Full-Stack Web Application Project with Code

1. Project Overview

This is a full-stack employee management web application with login, employee addition, search, edit, delete, and file upload functionalities. Developed using Java Spring Boot (backend), MySQL (database), and HTML/CSS/Bootstrap with JavaScript (frontend).

2. Tech Stack

- Backend: Java Spring Boot
- Frontend: HTML, CSS (Bootstrap), JavaScript
- Database: MySQL
- File Uploads: Spring Multipart
- Pagination: Server-side

3. Folder Structure

```
/employeeapp
  controller/
      AuthController.java
      EmployeeController.java
  model/
      Employee.java
  repository/
      EmployeeRepository.java
  service/
      EmployeeService.java
  resources/templates/
      login.html
      add_employee.html
      search_employee.html
  resources/application.properties
  EmployeeAppApplication.java
```

4. Backend Code (Spring Boot)

Employee.java (Model):

```
@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @Column(unique = true)
    private String employeeId;
    private String firstName, lastName, middleName, loginId;
    private LocalDate dob;
    private String department, salary;
    private String permAddress, currAddress;
    private String idProofPath;
}
```

EmployeeRepository.java:

```
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
    boolean existsByLoginId(String loginId);
    List<Employee> findByDepartment(String dept);
}
```

EmployeeService.java (partial):

```
public String generateLoginId(String fname, String lname) {
    String loginId = fname.charAt(0) + lname;
    while (repository.existsByLoginId(loginId)) {
        loginId = fname.charAt(0) + lname + (int)(Math.random() * 1000);
    }
    return loginId;
}
```

EmployeeController.java (partial):

```
@PostMapping("/add")
public String addEmployee(@ModelAttribute Employee emp, @RequestParam("file") MultipartFile file)
{
        if (!file.getContentType().equals("application/pdf") || file.getSize() > 1000000 ||
file.getSize() < 10000)
        throw new IllegalArgumentException("Invalid file");
    String path = saveFile(file);
    emp.setIdProofPath(path);
    emp.setLoginId(service.generateLoginId(emp.getFirstName(), emp.getLastName()));
    repository.save(emp);
    return "redirect:/search";
}
```

5. Frontend (HTML/Bootstrap)

login.html:

```
<form method="POST" action="/login">
```

```
  <input type="text" name="username" required placeholder="Username"/>
  <input type="password" name="password" required placeholder="Password"/>
  <button type="submit">Login</button>
</form>
```

add_employee.html (partial):

```
<form method="POST" enctype="multipart/form-data" action="/add">
  <input name="firstName" required />
  <input name="lastName" required />
  <input type="date" name="dob" required />
  <select name="department">
    <option>Engineering</option><option>Support</option>
    <option>HR</option><option>Finance</option>
  </select>
  <input type="file" name="file" accept="application/pdf" />
  <button type="submit">Add</button>
</form>
```

6. Database Schema

```
CREATE TABLE employee (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    employee_id VARCHAR(20) UNIQUE,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    middle_name VARCHAR(100),
    login_id VARCHAR(100) UNIQUE,
    dob DATE,
    department VARCHAR(50),
    salary VARCHAR(20),
    perm_address TEXT,
    curr_address TEXT,
    id_proof_path TEXT
);
```

7. Run Instructions

1. Configure database in application.properties
2. Run using Spring Boot CLI or IDE
3. Access endpoints: /login, /add, /search
4. Bootstrap CDN and JS for UI