Project Document

Project Overview

This Employee Management System Project application stores all the employee's information in a database. It is an application developed in Java GUI technology and database used is SQLite. It contains employee information like employee id, first name, surname, and age. It is an easy to use application and has a user-friendly interface. It is totally built at the administrative end which means that only the admin has access rights to change or modify any records. So this makes it safe and reliable application to use. The main aim of developing this application was to reduce the errors that occur in the manual system. One can search the details easily by just entering employee id. In earlier systems, there was not such a facility to do so. All the details are stored in an SQLite database. It is easy to update any employee details. All the employee records are integrated and so this makes it user-friendly and easy to use application.

Explain project in short

Employee Management System Modules

- Signup module**:** This module is used for the new employee where he or she can signup themselves and get their login details.
- Login module**:** This module is created for login details of the employee. There are different login details provided and both can login using their respective user id and password.
- Manage employee module**:** This module enables user to add and edit employee records. It is also for viewing the each and every employee details by the admin.
- Delete module**:** This module is created to delete any employee details.

Technical Specification

Here you can put what tech stack you used to create this
project and why. eg.

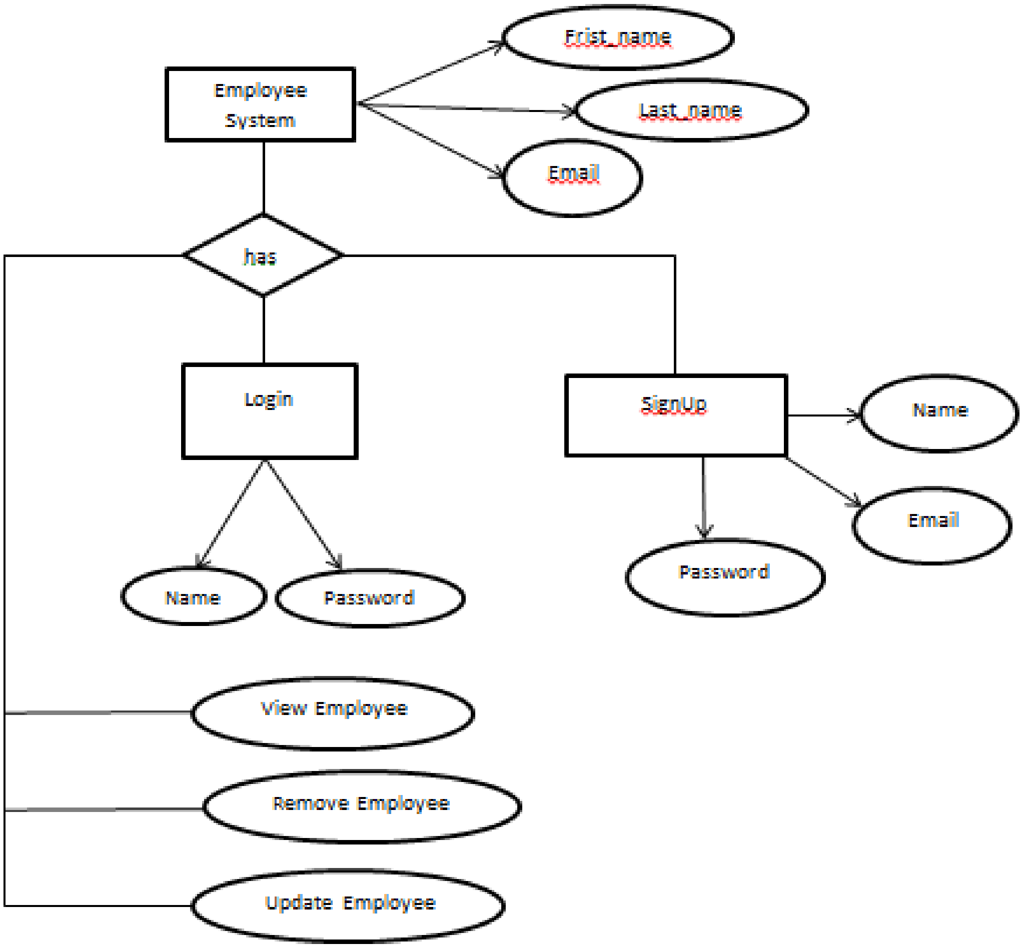
UI	ReactBootstrap &
Backend	Java
Database	MySql
Framework Used	SpringBoot
Plugins Used	MySQLWorkbench

What are the endpoints(services) created for data manipulation. eg.

Add/Edit	actions/add,Edit,remove,and View
List Page	service/get and post
Sign Up/Login	action/sig nup action/log in

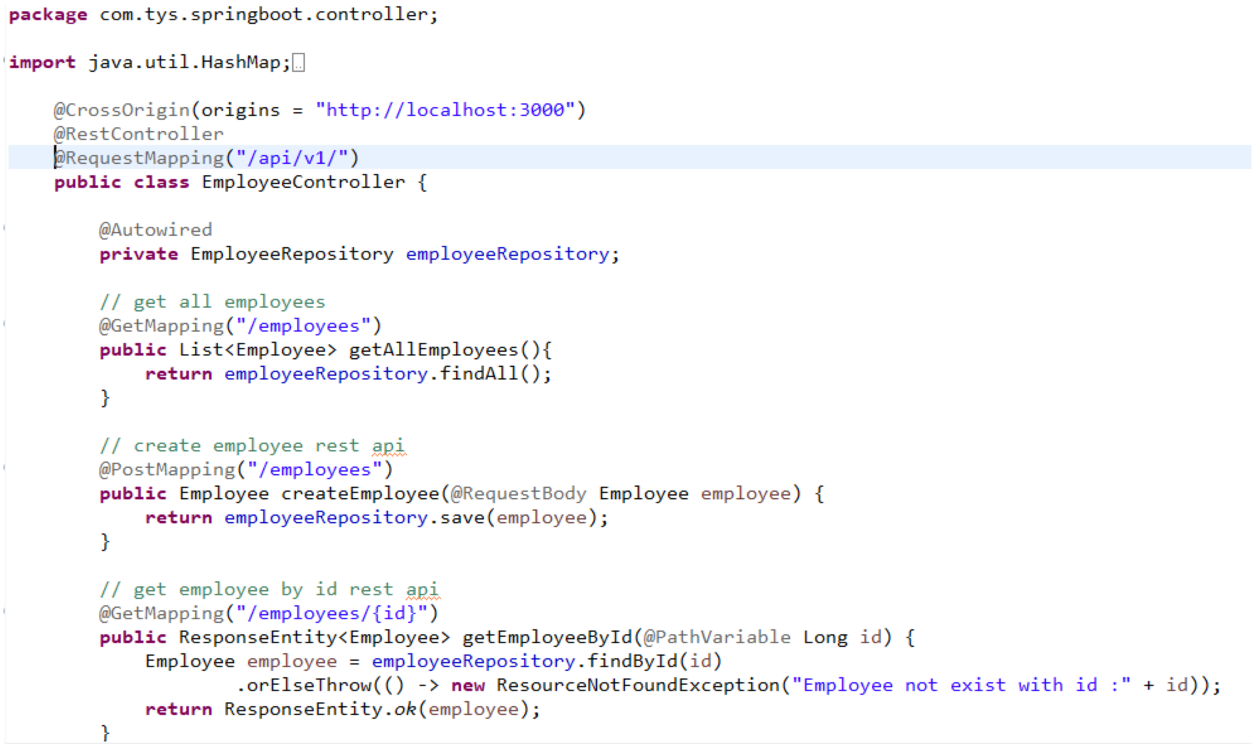
Workflow Diagram

Please create a basic workflow diagram of project. You can user draw.io to create this diagram. eg.

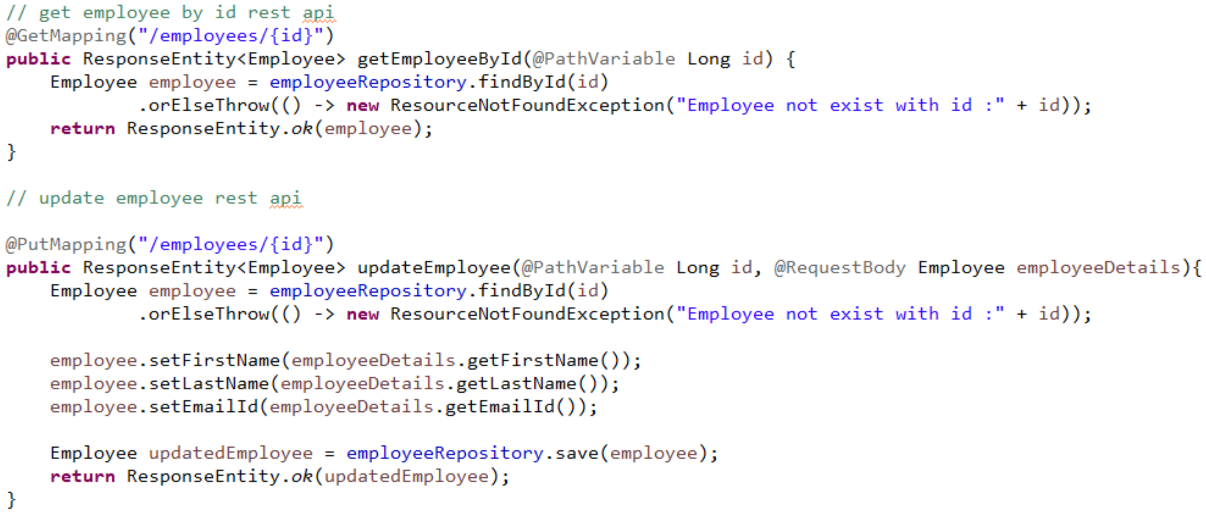


Screenshots

Please put screenshots of pages and some code snippets



```java
package com.tys.springboot.controller;

import java.util.HashMap;

    @CrossOrigin(origins = "http://localhost:3000")
    @RestController
    @RequestMapping("/api/v1/")
    public class EmployeeController {

        @Autowired
        private EmployeeRepository employeeRepository;

        // get all employees
        @GetMapping("/employees")
        public List<Employee> getAllEmployees(){
            return employeeRepository.findAll();
        }

        // create employee rest api
        @PostMapping("/employees")
        public Employee createEmployee(@RequestBody Employee employee) {
            return employeeRepository.save(employee);
        }

        // get employee by id rest api
        @GetMapping("/employees/{id}")
        public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {
            Employee employee = employeeRepository.findById(id)
                    .orElseThrow(() -> new ResourceNotFoundException("Employee not exist with id :" + id));
            return ResponseEntity.ok(employee);
        }
```



```java
// get employee by id rest api
@GetMapping("/employees/{id}")
public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {
    Employee employee = employeeRepository.findById(id)
            .orElseThrow(() -> new ResourceNotFoundException("Employee not exist with id :" + id));
    return ResponseEntity.ok(employee);
}

// update employee rest api

@PutMapping("/employees/{id}")
public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee employeeDetails){
    Employee employee = employeeRepository.findById(id)
            .orElseThrow(() -> new ResourceNotFoundException("Employee not exist with id :" + id));

    employee.setFirstName(employeeDetails.getFirstName());
    employee.setLastName(employeeDetails.getLastName());
    employee.setEmailId(employeeDetails.getEmailId());

    Employee updatedEmployee = employeeRepository.save(employee);
    return ResponseEntity.ok(updatedEmployee);
}
```