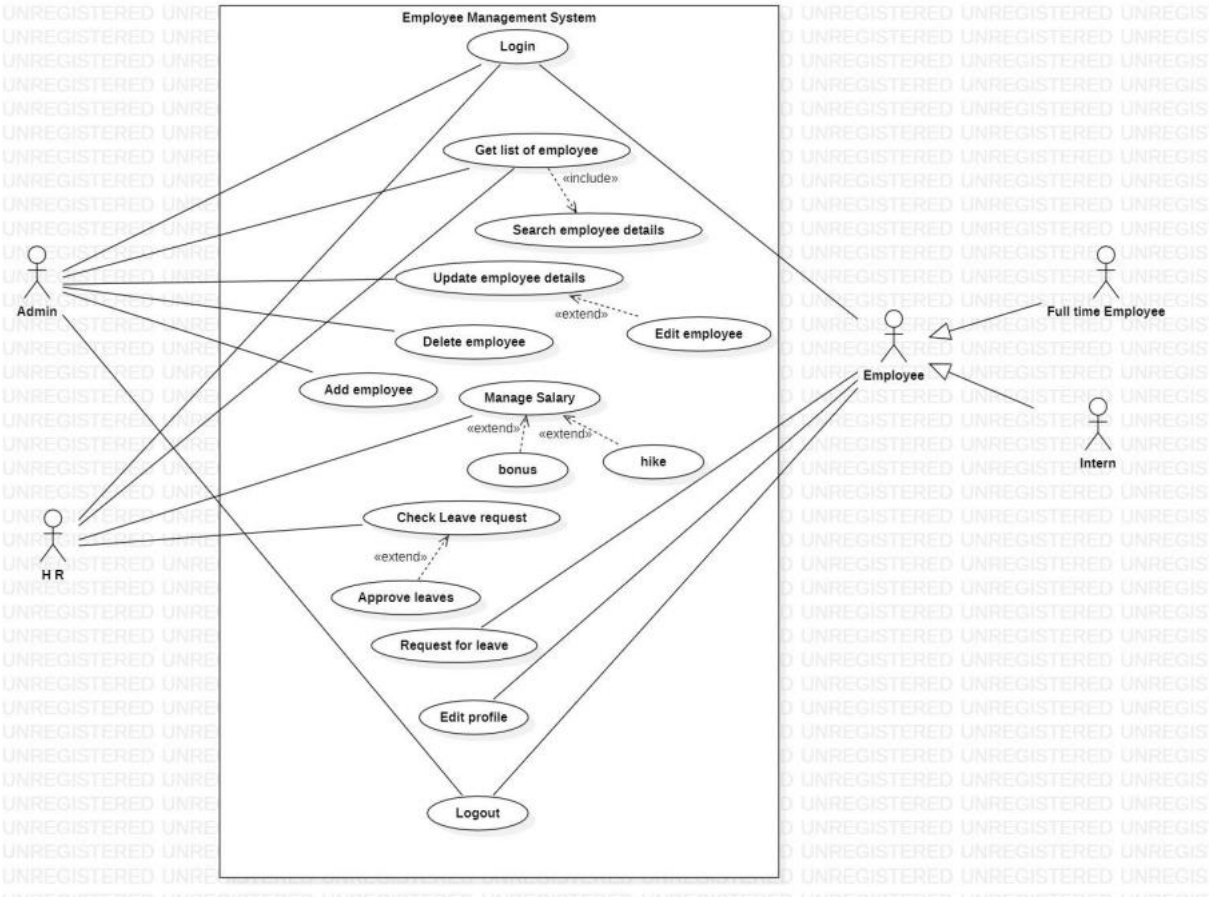
Employee-Management-System

LINK TO GITHUB REPOSITORY:

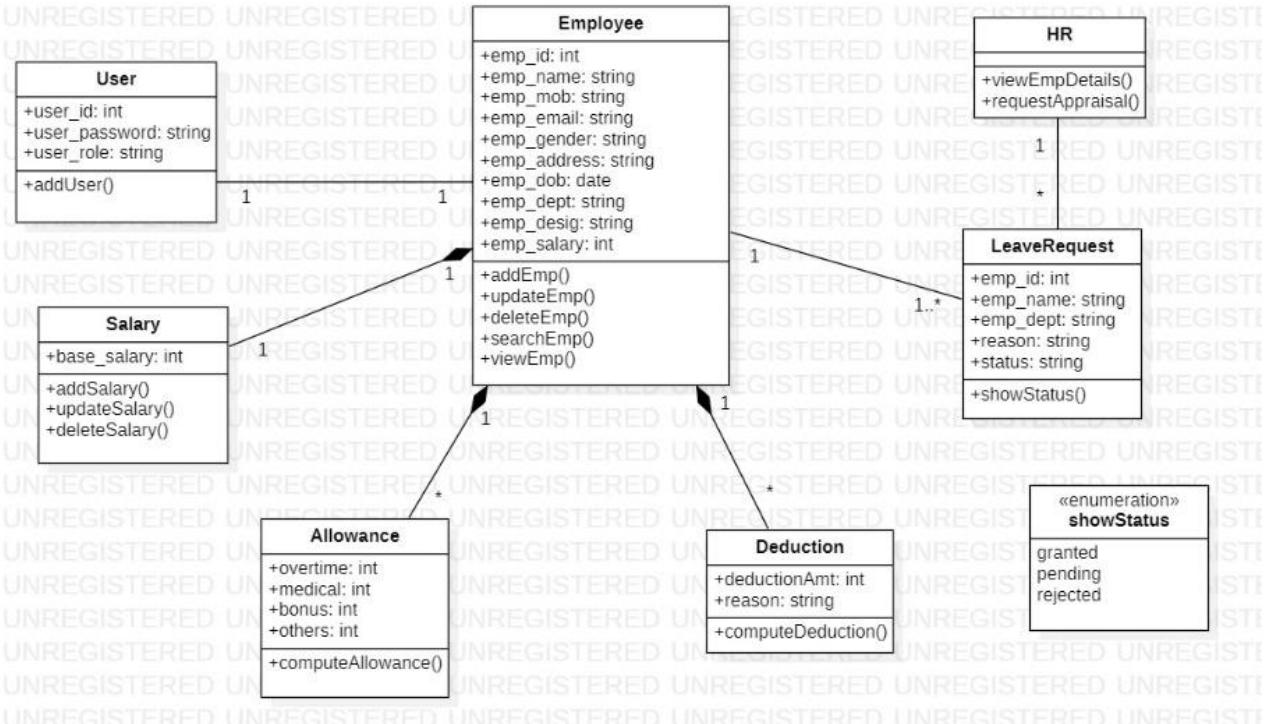
https://github.com/manojvd/Employee-Management-system

ANALYSIS AND DESIGN MODELS:

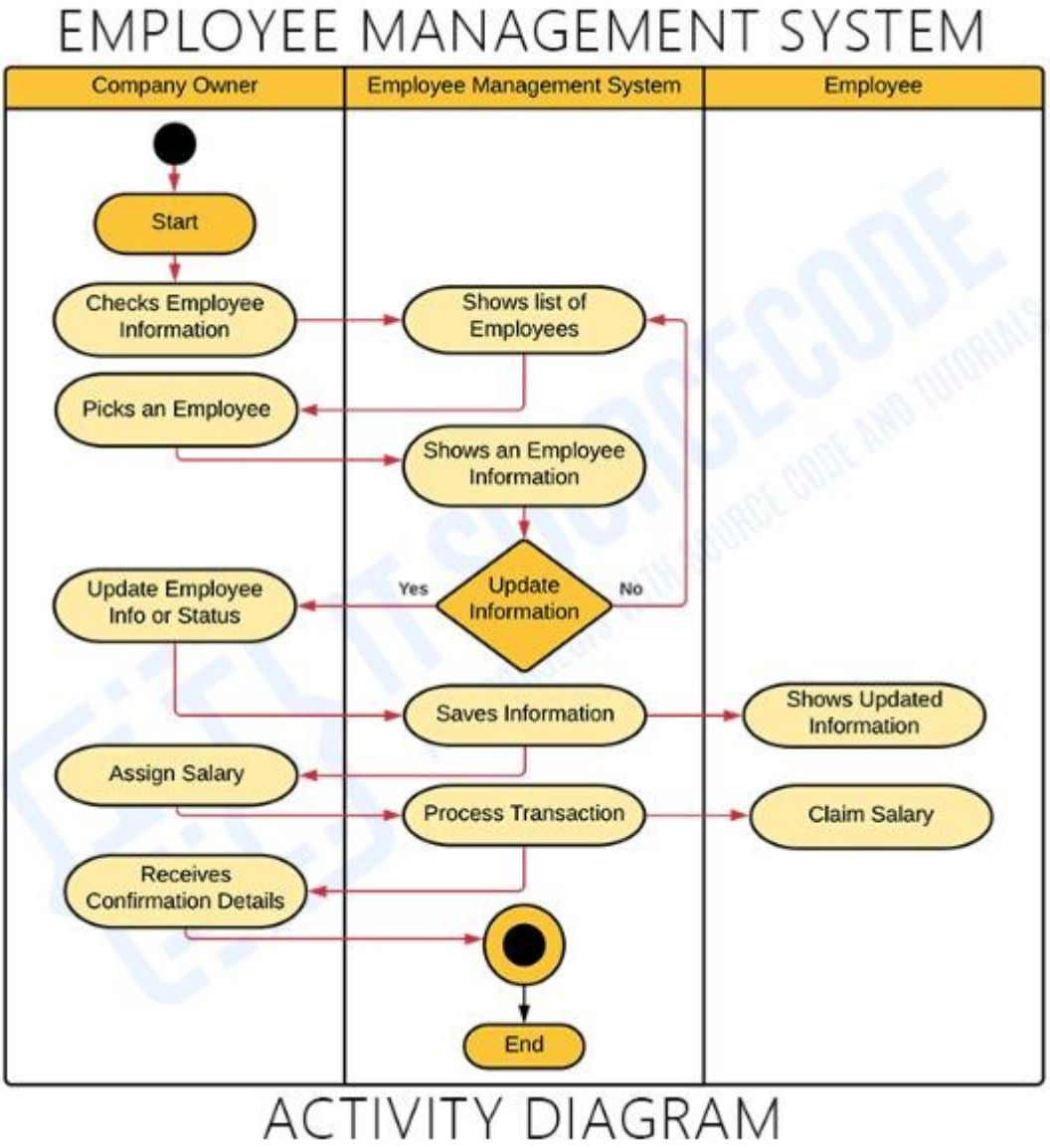
USE CASE DIAGRAM:



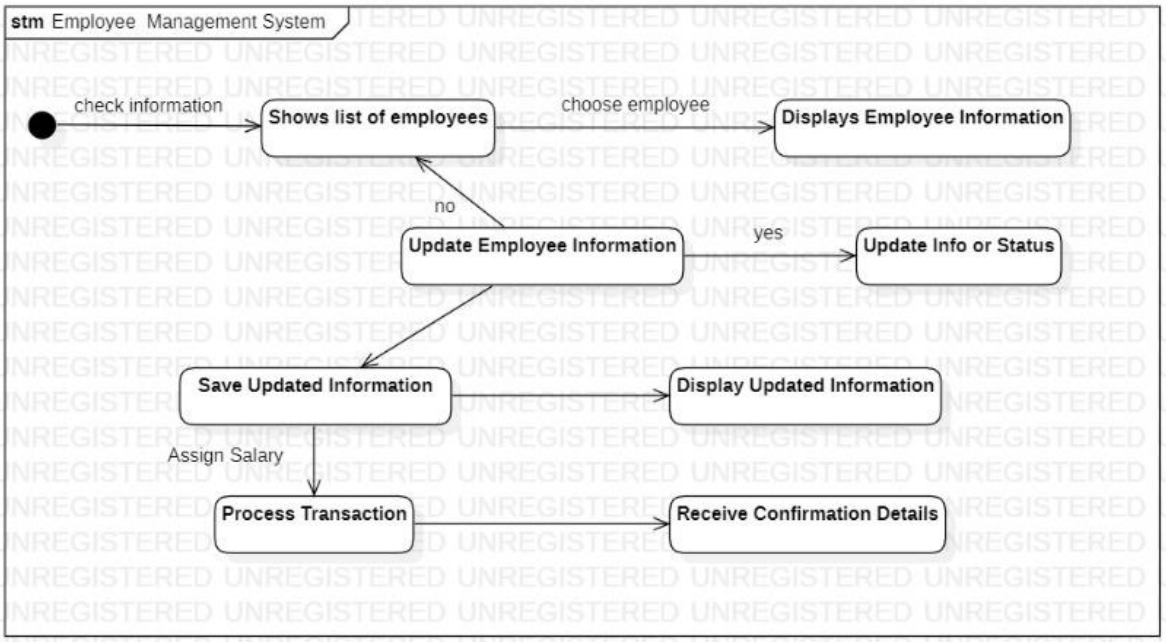
CLASS DIAGRAM:

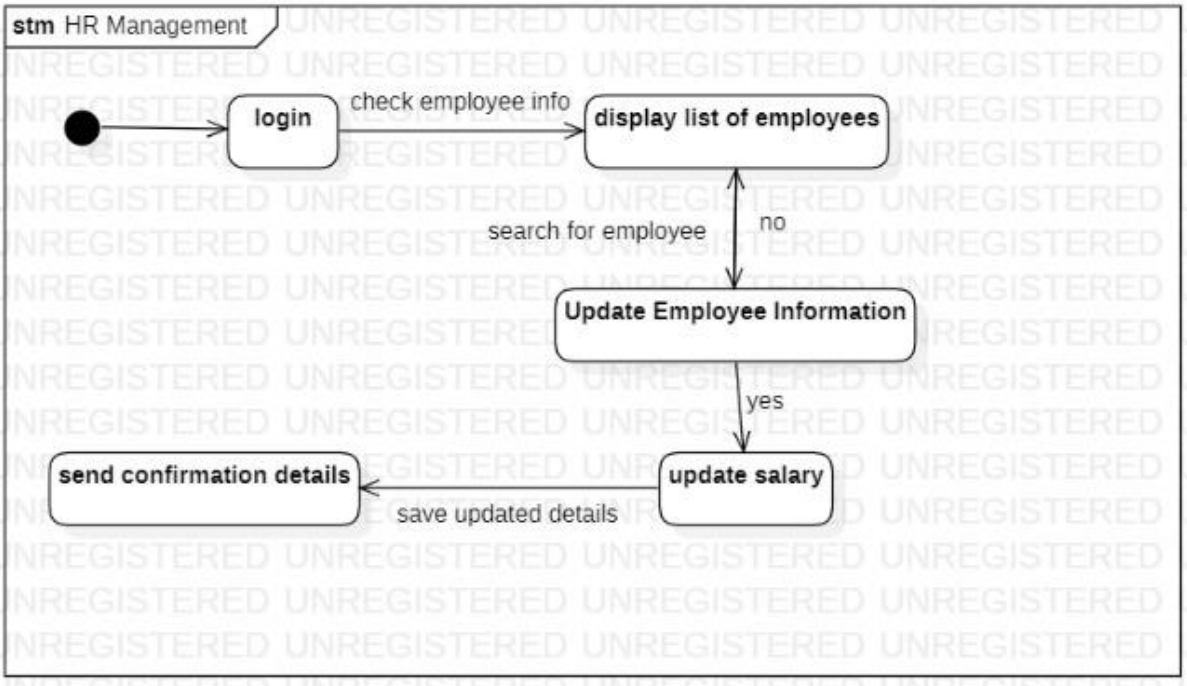


ACTIVITY DIAGRAM:



STATE DIAGRAM:


stm Employee Management System
check information
Shows list of employees
choose employee
Displays Employee Information
no
Update Employee Information
yes
Update Info or Status
Save Updated Information
Display Updated Information
Assign Salary
Process Transaction
Receive Confirmation Details


stm HR Management
login
check employee info
display list of employees
search for employee
no
Update Employee Information
yes
send confirmation details
update salary
save updated details

TOOLS AND FRAMEWORKS USED:

1. The “javax.swing” for the GUI.
2. Mysql database backend, using a JDBC Mysql driver to connect it to the JRE.
3. “java.awt” for events and triggers.
4. “java.sql” for the database backend connection, exception and querying.

DESIGN PRINCIPLES AND DESIGN PATTERNS APPLIED:

SINGLETON DESIGN PATTERN:

The Singleton design pattern is a creational pattern that ensures only one instance of a class is created and provides a global point of access to that instance. For a government official, the Singleton design pattern can be used to ensure that there is only one instance of that official in the system.
To implement the Singleton design pattern for a government official, we create a class that represents the official and ensure that only one instance of that class can be created.

FACTORY DESIGN PATTERN:

For creation of each instance in the landing page;
We have created an interface called Factoryint, which acts as the abstract class to create an object. Each subclass, all of whose methods are called in Factory.java, which is responsible for a specific functionality, is responsible for its relevant instantiation.

OPEN-CLOSE PRINCIPLE:

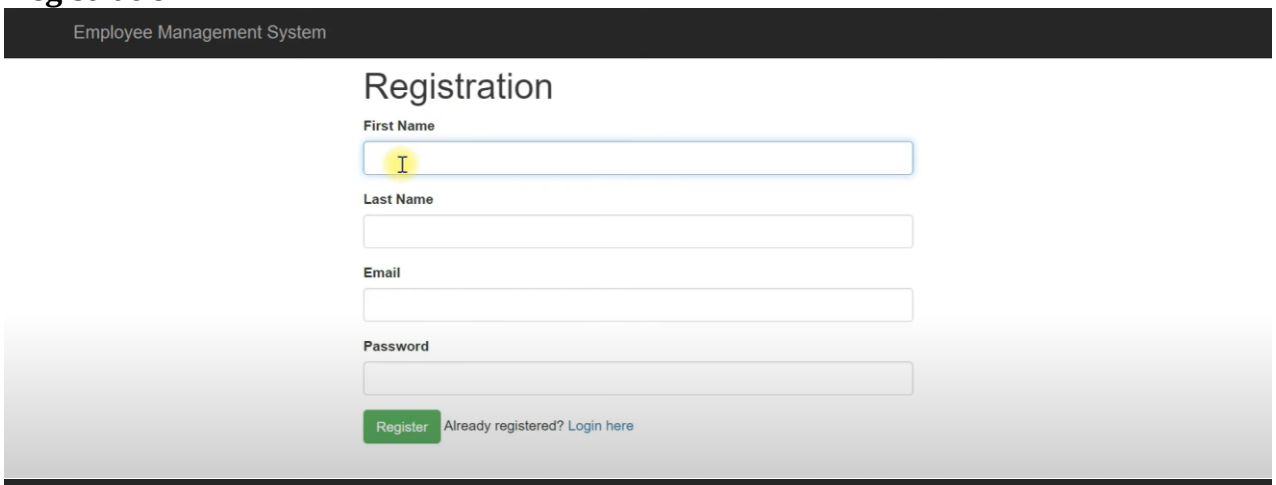
Instead of adding condition check value in fetch_all() and modifying it, we make a new function get_from_db() so that each function can be used to get rows from the database based on whether we need to get rows on condition or not.

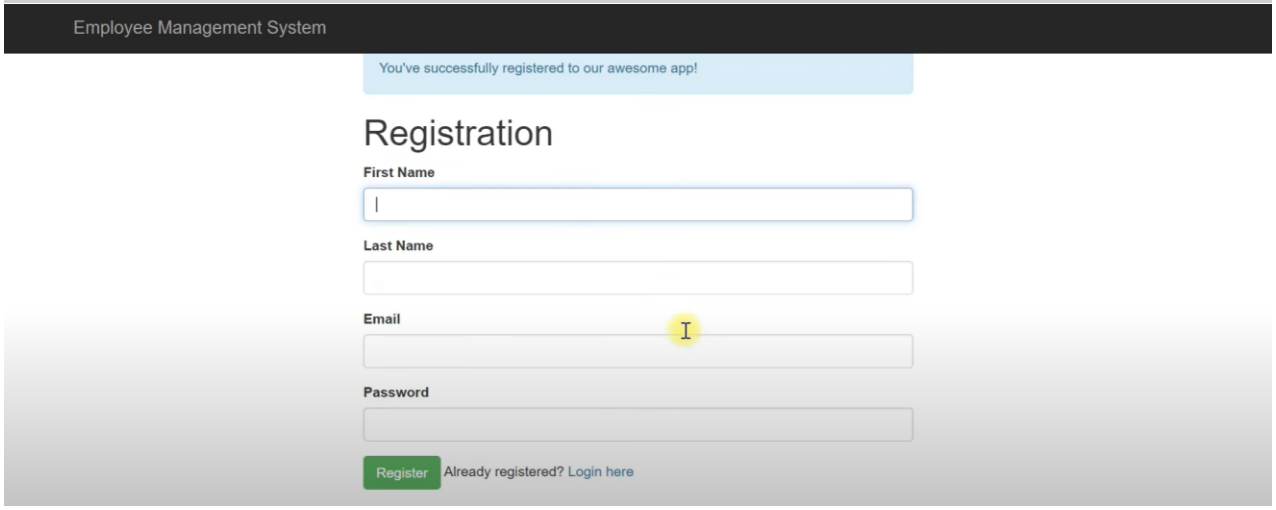
Single RESPO PRINCIPLE FROM SOLID:

The database.java class is responsible only for executing queries related to the database and does not perform any other function.

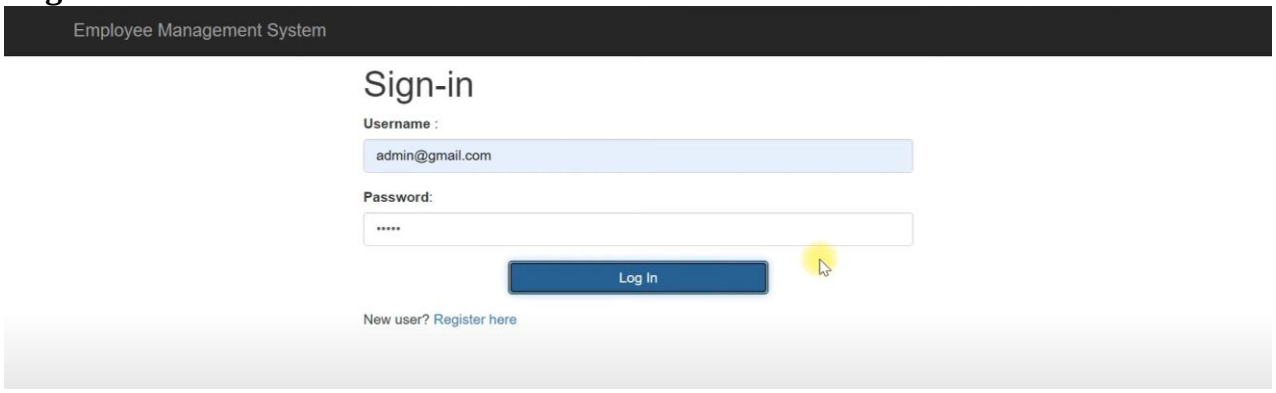
APPLICATION SCREENSHOTS:

Registration

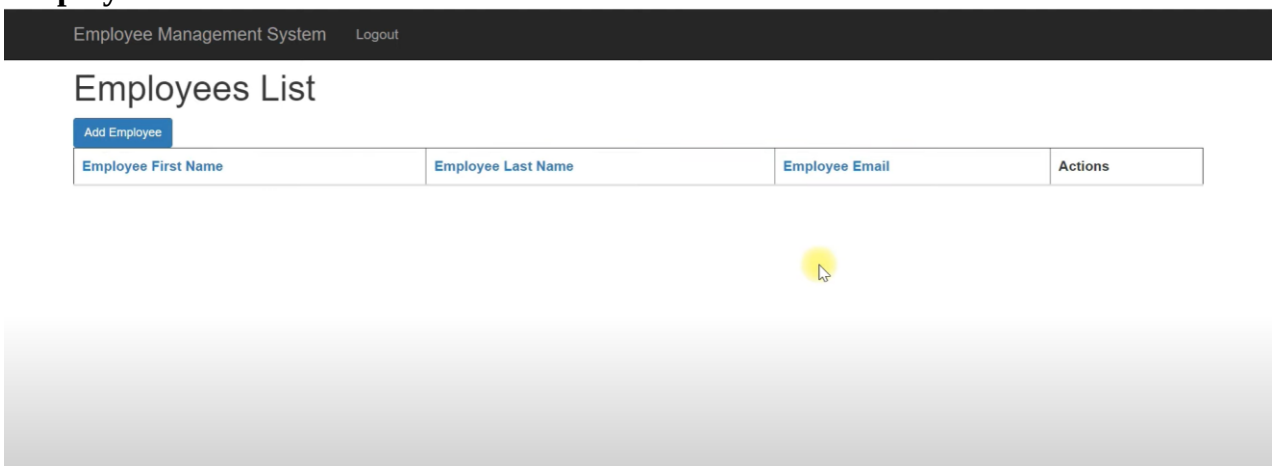


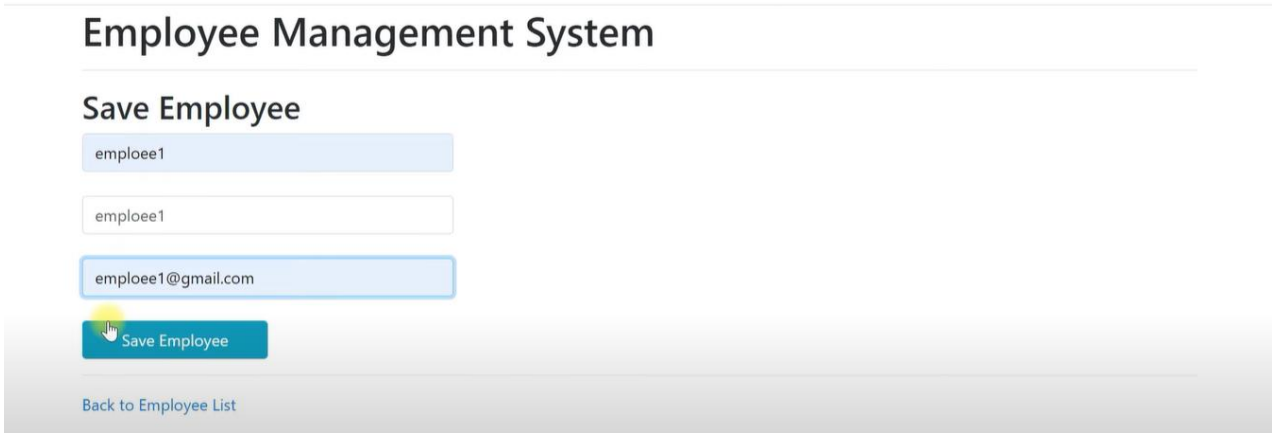


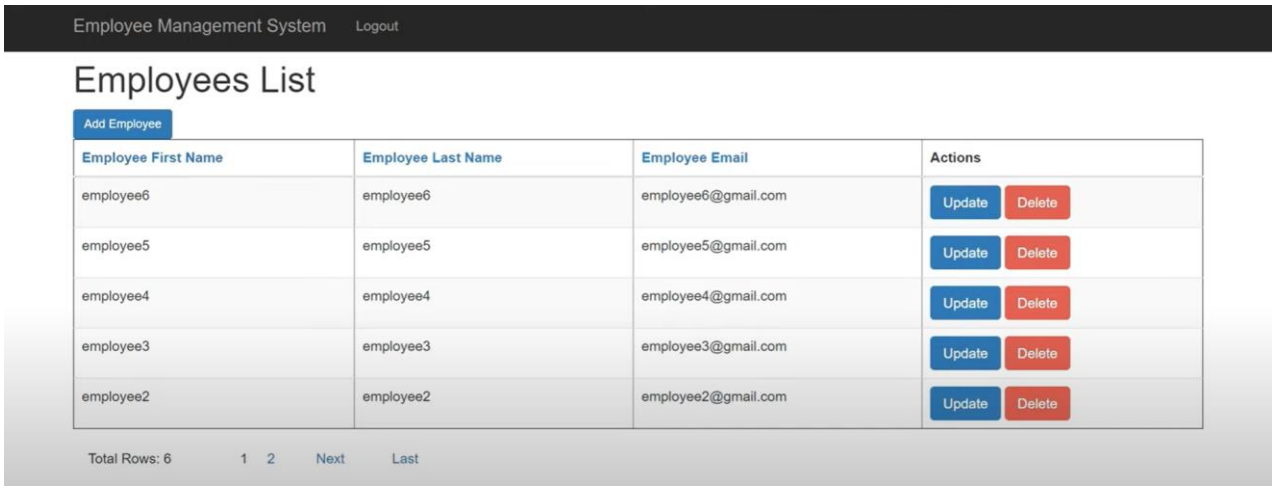
Login



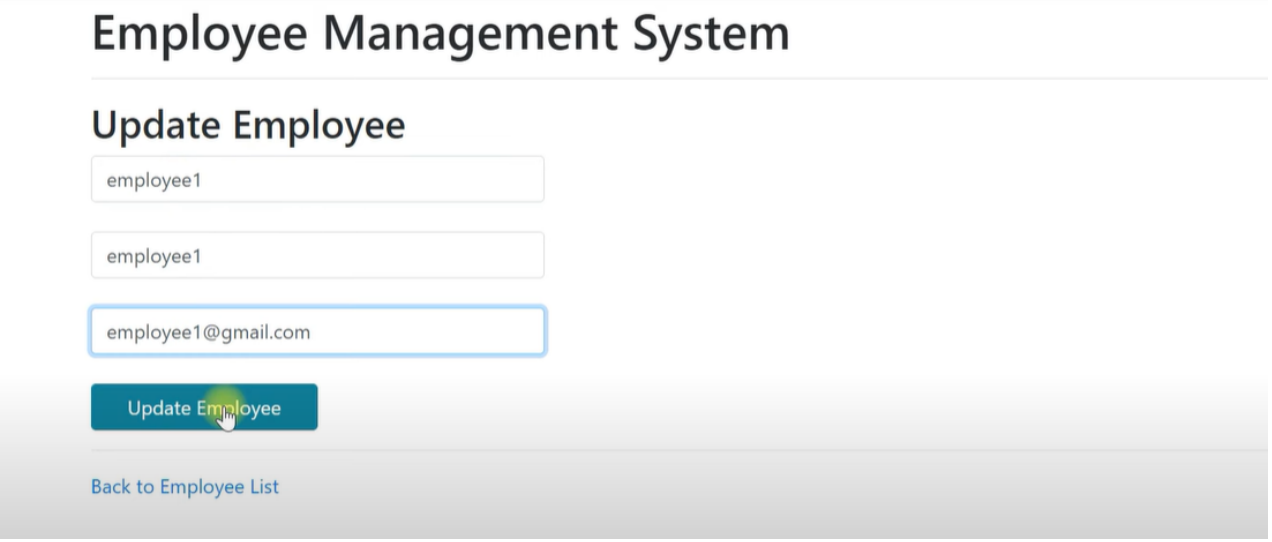
Employee List







Update Employee



Delete employee