**Employee Management System**

**Table of Contents**

1. **Introduction**
2. **Project Objective**
3. **Tools and Technologies**
4. **Background of the Project**
5. **Functional Requirements**
   * **Admin User Capabilities**
   * **Employee Capabilities**
6. **Back-End Development**
7. **Front-End Development**
8. **Unit Testing**
9. **Conclusion**

### **Introduction:**

The Employee Management System is a comprehensive solution designed to streamline and automate the management of employee data and leave applications. Leveraging modern web technologies such as Spring Boot, Thyme leaf, and MongoDB, the system offers a user-friendly and secure environment that enhances organizational productivity by reducing administrative burdens and minimizing errors. The system's architecture includes a presentation layer for user interaction, an application layer for business logic, and a data layer for storage. Key features include employee registration, login, profile management, leave management, and an administrative dashboard. Benefits of the system include increased efficiency, improved data accuracy, enhanced user experience, and better leave management. Future enhancements could include a mobile application, advanced reporting and analytics, integration with other HR systems, customizable workflows, and machine learning capabilities to predict employee behavior.

**Project Objective:**

The primary objective of the Employee Management System is to create a robust and efficient platform for managing employee data and leave applications. This system enables seamless interaction between employees and administrators, facilitating tasks such as employee registration, login, profile management, and leave management. The goal is to provide an intuitive and secure environment that streamlines administrative processes and enhances overall organizational efficiency.

**Tools and Technologies**

* **Spring Boot**: A framework for building web applications and microservices with Java.
* **Thyme leaf**: A template engine for rendering web pages in Spring Boot applications.
* **Mockito**: A testing framework for mocking objects in unit tests.
* **Java**: The programming language used to develop the application.
* **MongoDB**: A NoSQL database used for storing employee and leave data.
* **Maven**: A build automation tool used to manage project dependencies and build lifecycle.
* **Eclipse**: Integrated Development Environments (IDEs) used for writing and managing the codebase.

### **Background of the Project**

The Employee Management System is designed to address common challenges faced by organizations in managing employee data and leave applications. Traditional methods of handling such tasks often involve manual processes and paperwork, leading to inefficiencies and errors. By leveraging modern web technologies and frameworks, this project aims to create an automated, user-friendly, and secure system that simplifies employee management and enhances organizational productivity.

**Functional Requirements**

**The Admin User Should Be Able To:**

1. **Login:** Authenticate using their email and password to access the system.
2. **View Dashboard:** Access an administrative dashboard that provides an overview of employee leave applications and statuses.
3. **Approve Leave:** Approve leave applications submitted by employees, changing their status to "Approved".
4. **Decline Leave**: Decline leave applications submitted by employees, changing their status to "Declined".
5. **Manage Employees:** View and manage employee profiles, including the ability to update details or deactivate accounts.

**The Employee Should Be Able To:**

1. **Register:** Create a new account by providing necessary personal and contact details.
2. **Login:** Authenticate using their email and password to access their account.
3. **Update Profile:** Modify and update personal information such as name, contact details, and job position.
4. **Change Password:** Update their login password for security purposes.
5. **Apply for Leave:** Submit leave applications specifying the start date, end date, and reason for leave.
6. **View Leave Status:** Check the status of their submitted leave applications, whether pending, approved, or declined.
7. **View Leave History:** Access a record of all their past leave applications and their outcomes.

**Back End Development**

The back-end development of the Employee Management System involves setting up a Spring Boot application that interacts with MongoDB to manage employee and leave data. The back end provides RESTful web services that handle various operations such as employee registration, login, profile management, and leave management.

**Key Components:**

**Entities**: Represent the data models for employees and leaves.

### **Employee Class**

* **Description**: The Employee class represents an employee entity in the Employee Management System. It is annotated with @Document to specify that it is a MongoDB document stored in the "employees" collection.

### **Leave Class**

* **Description**: The Leave class represents a leave entity in the Employee Management System. It is annotated with @Document to specify that it is a MongoDB document stored in the "leaves" collection.

**Repositories**: Interface with the MongoDB database.

### **Employee Repository Interface**

* **Description**: The Employee Repository interface is a repository for managing Employee entities in the MongoDB database. It extends the Mongo Repository interface, providing standard CRUD operations and additional custom methods for querying employee data.

### **Leave Repository Interface**

* **Description**: The Leave Repository interface is a repository for managing Leave entities in the MongoDB database. It extends the Mongo Repository interface, providing standard CRUD operations and additional custom methods for querying leave data.

**Services:** Contain the business logic for the application.

### **Employee Service Class**

* **Description**: The Employee Service class provides business logic related to employee operations. It includes methods for registering employees, logging in, updating profiles, and changing passwords. This class interacts with the Employee Repository to perform CRUD operations on the employee data.

### **Leave Service Class**

* **Description**: The Leave Service class manages leave-related operations, including applying for leave, fetching leave records, and approving or declining leave requests. This class interacts with both the Leave Repository and Employee Repository to handle leave applications and associated employee data.

**Controllers**: Handle HTTP requests and return responses.

### **Admin Controller Class**

* **Description**: The Admi Controller class manages the functionalities for the admin user. It provides endpoints for admin login and dashboard access, where admins can view and manage leave applications.

### **Employee Controller Class**

* **Description**: The Employee Controller class handles employee-related operations, including registration, login, profile management, password changes, and viewing leave status

### **Leave Controller Class**

* **Description**: The Leave Controller class manages leave-related operations, including applying for leave, approving leave, and declining leave.

**Data initializer**: For Admin Credentials.

### **Data Initializer Class**

* **Description**: The Data Initializer class is responsible for initializing the database with default data when the application starts. Specifically, it ensures that an admin user is created if one does not already exist. This setup is useful for applications that need a default admin account for initial access and management.

**Unit Testing**

Unit testing is a critical part of the development process to ensure that individual components work as expected. Mockito is used to mock dependencies and simulate the behavior of the service layer during testing.

**Testing Services:**

* **Employee Service Test**:
  + Test methods for registering an employee, logging in, updating profiles, and changing passwords.
  + Use @Mock-to-mock Employee Repository and @InjectMocks to inject Employee Service.
* **Leave Service Test**:
  + Test methods for applying for leave, getting leaves, approving, and declining leaves.
  + Use @Mock-to-mock Leave Repository and Employee Repository, and @InjectMocks to inject Leave Service.

**Testing Controllers:**

* **Admin Controller Test**:
  + Test admin functionalities such as login and dashboard view.
  + Use Mock Mvc to perform HTTP requests and @MockBean to mock service dependencies.
* **Employee Controller Test**:
  + Test employee functionalities such as registration, login, profile update, and leave status.
  + Use Mock Mvc to perform HTTP requests and @MockBean to mock service dependencies.
* **Leave Controller Test**:
  + Test leave functionalities such as applying for leave, approving, and declining leave requests.
  + Use MockMvc to perform HTTP requests and @MockBean to mock service dependencies.

By following these steps and performing comprehensive unit testing, the Employee Management System can be built robustly and effectively, ensuring the correctness of its functionalities.

**Front-End Development**

The front-end development of the Employee Management System involves creating an intuitive and user-friendly interface that facilitates smooth interactions between the users (employees and administrators) and the system. The front-end is built using HTML, CSS, and Thyme leaf, which integrates seamlessly with Spring Boot to render dynamic web pages.

**Technologies Used**

* **HTML**: Used for structuring the content of the web pages.
* **Inline CSS**: Used for styling the web pages to enhance their visual appeal.
* **Thyme leaf**: A server-side Java template engine used to render dynamic HTML content.

**Key Components**

1. **HTML Templates**
   * **admin-leaves.html**: Displays the list of leave applications for the admin to review.
   * **admin-login.html**: Admin login page.
   * **employee-leaves.html**: Displays the leave history for an employee.
   * **employee-profile.html**: Displays and allows updates to the employee's profile.
   * **leave-status.html**: Shows the status of the employee's leave applications.
   * **login.html**: Employee login page.
   * **register.html**: Employee registration page.
2. **CSS (Inline)**

Custom CSS styles are defined in styles.css to enhance the appearance of the web pages. This includes layout adjustments, color schemes, fonts, and responsive design considerations to ensure the application looks good on various devices.

**Integration with Thyme leaf**

Thyme leaf templates are used to dynamically render HTML content based on the data provided by the Spring Boot backend. This allows for seamless integration between the front-end and back-end, ensuring that user interactions are handled efficiently.

**Conclusion**

The Employee Management System stands as a comprehensive solution designed to tackle the challenges associated with managing employee data and leave applications. By leveraging the power of modern web technologies such as Spring Boot, Thyme leaf, and MongoDB, this system provides an efficient, automated, and user-friendly platform that caters to both employees and administrators.

Throughout the documentation, we have explored the intricacies of the system's architecture, the functionalities offered, and the methodologies employed in its development. The back-end development, enriched with robust service and repository layers, ensures reliable data handling and business logic execution. Meanwhile, the front-end development, with its intuitive design and seamless integration with Thyme leaf, ensures a smooth user experience.

The Employee Management System not only streamlines routine administrative tasks but also enhances data accuracy and security, thereby freeing up valuable time for human resources to focus on strategic initiatives. By automating the management of employee information and leave applications, the system reduces errors and increases productivity within the organization.

In summary, this system represents a significant advancement in employee management practices, providing a scalable and secure solution that meets the needs of modern organizations. As technology continues to evolve, the Employee Management System is well-positioned to adapt and grow, ensuring that it remains a vital tool for organizational efficiency and effectiveness.

We believe that the Employee Management System will play a crucial role in transforming how organizations handle employee data and leave management, paving the way for a more efficient and productive workplace. We hope this documentation provides a clear understanding of the system's capabilities and encourages its successful implementation and use.