ProManageHub By TalentTech Innovations

Project By:

1. Anjali Kumar More

anjalikmore.18@gmail.com

2. Rahul Bamane

rahulbamane09@gmail.com

3. Sakshi Naravankar

sakxnaravankar@gmail.com

ABSTRACT

In this world of growing technologies everything has been computerized. With large number of work opportunities the Human workforce has increased. Thus there is a need of a system which can handle the data of such a large number of Employees in an organization. This project simplifies the task of maintain records because of its user friendly nature.

The "**ProManageHub**" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system.

ACKNOWLEDGEMENT

We are very glad to present this project "**ProManageHub**". We take this opportunity to express our gratitude to many people , whose good wishes and positive approach have inspired us to take this project as success. However , it is imperative for us to mention some of them who played significant role in our project.

DECLARATION

We here by declaring that the project entitled “Employee Management System” has not been any case duplicated to submit to any other university for the award of any degree. To the best of our knowledge other than us, no one has submitted to any other university.

TABLE OF CONTENT

INTRODUCTION

Every organization, whether big or small has human resource challenges to overcome, every organization has different employee management needs, therefore we design exclusive employee management systems that are adapted to your managerial requirements. This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of human resources for your future goals. Also, for those buzy executive who are always on to go, our systems come with remote access features, which will allow you to manage your workforce anytime, at all times. These systems will ultimately allow you to better manage resources. One of the main features in employee management system is time tracking for employees. Effective time tracking mechanism saves both time and money for organization.

This project is used to develop CRUD RESTFul APIs for a simple Employee Management System using Spring Boot , JPA and MySQL as a database.

We will implement the following features in this project:

- List Employee Feature
- Add Employee Feature
- Update Employee Feature
- Delete Employee Feature
- Sorting Feature
- Login Feature
- Registration Feature

FEATURES AND FUNCTIONALITY OF THE EMPLOYEE MANAGEMENT SYSTEM

Employee Module:

- Admin can add new employee records
- Admin can see the list of employee details
- Only admin can edit and update the record of the employee
- Admin will be able to delete the records of the employee

Leave Module:

- Admin can manage the leave
- Admin can edit/delete the leave
- Admin can see the list of all leave
- Employee can see his leave

Salary Module:

- **Admin** can manage the salary
- Admin can edit/delete the salary
- Admin can see the list of all salary
- Employee can see salary

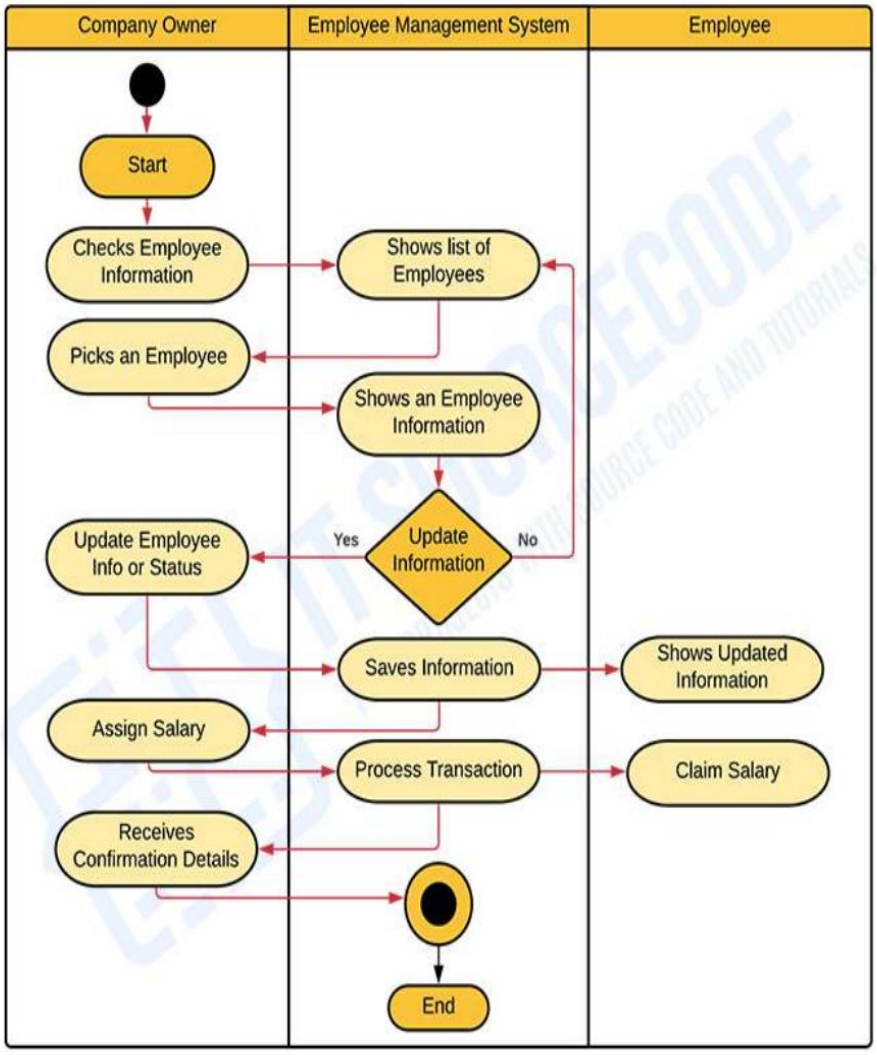
Tools and technologies used:

- Java 17
- Spring Boot
- Spring Data JPA (Hibernate)
- Spring Security
- MySQL
- Eclipse STS
- Maven
- Tomcat
- Thymeleaf

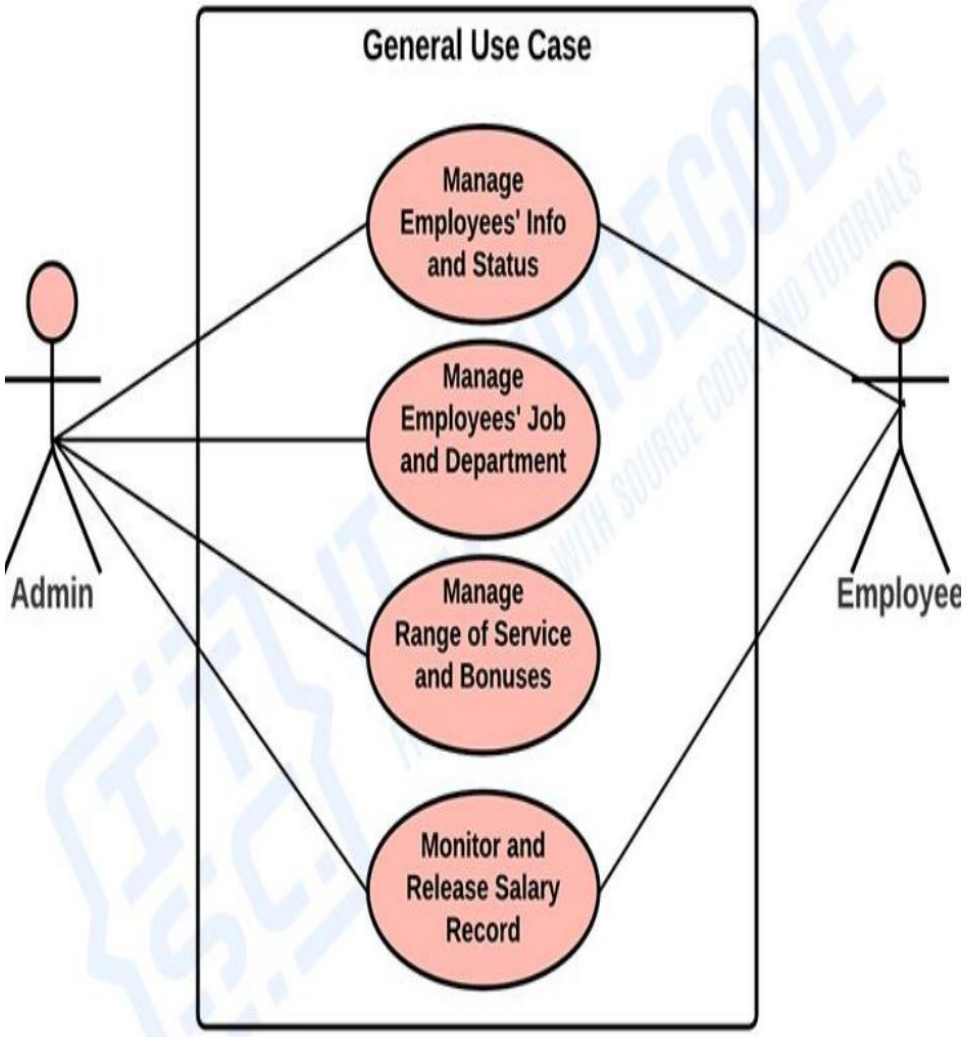
Front-End: CSS, HTML

DIAGRAMS FOR EMPLOYEE MANAGEMENT SYSTEM PROJECT

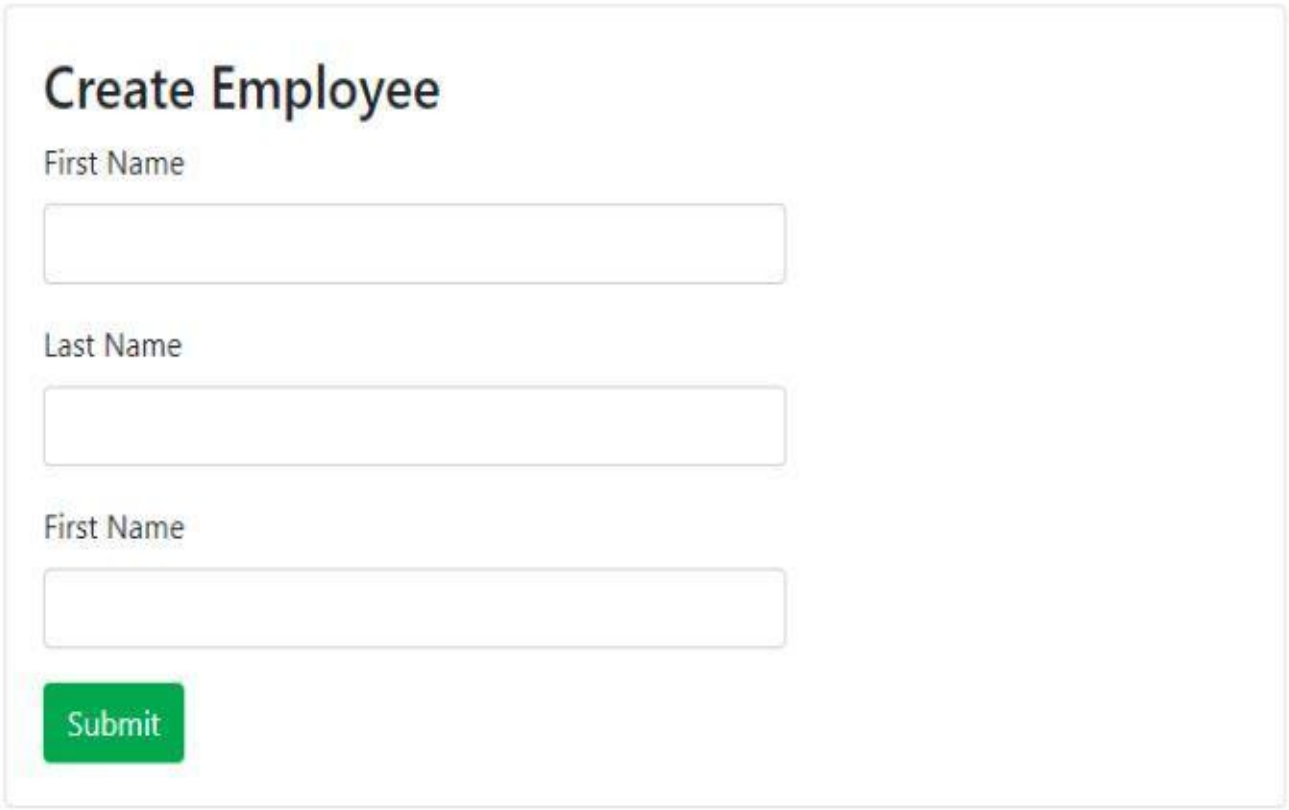
Activity Diagram:

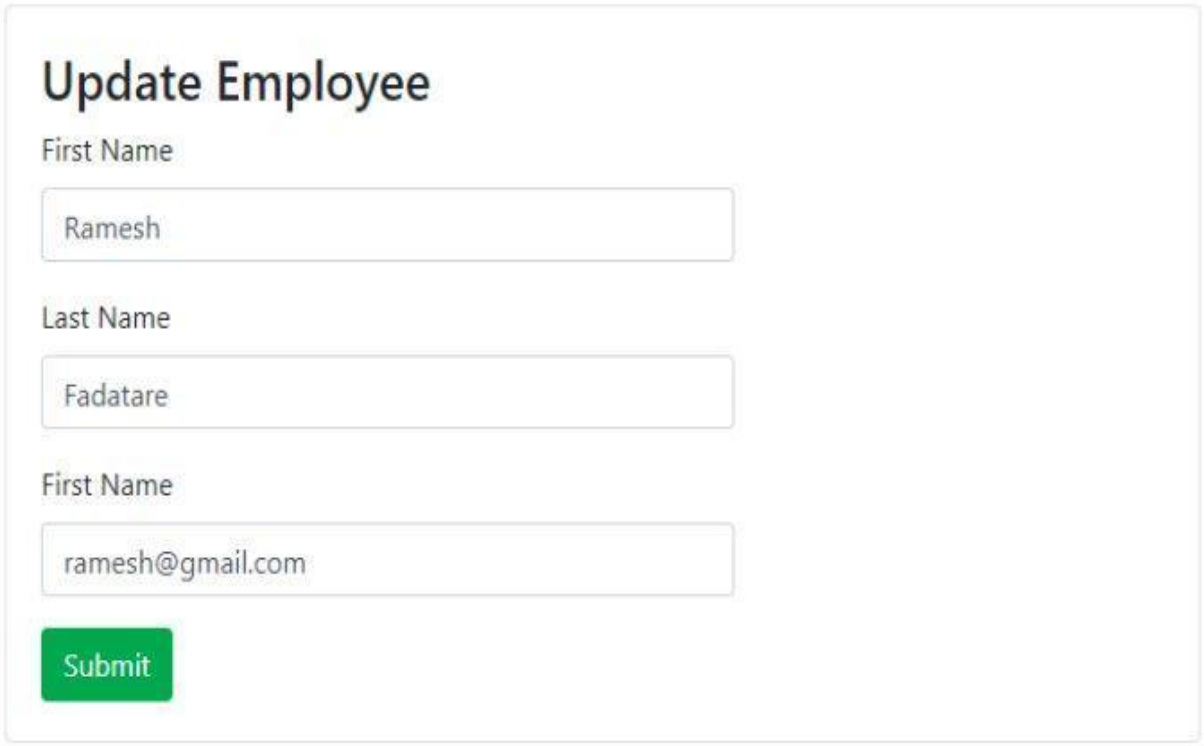


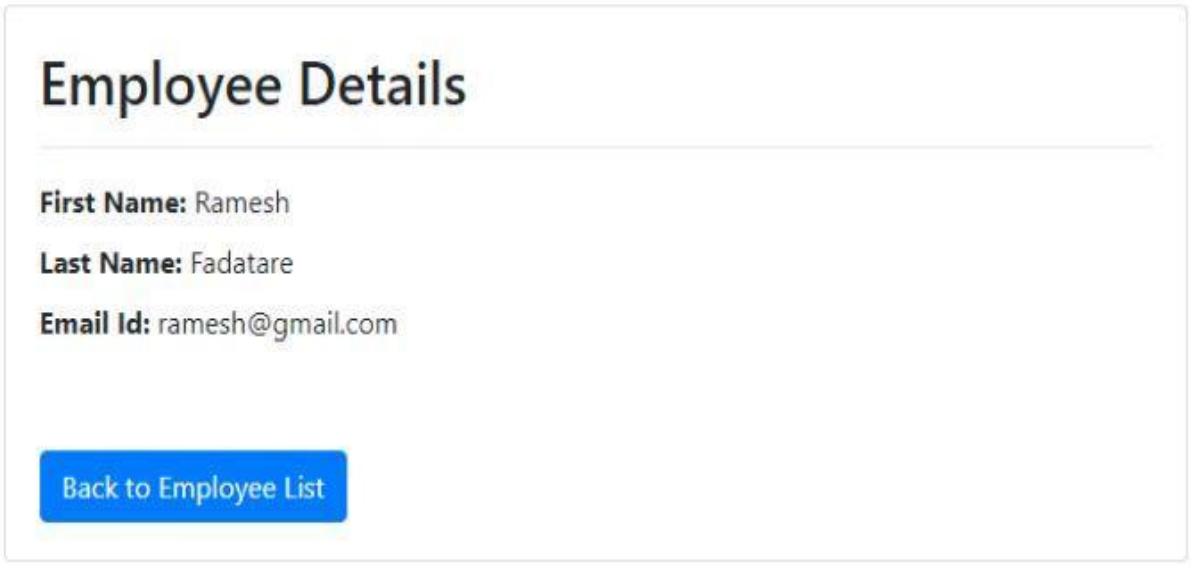
Use Case Diagram:



DESIGN AND IMPLEMENTATION OF EMPLOYEE MANAGEMENT SYSTEM




Update Employee
First Name
Ramesh
Last Name
Fadatare
First Name
ramesh@gmail.com
Submit


Employee Details
First Name: Ramesh
Last Name: Fadatare
Email Id: ramesh@gmail.com
Back to Employee List

Employee List

Firstname	Lastname	Email	Actions
Ramesh	Fadatare	ramesh@gmail.com	Delete Update Details
John	Cena	john@gmail.com	Delete Update Details
Tom	Cruise	tom@gmail.com	Delete Update Details
Tony	Stark	tony@gmail.co	Delete Update Details

SOURCE CODE:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.3.0.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>net.javaguides</groupId>
	<artifactId>employee-management-webapp</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>employee-management-webapp</name>
	<description>Demo project for Spring Boot Thymeleaf and Hibernate </description>

	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-security</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-thymeleaf</artifactId>
		</dependency>

		<dependency>
			<groupId>org.thymeleaf.extras</groupId>
```

```
            <artifactId>thymeleaf-extras-springsecurity5</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
```

```
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                    </plugin>
                </plugins>
        </build>
</project>

# DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)

spring.datasource.url=jdbc:mysql://localhost:3306/demo?useSSL=false&serverTimezone=UTC
&useLegacyDatetimeCode=false
spring.datasource.username=root
spring.datasource.password=root

# Hibernate

# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect

# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto = update

logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type=TRACE


package net.javaguides.springboot.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "employees")
public class Employee {

        @Id
        @GeneratedValue(strategy =  GenerationType.IDENTITY)
```

```
	private long id;

	@Column(name = "first_name")
	private String firstName;

	@Column(name = "last_name")
	private String lastName;

	@Column(name = "email")
	private String email;
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
}

package net.javaguides.springboot.model;


import java.util.Collection;
```

```
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;
import javax.persistence.JoinColumn;

@Entity
@Table(name =  "user", uniqueConstraints = @UniqueConstraint(columnNames = "email"))
public class User {

	@Id
	@GeneratedValue(strategy =  GenerationType.IDENTITY)
	private Long id;

	@Column(name = "first_name")
	private String firstName;
```

```
    @Column(name = "last_name")
    private String lastName;

    private String email;

    private String password;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinTable(
                    name = "users_roles",
                    joinColumns = @JoinColumn(
                    name = "user_id", referencedColumnName = "id"),
                    inverseJoinColumns = @JoinColumn(
                                    name = "role_id", referencedColumnName = "id"))

    private Collection<Role> roles;

    public User() {

    }

    public User(String firstName, String lastName, String email, String password,
Collection<Role> roles) {
            super();
            this.firstName = firstName;
```

```
        this.lastName = lastName;
        this.email = email;
        this.password = password;
        this.roles = roles;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getEmail() {
        return email;
```

```
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public Collection<Role> getRoles() {
		return roles;
	}
	public void setRoles(Collection<Role> roles) {
		this.roles = roles;
	}
}
package net.javaguides.springboot.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
```

```
@Entity
@Table(name = "role")
public class Role {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	private String name;

	public Role() {

	}
	public Role(String name) {
		super();
		this.name = name;
	}

	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getName() {
		return name;
```

```
	}
	public void setName(String name) {
		this.name = name;
	}
}
```

```
package net.javaguides.springboot.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import net.javaguides.springboot.model.Employee;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Long>{

}
```

```
package net.javaguides.springboot.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import net.javaguides.springboot.model.User;

@Repository
```

```
public interface UserRepository extends JpaRepository<User, Long>{
	User findByEmail(String email);
}

package net.javaguides.springboot.service;

import java.util.List;

import org.springframework.data.domain.Page;

import net.javaguides.springboot.model.Employee;

public interface EmployeeService {
	List<Employee> getAllEmployees();
	void saveEmployee(Employee employee);
	Employee getEmployeeById(long id);
	void deleteEmployeeById(long id);
	Page<Employee> findPaginated(int pageNo, int pageSize, String sortField, String sortDirection);
}

package net.javaguides.springboot.service;

import java.util.List;
import java.util.Optional;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

import net.javaguides.springboot.model.Employee;
import net.javaguides.springboot.repository.EmployeeRepository;

@Service
public class EmployeeServiceImpl implements EmployeeService {

	@Autowired
	private EmployeeRepository employeeRepository;

	@Override
	public List<Employee> getAllEmployees() {
		return employeeRepository.findAll();
	}

	@Override
	public void saveEmployee(Employee employee) {
		this.employeeRepository.save(employee);
	}
```

```
	@Override
	public Employee getEmployeeById(long id) {
		Optional<Employee> optional = employeeRepository.findById(id);
		Employee employee = null;
		if (optional.isPresent()) {
			employee = optional.get();
		} else {
			throw new RuntimeException(" Employee not found for id :: " + id);
		}
		return employee;
	}

	@Override
	public void deleteEmployeeById(long id) {
		this.employeeRepository.deleteById(id);
	}

	@Override
	public Page<Employee> findPaginated(int pageNo, int pageSize, String sortField, String
sortDirection) {
		Sort sort = sortDirection.equalsIgnoreCase(Sort.Direction.ASC.name()) ?
Sort.by(sortField).ascending() :
			Sort.by(sortField).descending();

		Pageable pageable = PageRequest.of(pageNo - 1, pageSize, sort);
```

```
                return this.employeeRepository.findAll(pageable);
        }
}
package net.javaguides.springboot.service;

import org.springframework.security.core.userdetails.UserDetailsService;

import net.javaguides.springboot.dto.UserRegistrationDto;
import net.javaguides.springboot.model.User;

public interface UserService extends UserDetailsService{
        User save(UserRegistrationDto registrationDto);
}

package net.javaguides.springboot.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
```

```
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import net.javaguides.springboot.model.Employee;
import net.javaguides.springboot.service.EmployeeService;

@Controller
public class EmployeeController {

	@Autowired
	private EmployeeService employeeService;

	// display list of employees
	@GetMapping("/")
	public String viewHomePage(Model model) {
		return findPaginated(1, "firstName", "asc", model);
	}

	@GetMapping("/showNewEmployeeForm")
	public String showNewEmployeeForm(Model model) {
		// create model attribute to bind form data
		Employee employee = new Employee();
		model.addAttribute("employee", employee);
		return "new_employee";
	}
```

```
	@PostMapping("/saveEmployee")
	public String saveEmployee(@ModelAttribute("employee") Employee employee) {
		// save employee to database
		employeeService.saveEmployee(employee);
		return "redirect:/";
	}

	@GetMapping("/showFormForUpdate/{id}")
	public String showFormForUpdate(@PathVariable ( value = "id") long id, Model model)
{

		// get employee from the service
		Employee employee = employeeService.getEmployeeById(id);

		// set employee as a model attribute to pre-populate the form
		model.addAttribute("employee", employee);
		return "update_employee";
	}

	@GetMapping("/deleteEmployee/{id}")
	public String deleteEmployee(@PathVariable (value = "id") long id) {

		// call delete employee method
		this.employeeService.deleteEmployeeById(id);
```

```
        return "redirect:/";
    }


    @GetMapping("/page/{pageNo}")
    public String findPaginated(@PathVariable (value = "pageNo") int pageNo,
                                @RequestParam("sortField") String sortField,
                                @RequestParam("sortDir") String sortDir,
                                Model model) {
        int pageSize = 5;

        Page<Employee> page = employeeService.findPaginated(pageNo, pageSize,
sortField, sortDir);
        List<Employee> listEmployees = page.getContent();

        model.addAttribute("currentPage", pageNo);
        model.addAttribute("totalPages", page.getTotalPages());
        model.addAttribute("totalItems", page.getTotalElements());

        model.addAttribute("sortField", sortField);
        model.addAttribute("sortDir", sortDir);
        model.addAttribute("reverseSortDir", sortDir.equals("asc") ? "desc" : "asc");

        model.addAttribute("listEmployees", listEmployees);
        return "index";
```

```
	}
}

package net.javaguides.springboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class MainController {

	@GetMapping("/login")
	public String login() {
		return "login";
	}
	/*
	 * @GetMapping("/") public String home() { return "index"; }
	 */
}

package net.javaguides.springboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
```

```
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import net.javaguides.springboot.dto.UserRegistrationDto;
import net.javaguides.springboot.service.UserService;

@Controller
@RequestMapping("/registration")
public class UserRegistrationController {

	private UserService userService;

	public UserRegistrationController(UserService userService) {
		super();
		this.userService = userService;
	}

	@ModelAttribute("user")
   public UserRegistrationDto userRegistrationDto() {
      return new UserRegistrationDto();
   }

	@GetMapping
	public String showRegistrationForm() {
		return "registration";
```

```
	}


	@PostMapping
	public String registerUserAccount(@ModelAttribute("user") UserRegistrationDto
registrationDto) {
		userService.save(registrationDto);
		return "redirect:/registration?success";
	}
}


<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Employee Management System</title>

<link rel="stylesheet"
	href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
	integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
	crossorigin="anonymous">

</head>
<body>

<!-- create navigation bar ( header) -->
```

```
<nav class="navbar navbar-inverse navbar-fixed-top">
	<div class="container">
		<div class="navbar-header">
			<button type="button" class="navbar-toggle collapsed"
				data-toggle="collapse" data-target="#navbar" aria-expanded="false"
				aria-controls="navbar">
				<span class="sr-only">Toggle navigation</span> <span
					class="icon-bar"></span> <span class="icon-bar"></span> <span
					class="icon-bar"></span>
			</button>
			<a class="navbar-brand" href="#" th:href="@{/}">Employee Management System</a>
		</div>
		<div id="navbar" class="collapse navbar-collapse">
			<ul class="nav navbar-nav">
				<li sec:authorize="isAuthenticated()"><a th:href="@{/logout}">Logout</a></li>
			</ul>
		</div>
	</div>
</nav>

<br>
<br>
```

```
<div class="container my-2">

<h1>Employees List</h1>


<a th:href = "@{/showNewEmployeeForm}" class="btn btn-primary btn-sm mb-3"> Add Employee </a>


<table border="1" class = "table table-striped table-responsive-md">

<thead>

<tr>

<th>

<a th:href="@{'/page/' + ${currentPage} + '?sortField=firstName&sortDir=' + ${reverseSortDir}}">

Employee First Name</a>

</th>

<th>

<a th:href="@{'/page/' + ${currentPage} + '?sortField=lastName&sortDir=' + ${reverseSortDir}}">

Employee Last Name</a>

</th>

<th>

<a th:href="@{'/page/' + ${currentPage} + '?sortField=email&sortDir=' + ${reverseSortDir}}">

Employee Email</a>

</th>

<th> Actions </th>

</tr>

</thead>
```

```
                <tbody>
                    <tr th:each="employee : ${listEmployees}">
                        <td th:text="${employee.firstName}"></td>
                        <td th:text="${employee.lastName}"></td>
                        <td th:text="${employee.email}"></td>
                        <td> <a
th:href="@{/showFormForUpdate/{id}(id=${employee.id})}" class="btn btn-
primary">Update</a>
                            <a
th:href="@{/deleteEmployee/{id}(id=${employee.id})}" class="btn btn-danger">Delete</a>
                        </td>
                    </tr>
                </tbody>
            </table>

            <div th:if = "${totalPages > 1}">
                <div class = "row col-sm-10">
                    <div class = "col-sm-2">
                        Total Rows: [[${totalItems}]]
                    </div>
                    <div class = "col-sm-1">
                        <span th:each="i: ${#numbers.sequence(1, totalPages)}">
                            <a th:if="${currentPage != i}" th:href="@{'/page/'
+ ${i}+ '?sortField=' + ${sortField} + '&sortDir=' + ${sortDir}}">[[${i}]]</a>
                            <span th:unless="${currentPage !=
i}">[[${i}]]</span>     
                        </span>
```

```
                    </div>
                    <div class = "col-sm-1">
                        <a th:if="${currentPage < totalPages}" th:href="@{'/page/'
+ ${currentPage + 1}+ '?sortField=' + ${sortField} + '&sortDir=' + ${sortDir}}">Next</a>
                        <span th:unless="${currentPage <
totalPages}">Next</span>
                    </div>

                    <div class="col-sm-1">
                    <a th:if="${currentPage < totalPages}" th:href="@{'/page/' +
${totalPages}+ '?sortField=' + ${sortField} + '&sortDir=' + ${sortDir}}">Last</a>
                        <span th:unless="${currentPage <
totalPages}">Last</span>
                    </div>
                </div>
            </div>
        </div>
</body>
</html>
```

CONCLUSION

The project is to digitalize the database of Employees in Organizations and enabling Administrators to have benefit from Computers. Software acts as a Information System between Employees and administrators. Here the user can keep his/her database secure and safe for a unlimited period of time.

Software provides Employee management system for inserting, updating, searching and deleting records with ease and simplicity. We will provide afresh new approach to our esteemed users to search for records and make databases in a digital way.