EMPLOYEE MANAGEMENT SYSTEM

CAPSTONE PROJECT

Case Study

By

Shivani Sri G

43130661

BE.ECE

EMPLOYEE MANAGEMENT SYSTEM

INTRODUCTION

In many organizations, employee data is managed manually using paperwork or spreadsheets, which is time-consuming and prone to errors. Such traditional systems often lead to data inconsistency, difficulty in retrieval, and security issues.

The Employee Management System is designed to overcome these challenges by providing an automated, digital solution for employee data management. The system allows administrators to efficiently manage employee records and enables employees to securely access their own details through a web-based interface.

ABSTRACT

The Employee Management System (EMS) is a web-based application developed to efficiently manage employee information within an organization. The system enables administrators to store, retrieve, update, and delete employee records in a centralized database. By automating employee-related tasks, the system minimizes manual paperwork, reduces errors, improves data accuracy, and saves time.

The application supports role-based access control, where **Admin users** can manage all employee records, and **Employees (Users)** can securely view their own personal information. The system is developed using **Spring Boot, Java, MySQL, JDBC, HTML, and CSS**, providing a robust backend and a user-friendly web interface.

OBJECTIVES OF THE PROJECT

The main objective of the Employee Management System is to design and develop a user-friendly web-based application that helps organizations manage employee information efficiently and securely.

The specific objectives are:

- To automate employee management tasks such as adding, updating, viewing, and deleting records
- To reduce manual paperwork and human errors
- To provide role-based access for Admin and Employee users
- To allow employees to securely view their own salary, bonus, and experience details
- To ensure fast data retrieval, accuracy, and data security
- To improve overall organizational productivity

CLIENT REQUIREMENT

The client requires a web-based Employee Management System with role-based authentication, where Admin users can perform full CRUD operations on employee records and Employees can securely view their own details. The system must be developed using Spring Boot and Java with JDBC for database connectivity, use MySQL as the backend database, and provide a responsive user interface using HTML and CSS. The application should ensure secure login, efficient data handling, and centralized employee data management.

WHAT WE ARE GOING TO BUILD

We are going to build a **role-based web application** using **Spring Boot (MVC architecture)** and **Java**, integrated with a **MySQL relational database** through **JDBC**. The system will implement **CRUD operations**, **user authentication and authorization**, **session management**, and **database-driven dynamic web pages** using **HTML and CSS**. The application will follow a **three-tier architecture** consisting of presentation, application, and database layers to ensure scalability, security, and efficient data management.

TECHNOLOGIES AND TOOLS USED

- **Programming Language:** Java – used for backend logic and application development
- **Framework:** Spring Boot – used to build the web application with MVC architecture and embedded Tomcat server
- **Database Connectivity:** JDBC (Java Database Connectivity) – used to connect the application with the MySQL database and perform CRUD operations
- **Database:** MySQL – used as the relational database to store employee and user information
- **Frontend Technologies:** HTML and CSS – used to design and style the user interface
- **IDE / Development Tool:** Spring Tool Suite (STS) / Eclipse – used for coding, debugging, and project management
- **Server:** Embedded Apache Tomcat – used to deploy and run the web application
- **Build Tool:** Maven – used for dependency management and project build automation
- **Browser:** Google Chrome – used for testing and accessing the application

SYSTEM REQUIREMENTS

Software Requirements:

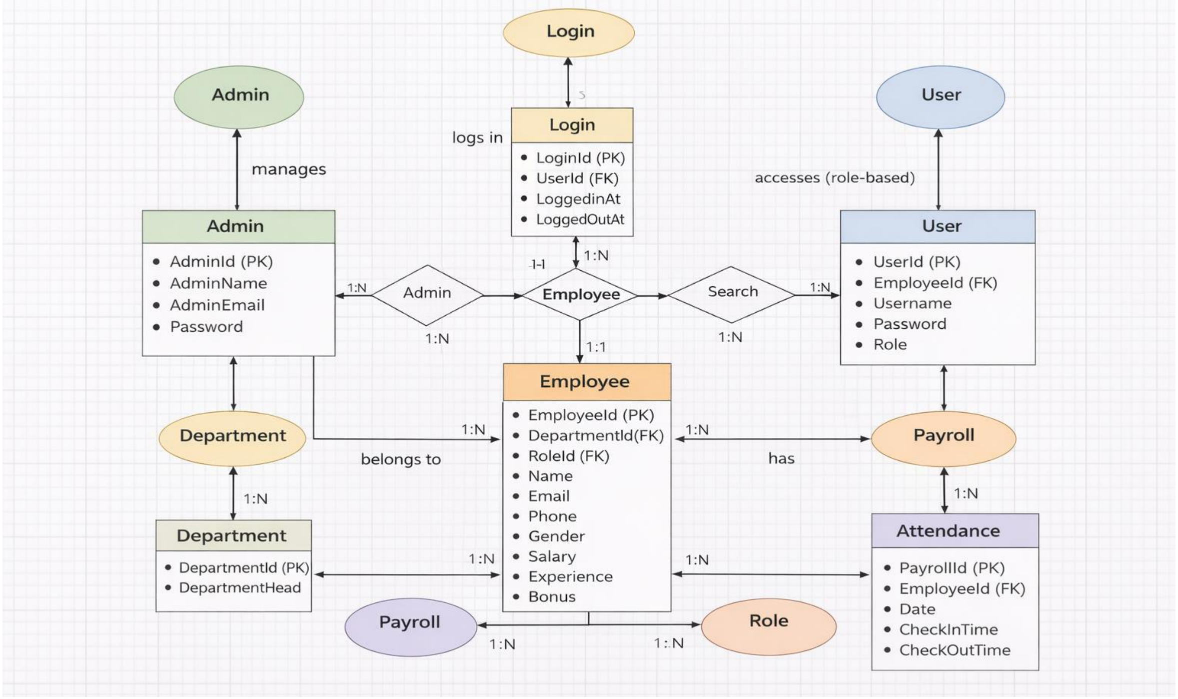
- **Operating System:** Windows / Linux
- **Programming Language:** Java (JDK 8 or above)
- **Framework:** Spring Boot
- **Database:** MySQL
- **Database Connectivity:** JDBC (Java Database Connectivity)
- **Frontend Technologies:** HTML, CSS

- **IDE / Development Tool:** Spring Tool Suite (STS) / Eclipse
- **Build Tool:** Maven
- **Server:** Embedded Apache Tomcat
- **Web Browser:** Google Chrome

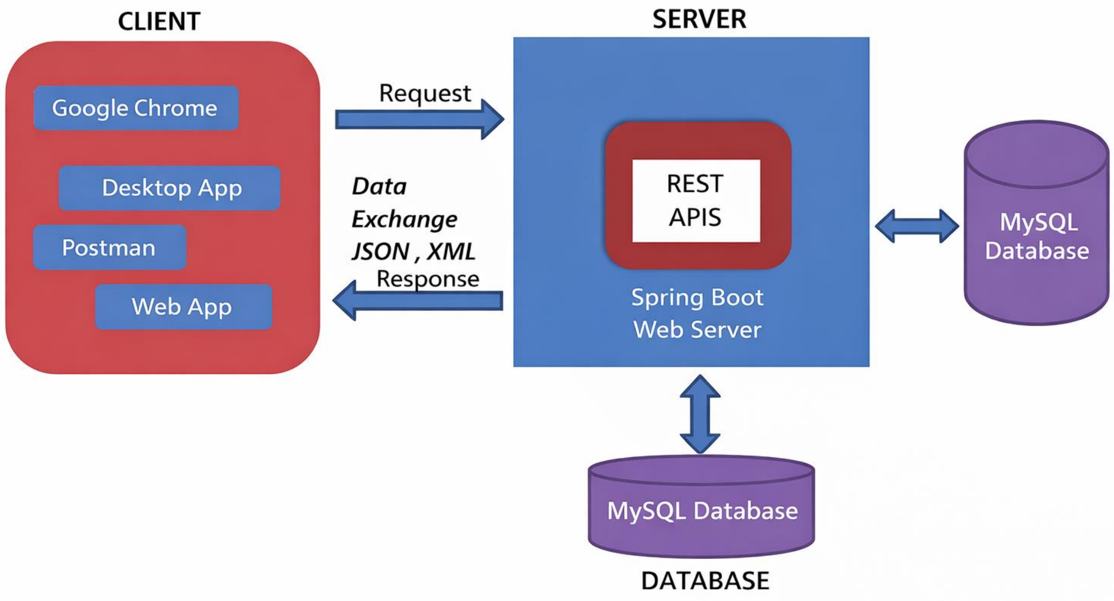
PROJECT MODULE

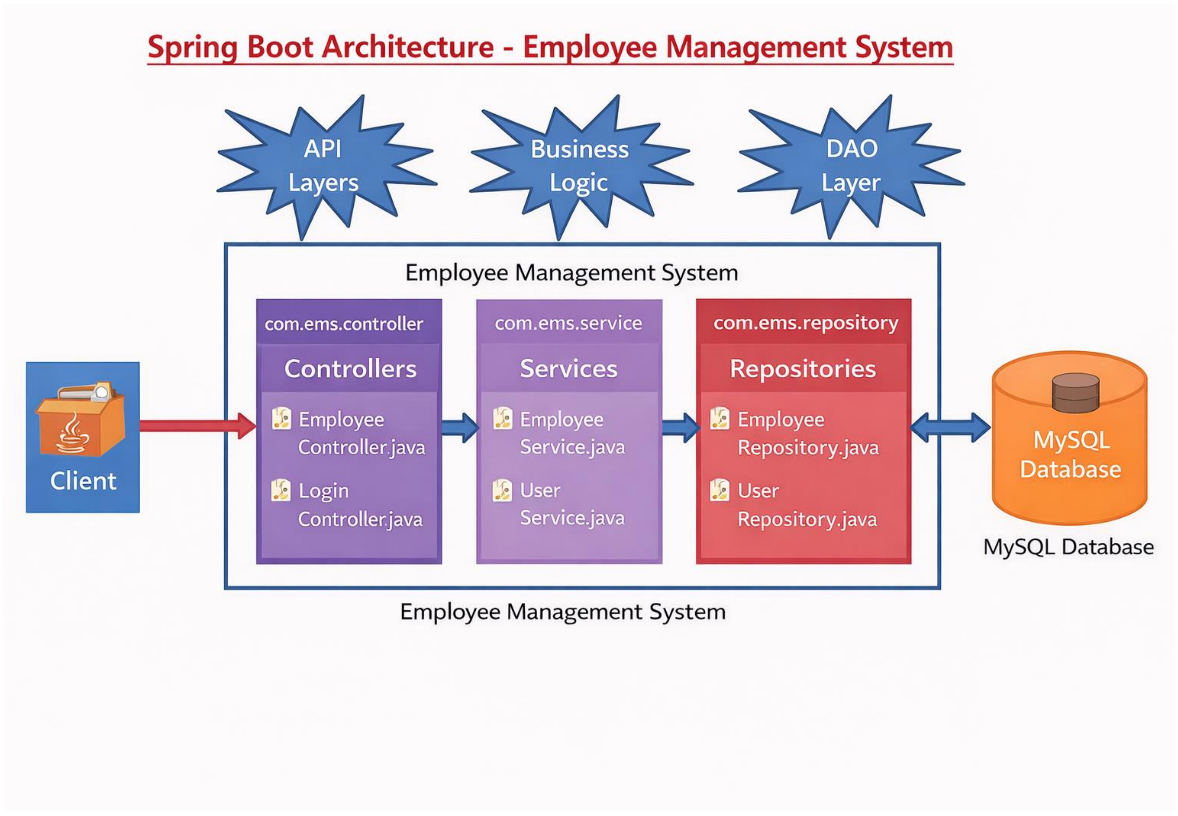
- **Admin Module:**
  The Admin Module allows the administrator to manage the entire system. The admin can add, update, view, and delete employee records and control overall system operations.
- **Employee (User) Module:**
  The Employee Module enables employees to securely access the system and view their own profile details such as salary, bonus, and experience.
- **Login & Authentication Module:**
  This module handles secure login functionality and ensures role-based access control for admin and employee users.
- **Employee Management Module:**
  The Employee Management Module is responsible for performing CRUD operations on employee data, ensuring accurate data handling.
- **Database Management Module:**
  This module manages data storage and retrieval using MySQL with JDBC for secure and efficient database connectivity.

ER DIAGRAM



Client – Server Architecture





LOGIN MODULE

- Provides secure authentication for users by validating username and password credentials against records stored in the MySQL database.
- Implements role-based access control to distinguish between Admin and Employee users and restrict system functionalities accordingly.
- Ensures that only authorized users can access protected modules after successful login.
- Redirects authenticated users to their respective dashboards based on their assigned roles.
- Maintains user session information to support secure navigation within the application.
- Serves as the central entry point for accessing all features of the Employee Management System.

ADMIN MODULE

- In the Admin Module, the Admin can access the system after successful login. The Admin can manage employee records using REST APIs and test the operations using the Postman tool.
- The Admin module supports CRUD operations where the Admin can save employee details using **POST mapping**, retrieve all employee records using **GET mapping**, update employee information using **PUT mapping**, and delete employee records using **DELETE mapping**.
- The Admin module contains service and test methods such as **Save Employee**, **List of Employees**, **Update Employee**, and **Delete Employee** to ensure proper functionality of the application.
- After completing the Admin module, child modules such as the **Employee (User) Module** are created and linked to the Admin module for role-based access and management.

EMPLOYEE MODULE

- After completing the Login Module, we can move into the Employee Module.
- In the Employee Module, employee data can be added, viewed, updated, and deleted using the **Postman Tool**.
- The Employee Module allows adding employee details using **POST Mapping**, retrieving all employee details using **GET Mapping**, updating employee information using **PUT Mapping**, and deleting employee records using **DELETE Mapping**.
- We can use **JUnit Test cases** to test all methods in the Employee Module to ensure proper functionality.
- After completing the Employee Module, we proceed with the **User Module**, which acts as a child module and provides role-based access to employee information.

USER MODULE

- In the User Module, the user (employee) can access the system after successful login. Users can view their own employee details securely using REST APIs.
- The User module allows users to retrieve their personal information using **GET mapping**. Users have restricted access and are not allowed to add, update, or delete other employee records.
- The User module contains service and test methods such as **View Profile**, **View Salary**, **View Bonus**, and **View Experience**, ensuring secure and role-based data access.
- The User module works as a child module of the Admin Module, where each user account is linked to a specific employee record for controlled and secure access.

HTTP / HTTPS Request Methods

Login Module

HTTP Method	URL	Description
POST	http://localhost:8080/login	Authenticates admin or user
POST	http://localhost:8080/logout	Logs out the user

Admin Module

HTTP Method	URL	Description
POST	http://localhost:8080/admin/save	Saves admin details
GET	http://localhost:8080/admin/all	Retrieves all admin records
PUT	http://localhost:8080/admin/update/{id}	Updates admin details

DELETE	http://localhost:8080/admin/delete/{id}	Deletes admin by ID

Employee Management Module

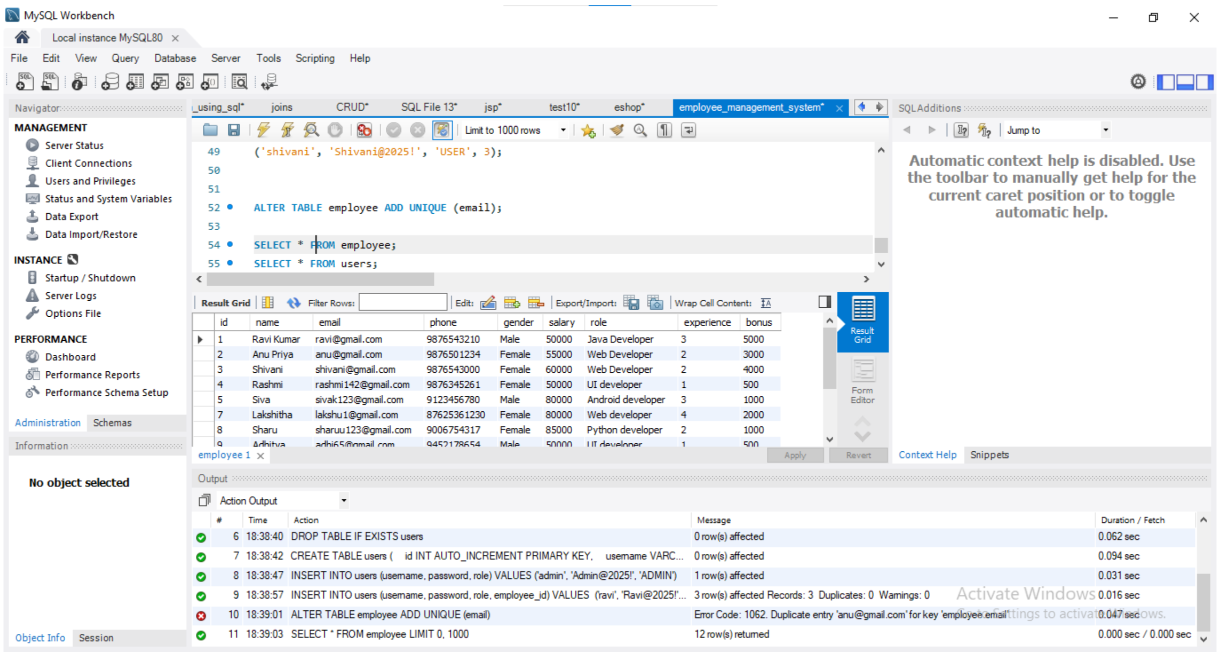
HTTP Method	URL	Description
POST	http://localhost:8080/employees/save	Adds a new employee
GET	http://localhost:8080/employees	Retrieves all employee records
GET	http://localhost:8080/employees/{id}	Retrieves employee by ID
PUT	http://localhost:8080/employees/update/{id}	Updates employee details
DELETE	http://localhost:8080/employees/delete/{id}	Deletes employee record

User Module

HTTP Method	URL	Description
POST	http://localhost:8080/user/login	User login authentication
GET	http://localhost:8080/user/profile/{id}	Retrieves user profile
GET	http://localhost:8080/user/salary/{id}	Retrieves user salary
GET	http://localhost:8080/user/bonus/{id}	Retrieves user bonus

DATA DICTIONARY

ADMIN DATABASE



No object selected

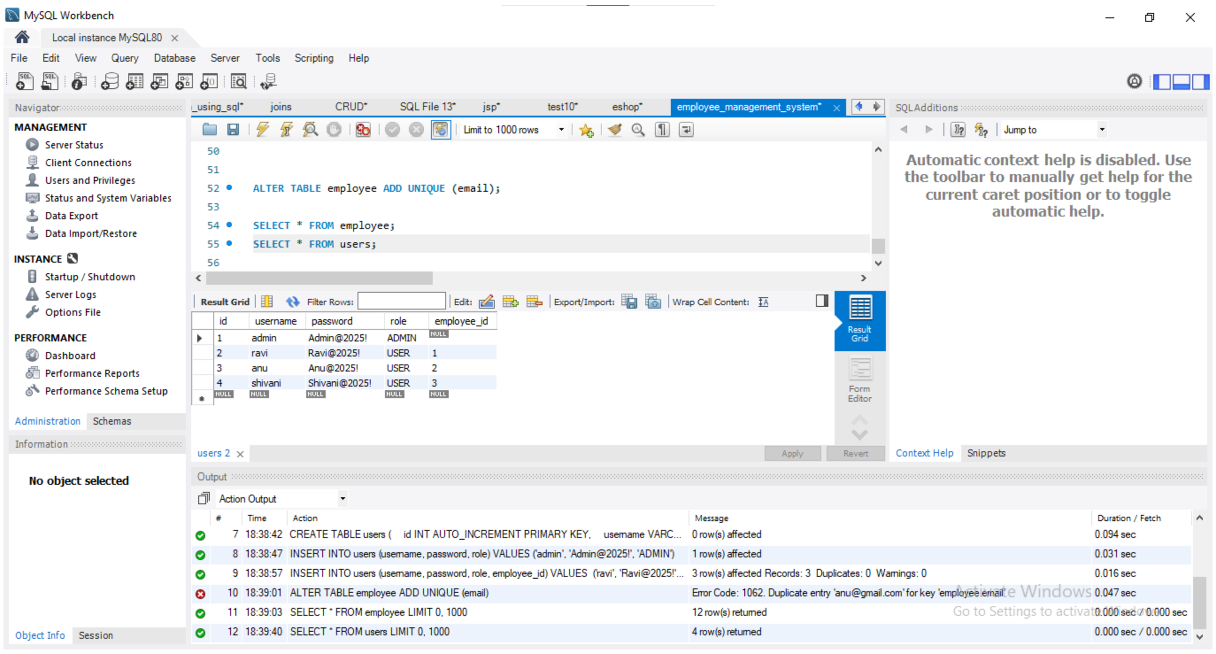
```sql
('shivani', 'Shivani@2025!', 'USER', 3);


ALTER TABLE employee ADD UNIQUE (email);

SELECT * FROM employee;
SELECT * FROM users;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

id	name	email	phone	gender	salary	role	experience	bonus
1	Ravi Kumar	ravi@gmail.com	9876543210	Male	50000	Java Developer	3	5000
2	Anu Priya	anu@gmail.com	9876501234	Female	55000	Web Developer	2	3000
3	Shivani	shivani@gmail.com	9876543000	Female	60000	Web Developer	2	4000
4	Rashmi	rashmi142@gmail.com	9876345261	Female	50000	UI developer	1	500
5	Siva	sivak123@gmail.com	9123456780	Male	80000	Android developer	3	1000
7	Lakshitha	lakshu1@gmail.com	87625361230	Female	80000	Web developer	4	2000
8	Sharu	sharuu123@gmail.com	9006754317	Female	85000	Python developer	2	1000
9	Adhitya	adhi65@gmail.com	9452178654	Male	50000	UI developer	1	500

```
#   Time      Action                                                                      Message                                                                     Duration / Fetch
6   18:38:40  DROP TABLE IF EXISTS users                                                  0 row(s) affected                                                           0.062 sec
7   18:38:42  CREATE TABLE users (    id INT AUTO_INCREMENT PRIMARY KEY,    username VARC... 0 row(s) affected                                                        0.094 sec
8   18:38:47  INSERT INTO users (username, password, role) VALUES ('admin', 'Admin@2025!', 'ADMIN')  1 row(s) affected                                               0.031 sec
9   18:38:57  INSERT INTO users (username, password, role, employee_id) VALUES  ('ravi', 'Ravi@2025!'...  3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.016 sec
10  18:39:01  ALTER TABLE employee ADD UNIQUE (email)                                     Error Code: 1062. Duplicate entry 'anu@gmail.com' for key 'employee.email'  0.047 sec
11  18:39:03  SELECT * FROM employee LIMIT 0, 1000                                        12 row(s) returned                                                          0.000 sec / 0.000 sec
```

USER DATABASE



No object selected

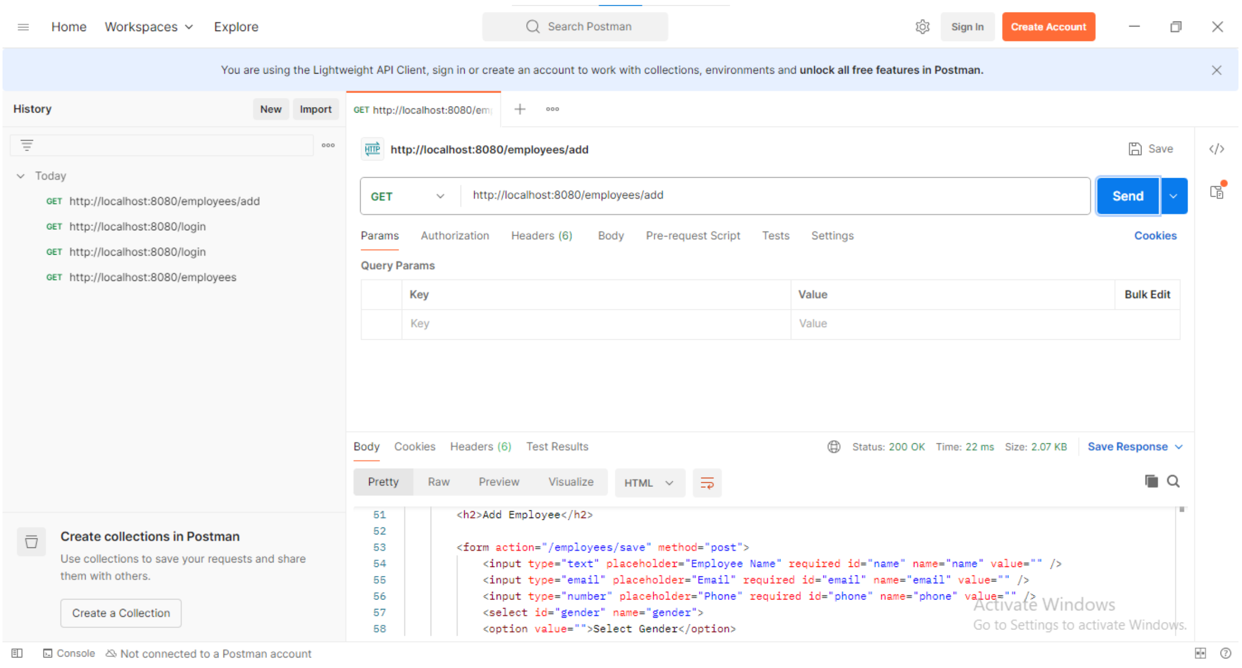
```sql


ALTER TABLE employee ADD UNIQUE (email);

SELECT * FROM employee;
SELECT * FROM users;

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

id	username	password	role	employee_id
1	admin	Admin@2025!	ADMIN	NULL
2	ravi	Ravi@2025!	USER	1
3	anu	Anu@2025!	USER	2
4	shivani	Shivani@2025!	USER	3
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                                      Message                                                                     Duration / Fetch
7   18:38:42  CREATE TABLE users (    id INT AUTO_INCREMENT PRIMARY KEY,    username VARC... 0 row(s) affected                                                        0.094 sec
8   18:38:47  INSERT INTO users (username, password, role) VALUES ('admin', 'Admin@2025!', 'ADMIN')  1 row(s) affected                                               0.031 sec
9   18:38:57  INSERT INTO users (username, password, role, employee_id) VALUES  ('ravi', 'Ravi@2025!'...  3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.016 sec
10  18:39:01  ALTER TABLE employee ADD UNIQUE (email)                                     Error Code: 1062. Duplicate entry 'anu@gmail.com' for key 'employee.email'  0.047 sec
11  18:39:03  SELECT * FROM employee LIMIT 0, 1000                                        12 row(s) returned                                                          0.000 sec / 0.000 sec
12  18:39:40  SELECT * FROM users LIMIT 0, 1000                                           4 row(s) returned                                                           0.000 sec / 0.000 sec
```

POST METHOD FOR ADD DATA



You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

History — Today:
- GET http://localhost:8080/employees/add
- GET http://localhost:8080/login
- GET http://localhost:8080/login
- GET http://localhost:8080/employees

GET http://localhost:8080/employees/add

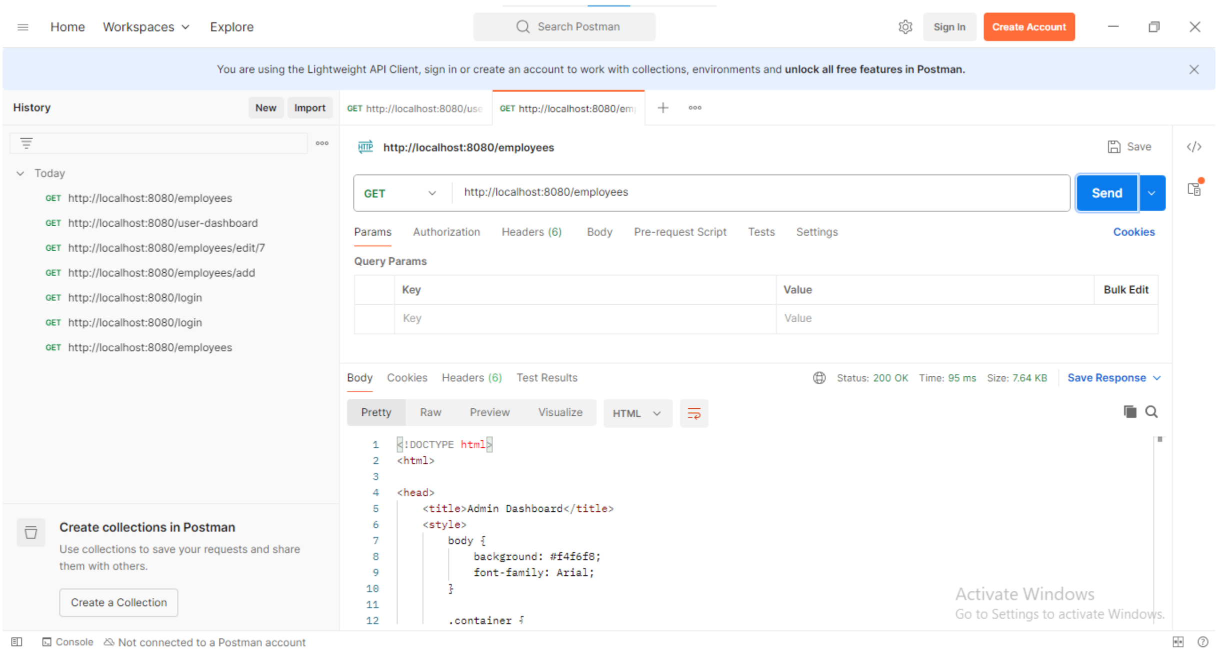
Query Params

Key	Value
Key	Value

Status: 200 OK Time: 22 ms Size: 2.07 KB

```
<h2>Add Employee</h2>

<form action="/employees/save" method="post">
    <input type="text" placeholder="Employee Name" required id="name" name="name" value="" />
    <input type="email" placeholder="Email" required id="email" name="email" value="" />
    <input type="number" placeholder="Phone" required id="phone" name="phone" value="" />
    <select id="gender" name="gender">
    <option value="">Select Gender</option>
```

GET METHOD FOR GET ALL DATA



You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

History — Today:
- GET http://localhost:8080/employees
- GET http://localhost:8080/user-dashboard
- GET http://localhost:8080/employees/edit/7
- GET http://localhost:8080/employees/add
- GET http://localhost:8080/login
- GET http://localhost:8080/login
- GET http://localhost:8080/employees

GET http://localhost:8080/employees

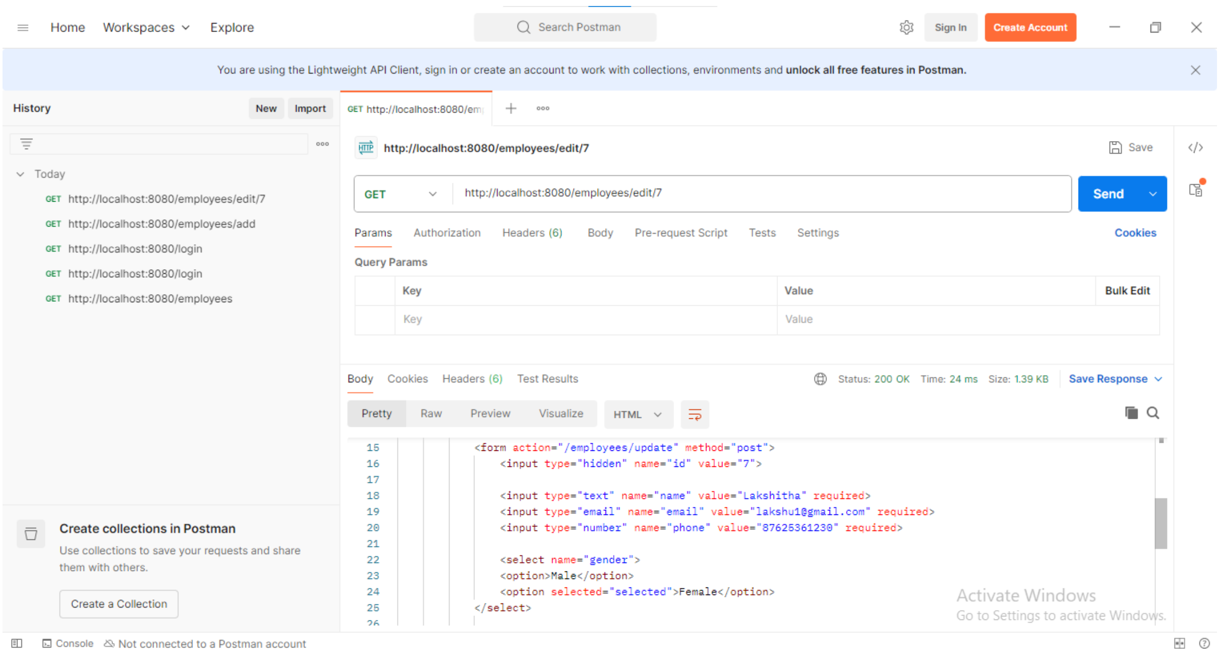
Query Params

Key	Value
Key	Value

Status: 200 OK Time: 95 ms Size: 7.64 KB

```
<!DOCTYPE html>
<html>

<head>
    <title>Admin Dashboard</title>
    <style>
        body {
            background: #f4f6f8;
            font-family: Arial;
        }

        .container {
```

PUT METHOD FOR UPDATE DATA


You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

History

Today

- GET http://localhost:8080/employees/edit/7
- GET http://localhost:8080/employees/add
- GET http://localhost:8080/login
- GET http://localhost:8080/login
- GET http://localhost:8080/employees

Create collections in Postman

Use collections to save your requests and share them with others.

http://localhost:8080/employees/edit/7

GET http://localhost:8080/employees/edit/7

Query Params

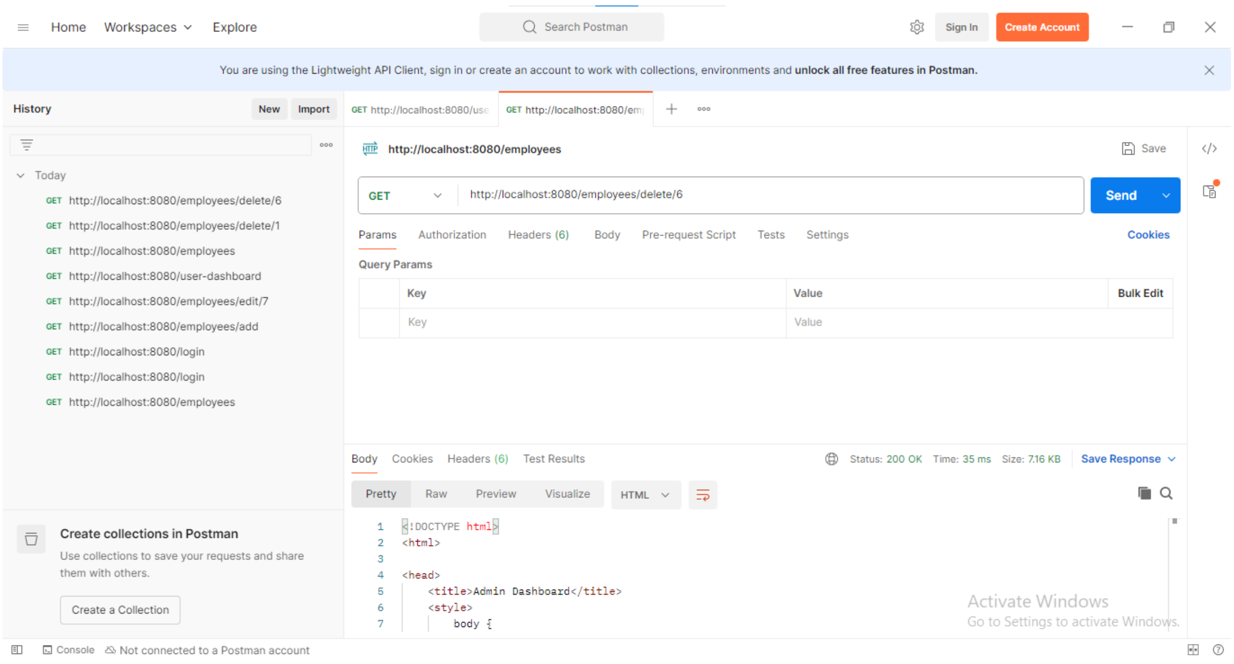
Key	Value
Key	Value

Status: 200 OK Time: 24 ms Size: 1.39 KB

```
<form action="/employees/update" method="post">
    <input type="hidden" name="id" value="7">

    <input type="text" name="name" value="Lakshitha" required>
    <input type="email" name="email" value="lakshu1@gmail.com" required>
    <input type="number" name="phone" value="87625361230" required>

    <select name="gender">
    <option>Male</option>
    <option selected="selected">Female</option>
</select>
```

Not connected to a Postman account

DELETE METHOD FOR DELETE DATA


You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

History

Today

- GET http://localhost:8080/employees/delete/6
- GET http://localhost:8080/employees/delete/1
- GET http://localhost:8080/employees
- GET http://localhost:8080/user-dashboard
- GET http://localhost:8080/employees/edit/7
- GET http://localhost:8080/employees/add
- GET http://localhost:8080/login
- GET http://localhost:8080/login
- GET http://localhost:8080/employees

Create collections in Postman

Use collections to save your requests and share them with others.

http://localhost:8080/employees

GET http://localhost:8080/employees/delete/6

Query Params

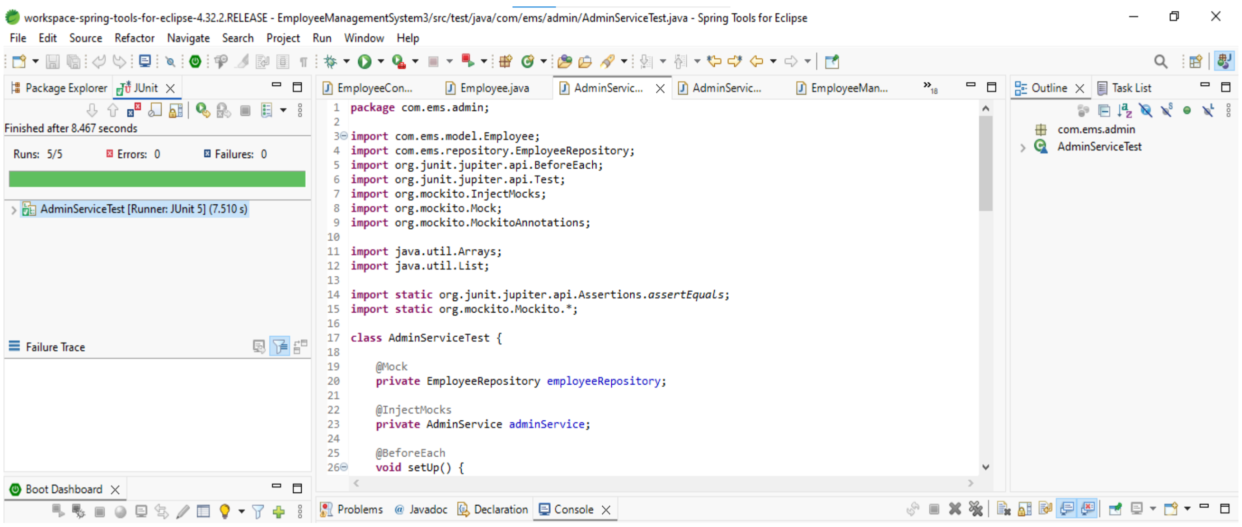
Key	Value
Key	Value

Status: 200 OK Time: 35 ms Size: 7.16 KB

```
<!DOCTYPE html>
<html>

<head>
    <title>Admin Dashboard</title>
    <style>
        body {
```

Not connected to a Postman account

JUNIT TEST CASES FOR ADMIN MODULE

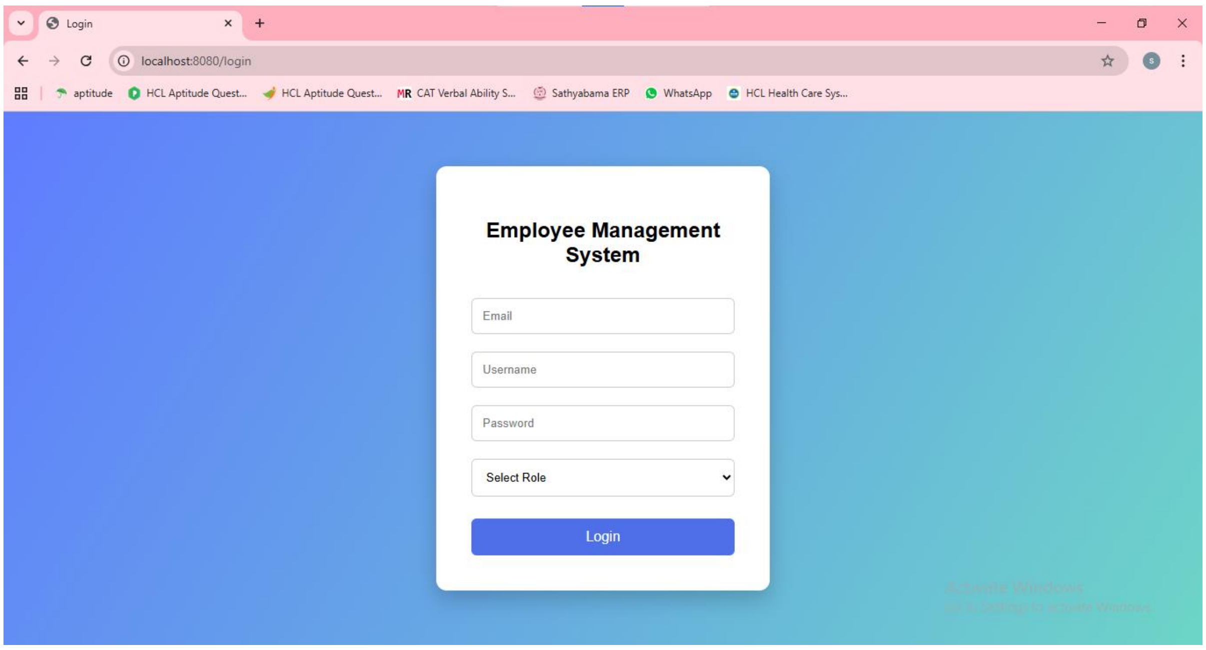

workspace-spring-tools-for-eclipse-4.32.2.RELEASE - EmployeeManagementSystem3/src/test/java/com/ems/admin/AdminServiceTest.java - Spring Tools for Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
EmployeeCon...
Employee.java
AdminServic...
AdminServic...
EmployeeMan...
Outline
Task List
com.ems.admin
AdminServiceTest
Failure Trace
Boot Dashboard
Problems
Javadoc
Declaration
Console

```
Finished after 8.467 seconds
Runs: 5/5    Errors: 0    Failures: 0
AdminServiceTest [Runner: JUnit 5] (7.510 s)
```

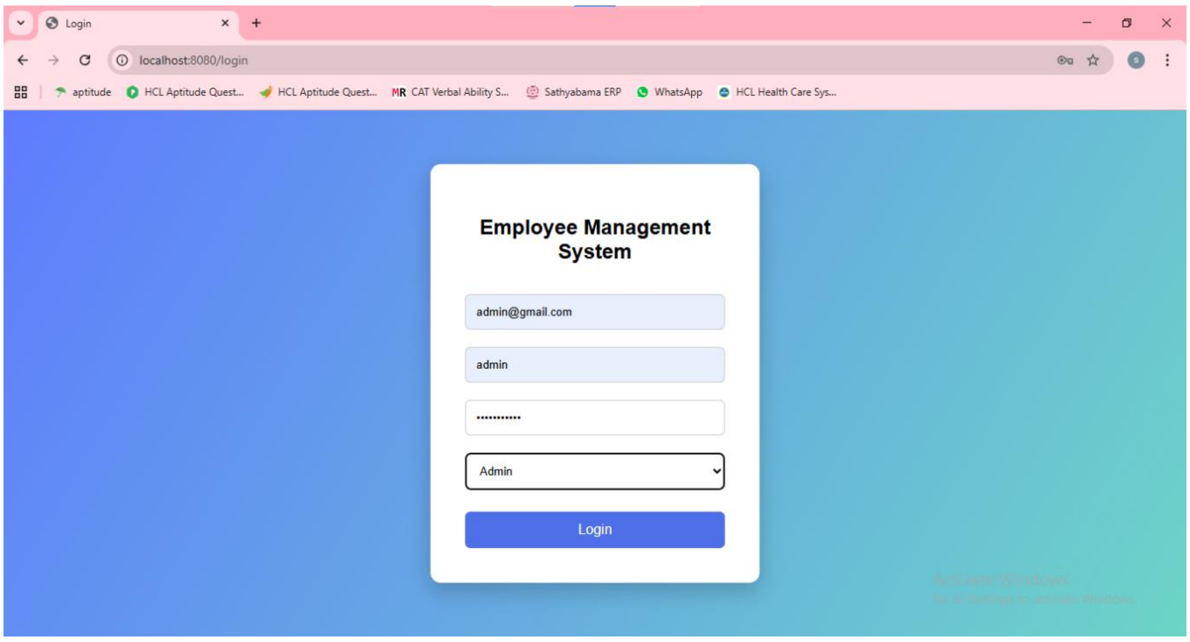
```java
package com.ems.admin;

import com.ems.model.Employee;
import com.ems.repository.EmployeeRepository;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

class AdminServiceTest {

    @Mock
    private EmployeeRepository employeeRepository;

    @InjectMocks
    private AdminService adminService;

    @BeforeEach
    void setUp() {
```

PROJECT OUTPUT

Login Page



Login Page (Admin)



Employee Management System

admin@gmail.com

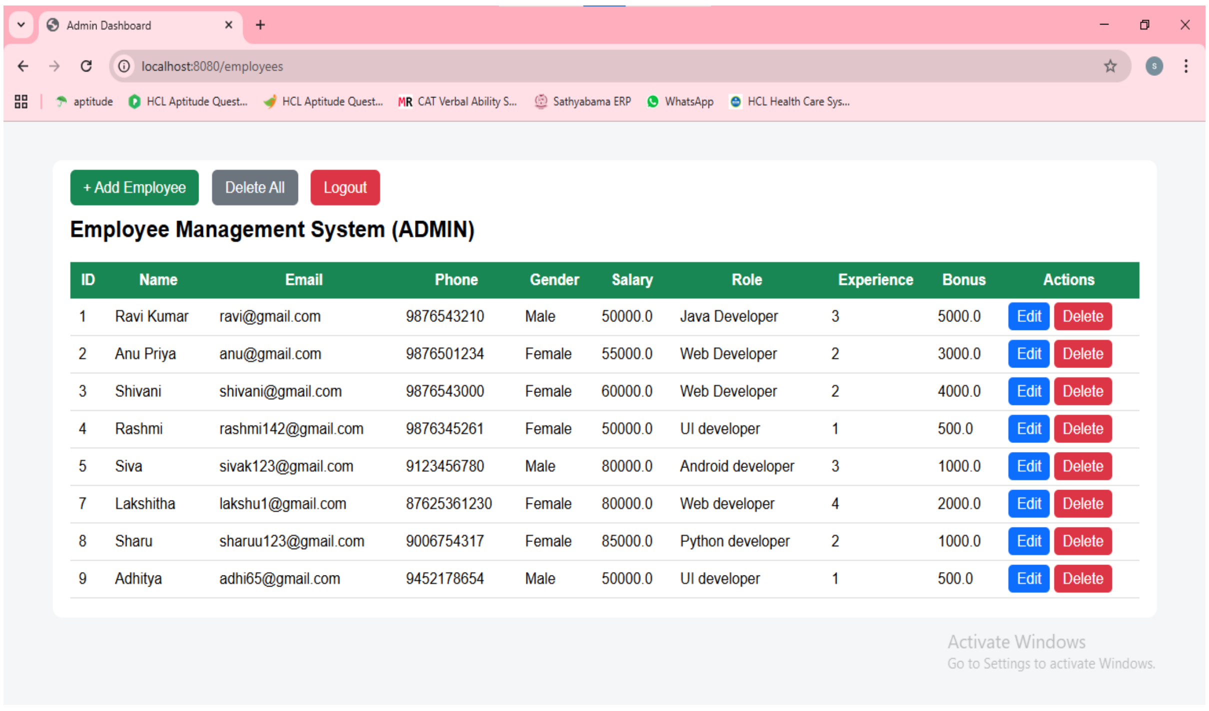
admin

•••••••••••

Admin

Login

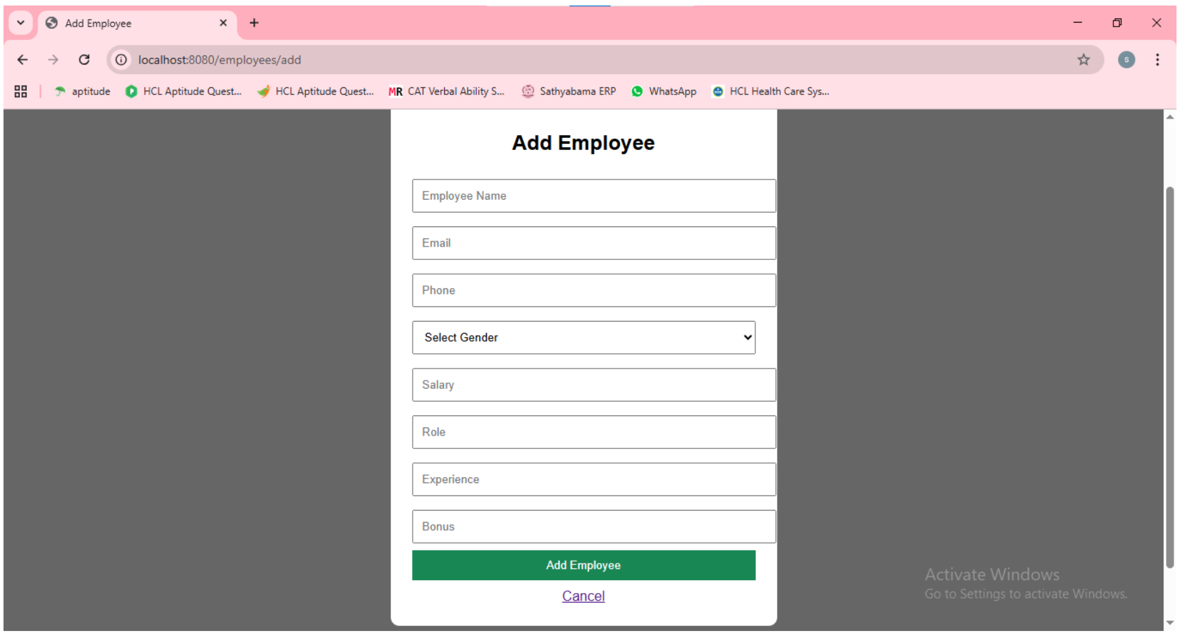
Employees List



Employee Management System (ADMIN)

ID	Name	Email	Phone	Gender	Salary	Role	Experience	Bonus	Actions
1	Ravi Kumar	ravi@gmail.com	9876543210	Male	50000.0	Java Developer	3	5000.0	Edit Delete
2	Anu Priya	anu@gmail.com	9876501234	Female	55000.0	Web Developer	2	3000.0	Edit Delete
3	Shivani	shivani@gmail.com	9876543000	Female	60000.0	Web Developer	2	4000.0	Edit Delete
4	Rashmi	rashmi142@gmail.com	9876345261	Female	50000.0	UI developer	1	500.0	Edit Delete
5	Siva	sivak123@gmail.com	9123456780	Male	80000.0	Android developer	3	1000.0	Edit Delete
7	Lakshitha	lakshu1@gmail.com	87625361230	Female	80000.0	Web developer	4	2000.0	Edit Delete
8	Sharu	sharuu123@gmail.com	9006754317	Female	85000.0	Python developer	2	1000.0	Edit Delete
9	Adhitya	adhi65@gmail.com	9452178654	Male	50000.0	UI developer	1	500.0	Edit Delete

Add Employee Page



Add Employee

Employee Name

Email

Phone

Select Gender

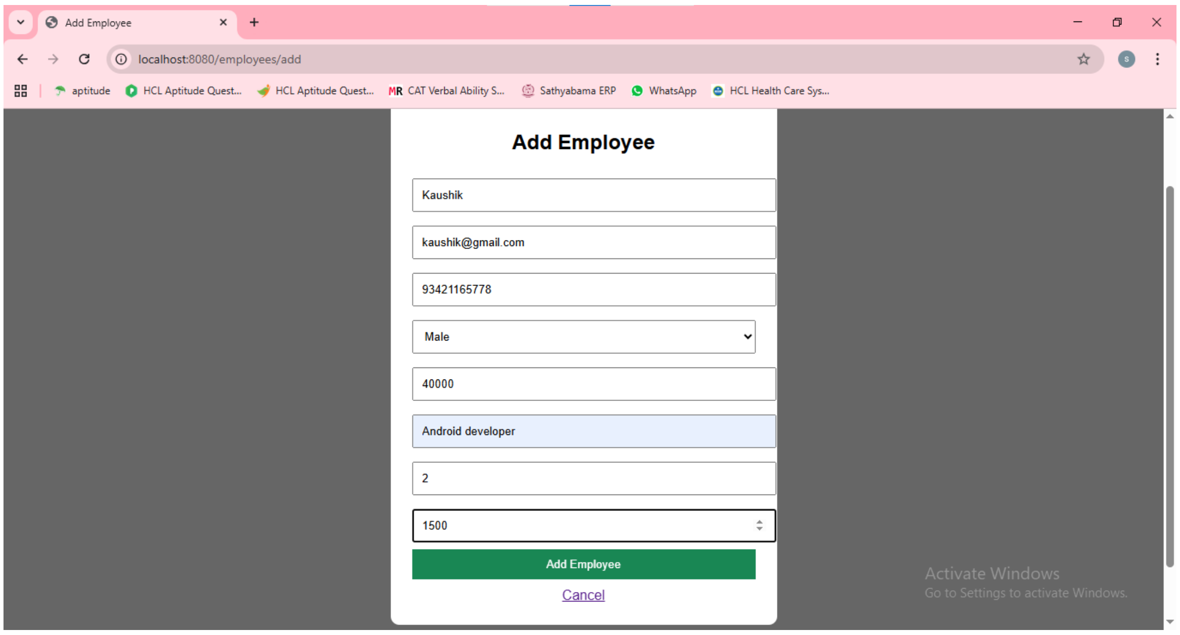
Salary

Role

Experience

Bonus

After filling Details



Add Employee

Kaushik

kaushik@gmail.com

93421165778

Male

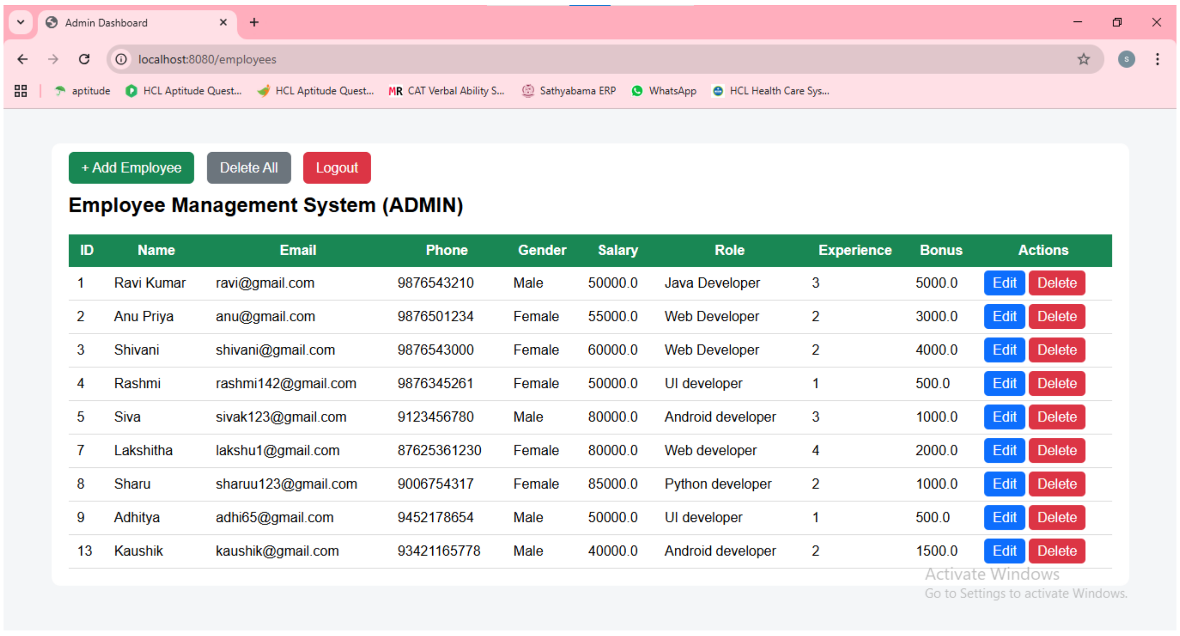
40000

Android developer

2

1500

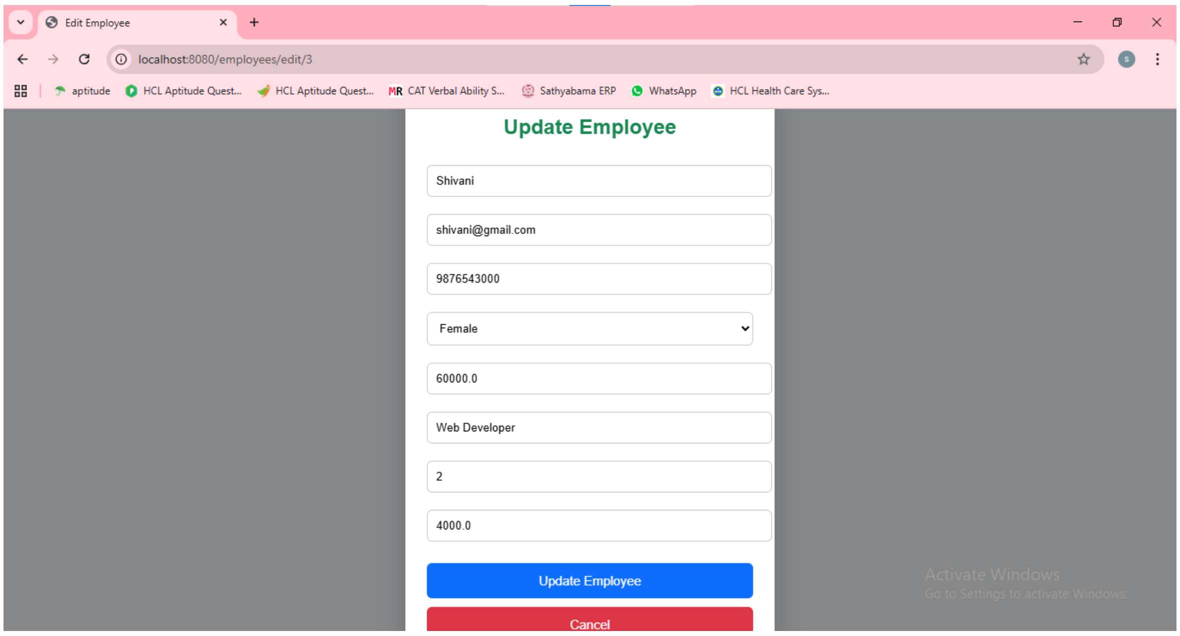
Added Employee



Employee Management System (ADMIN)

ID	Name	Email	Phone	Gender	Salary	Role	Experience	Bonus	Actions
1	Ravi Kumar	ravi@gmail.com	9876543210	Male	50000.0	Java Developer	3	5000.0	Edit Delete
2	Anu Priya	anu@gmail.com	9876501234	Female	55000.0	Web Developer	2	3000.0	Edit Delete
3	Shivani	shivani@gmail.com	9876543000	Female	60000.0	Web Developer	2	4000.0	Edit Delete
4	Rashmi	rashmi142@gmail.com	9876345261	Female	50000.0	UI developer	1	500.0	Edit Delete
5	Siva	sivak123@gmail.com	9123456780	Male	80000.0	Android developer	3	1000.0	Edit Delete
7	Lakshitha	lakshu1@gmail.com	87625361230	Female	80000.0	Web developer	4	2000.0	Edit Delete
8	Sharu	sharuu123@gmail.com	9006754317	Female	85000.0	Python developer	2	1000.0	Edit Delete
9	Adhitya	adhi65@gmail.com	9452178654	Male	50000.0	UI developer	1	500.0	Edit Delete
13	Kaushik	kaushik@gmail.com	93421165778	Male	40000.0	Android developer	2	1500.0	Edit Delete

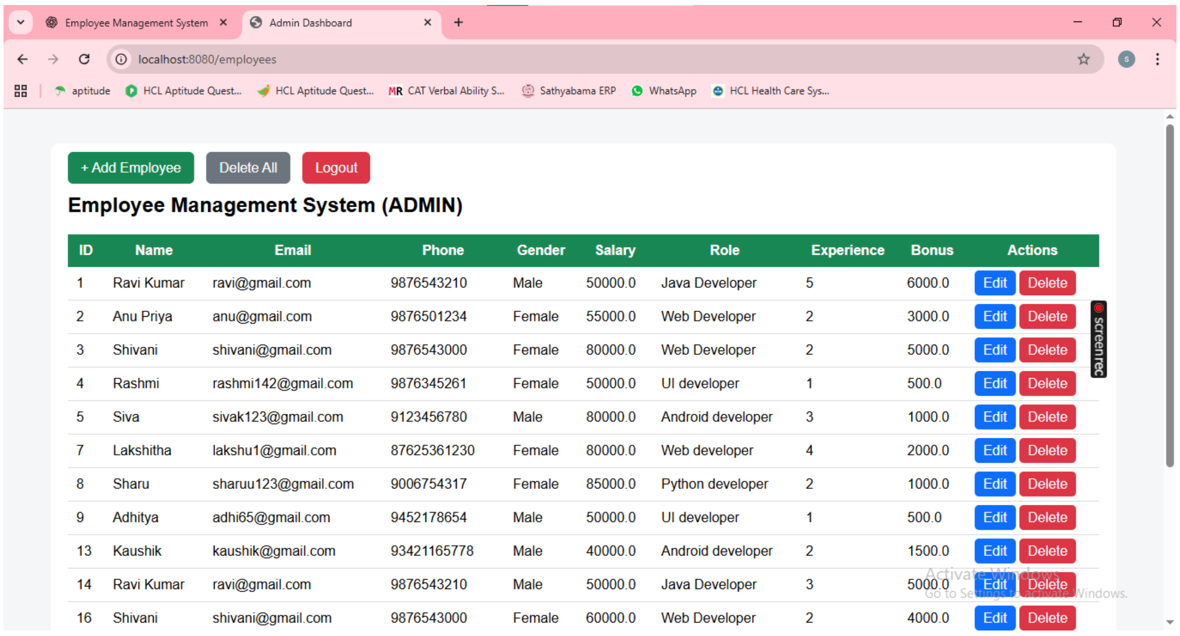
Update / Edit Employee



Update Employee

Shivani
shivani@gmail.com
9876543000
Female
60000.0
Web Developer
2
4000.0

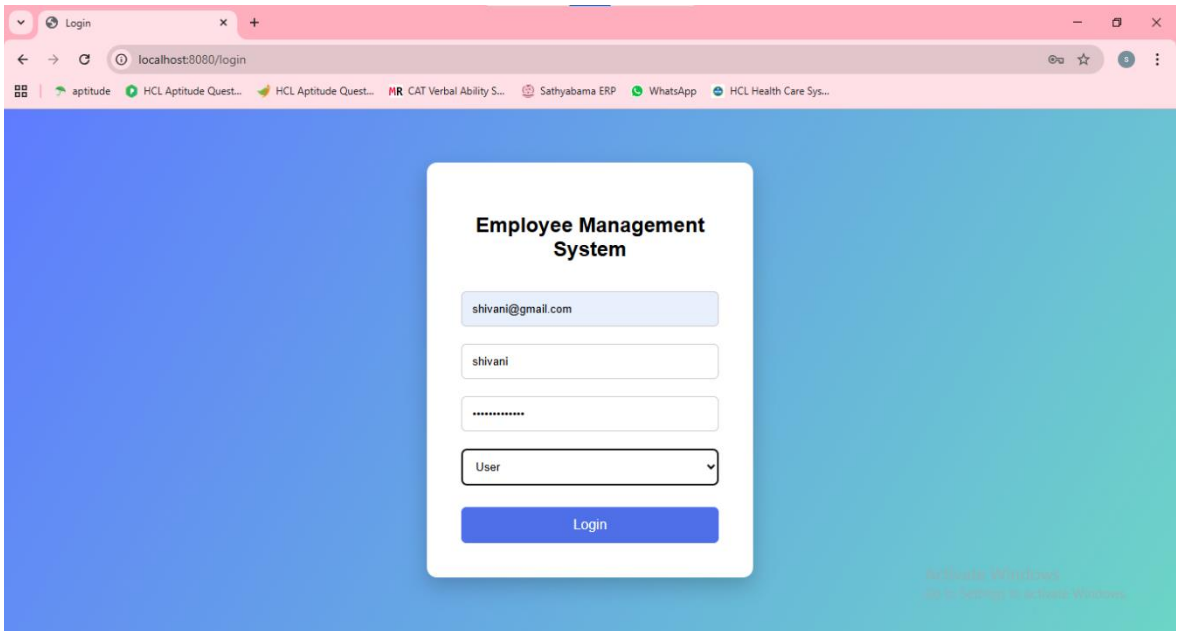
After Updating Employee



Employee Management System (ADMIN)

ID	Name	Email	Phone	Gender	Salary	Role	Experience	Bonus	Actions
1	Ravi Kumar	ravi@gmail.com	9876543210	Male	50000.0	Java Developer	5	6000.0	Edit Delete
2	Anu Priya	anu@gmail.com	9876501234	Female	55000.0	Web Developer	2	3000.0	Edit Delete
3	Shivani	shivani@gmail.com	9876543000	Female	80000.0	Web Developer	2	5000.0	Edit Delete
4	Rashmi	rashmi142@gmail.com	9876345261	Female	50000.0	UI developer	1	500.0	Edit Delete
5	Siva	sivak123@gmail.com	9123456780	Male	80000.0	Android developer	3	1000.0	Edit Delete
7	Lakshitha	lakshu1@gmail.com	87625361230	Female	80000.0	Web developer	4	2000.0	Edit Delete
8	Sharu	sharuu123@gmail.com	9006754317	Female	85000.0	Python developer	2	1000.0	Edit Delete
9	Adhitya	adhi65@gmail.com	9452178654	Male	50000.0	UI developer	1	500.0	Edit Delete
13	Kaushik	kaushik@gmail.com	93421165778	Male	40000.0	Android developer	2	1500.0	Edit Delete
14	Ravi Kumar	ravi@gmail.com	9876543210	Male	50000.0	Java Developer	3	5000.0	Edit Delete
16	Shivani	shivani@gmail.com	9876543000	Female	60000.0	Web Developer	2	4000.0	Edit Delete

User Login



Employee Management System

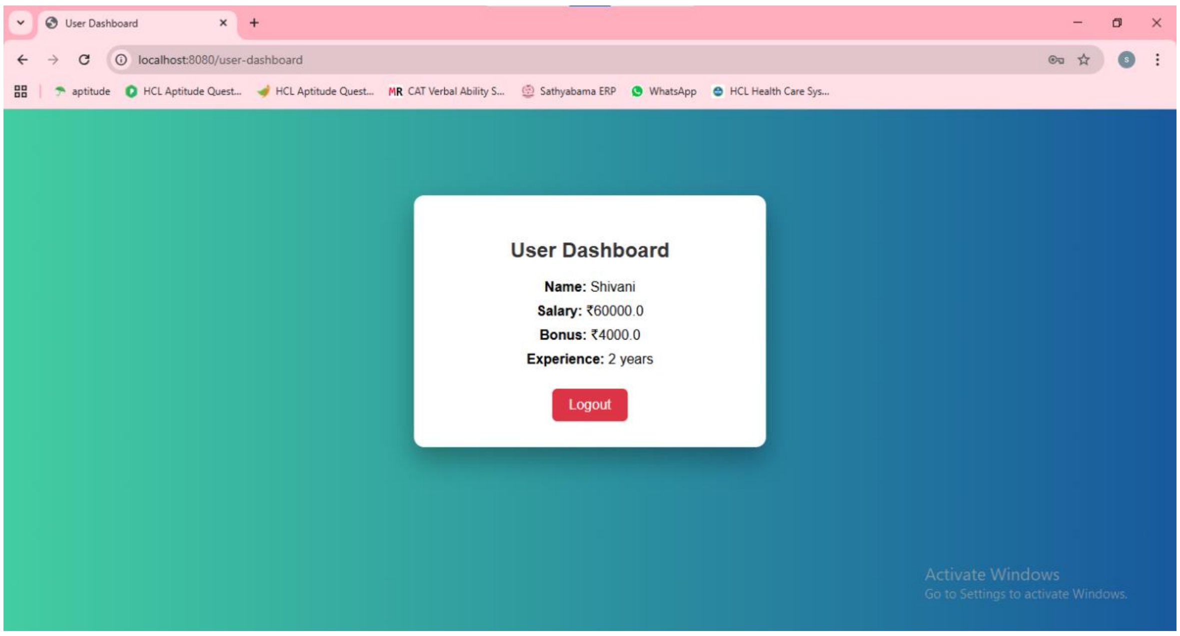
shivani@gmail.com

shivani

•••••••••••••

User

User Dashboard 1



User Dashboard

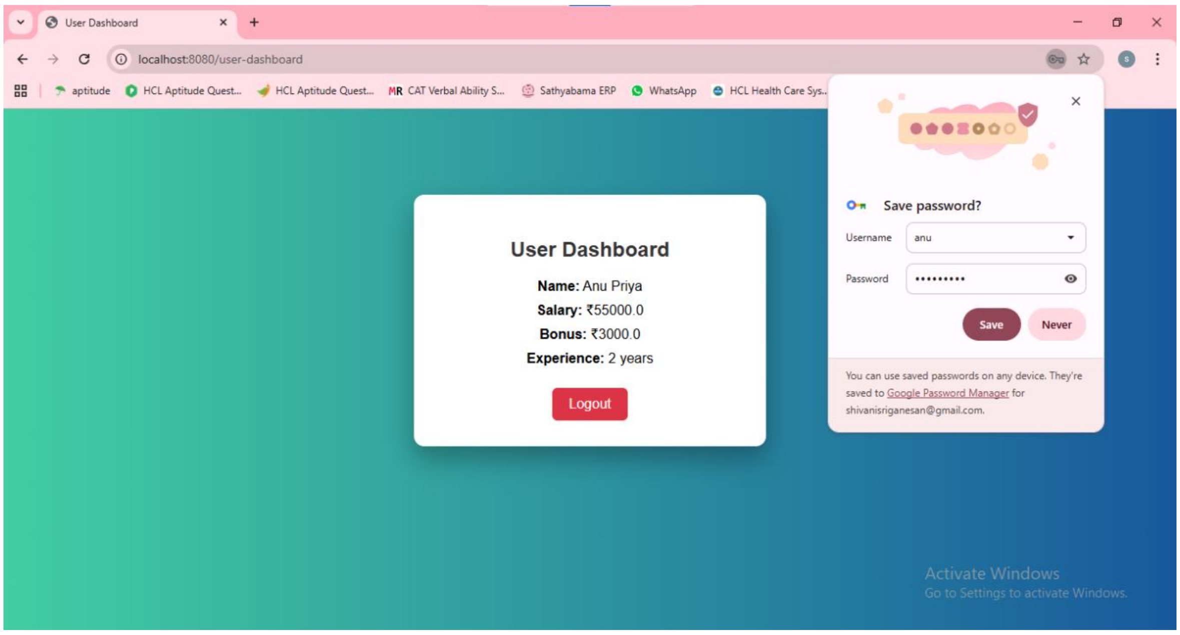
Name: Shivani

Salary: ₹60000.0

Bonus: ₹4000.0

Experience: 2 years

User Dashboard 2



User Dashboard

Name: Anu Priya

Salary: ₹55000.0

Bonus: ₹3000.0

Experience: 2 years

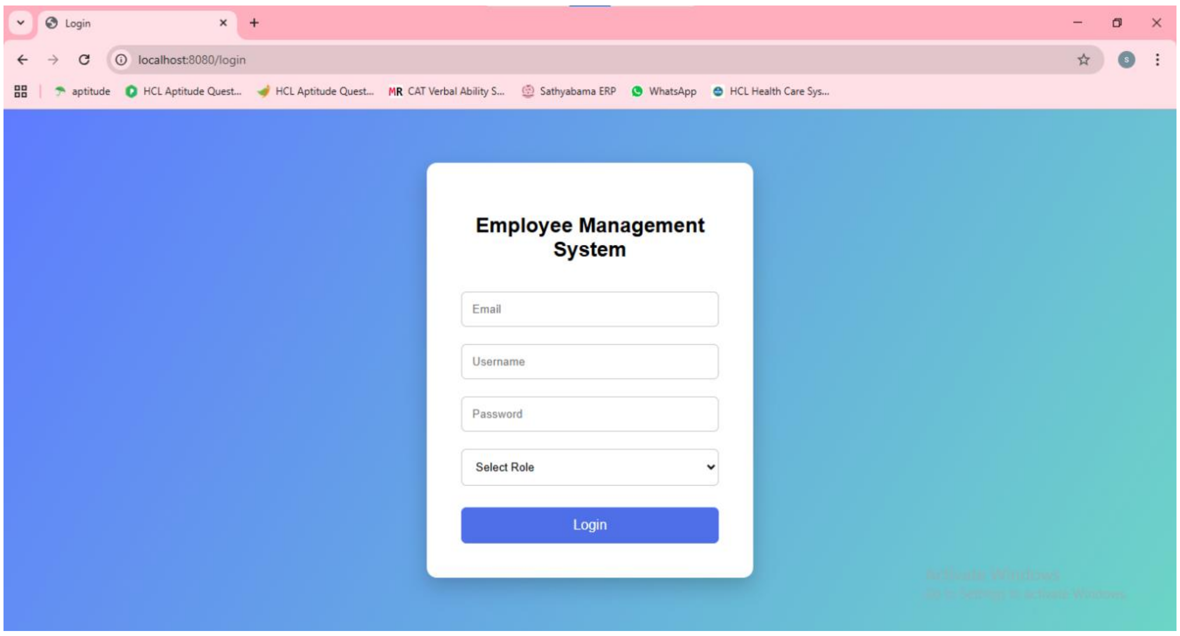
Save password?

Username: anu

Password: •••••••••

You can use saved passwords on any device. They're saved to Google Password Manager for shivanisriganesan@gmail.com.

Logged Out



Employee Management System

Email

Username

Password

Select Role

Login