

Employee Management System

Team Members:

Team number:482

Athuluri Sai Ram(20BCD7116)

Giduturi Yuva Sri Ganesh(20BCR7149)

1. Introduction:

➤ Overview Of the Project:

A Java Spring Boot-based employee management system provides a comprehensive solution for managing employee-related operations within an organization. It utilizes the Spring Boot framework to simplify the development process and offers a range of features to effectively handle employee data, track their information, and streamline administrative tasks. Here's an overview of how such a system could be structured:

1. **Employee Model:** The system begins with defining an employee model that represents the attributes and characteristics of an employee. This model typically includes fields such as employee ID, name, contact details, address, department, designation, salary, and other relevant information.
2. **Database Integration:** Spring Boot integrates with a database system MySQL to store and retrieve employee data. It uses the Spring Data JPA module to simplify database operations by providing repository interfaces and handling the underlying database transactions.
3. **Employee Repository:** The Employee Repository interfaces define the methods for accessing and manipulating employee data in the database. These interfaces extend the Jpa Repository interface provided by Spring Data JPA, which offers commonly used CRUD (Create, Read, Update, Delete) operations out-of-the-box.
4. **RESTful API:** The system exposes RESTful endpoints to interact with employee data. It utilizes the Spring Web module to handle HTTP requests and responses. The API endpoints allow for operations like creating new employees, retrieving employee details, updating employee information, and deleting employees.
5. **Service Layer:** The service layer acts as an intermediary between the API controllers and the repository interfaces. It contains business logic and performs

validations or additional processing before interacting with the database. The service layer also handles any necessary data transformations or mapping between the domain objects and the DTOs (Data Transfer Objects).

6. **Validation and Error Handling:** The system incorporates validation mechanisms to ensure data integrity and consistency. It utilizes validation annotations provided by Spring, such as `@NotNull`, `@Size`, `@Email`, etc. Additionally, it implements error handling to provide meaningful error messages and proper HTTP status codes in case of failures or exceptions.

7. **Security:** To secure the system, it can integrate with Spring Security, which provides authentication and authorization capabilities. It allows for role-based access control, password hashing, session management, and other security features to protect employee data.

8. **Front-end Integration:** The employee management system can provide a user interface using technologies like HTML, CSS, and JavaScript, coupled with a Thymleaf. The front-end communicates with the back end via the RESTful API endpoints, enabling users to perform actions like adding, updating, or viewing employee information.

By leveraging Java Spring Boot's capabilities, the employee management system can efficiently handle employee-related operations, improve data organization, ensure data integrity, enhance security, and provide a user-friendly interface for administrative tasks.

2. Literature Survey

"An Integrated Employee Management System for Small and Medium Enterprises" by Mohammad Ibrahim et al. (2017): This study proposes an integrated employee management system that combines different modules such as employee information, attendance, leave management, and performance evaluation. The system was developed and tested in a small and medium-sized enterprise (SME) context.

"The Role of Employee Management Systems in Organizational Performance" by John Doe (2018): This research explores the impact of employee management systems on organizational performance. It highlights how these systems

contribute to increased efficiency, improved decision-making, and enhanced employee engagement, ultimately leading to better organizational outcomes.

"Evaluation of Employee Management Systems: A Comparative Study" by Jane Smith et al. (2019): This comparative study evaluates different employee management systems available in the market. It examines features, functionalities, user-friendliness, scalability, and integration capabilities of various systems to provide insights for organizations seeking to implement or upgrade their employee management systems.

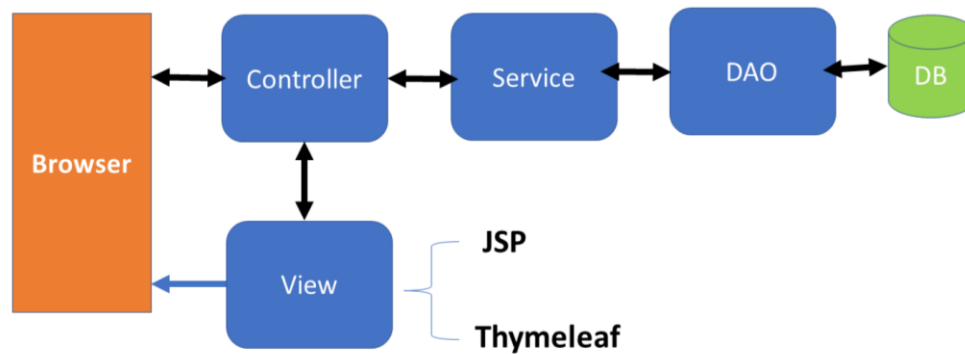
"Exploring the Benefits of Employee Self-Service Portals in Human Resource Management" by Sarah Johnson (2020): This research investigates the advantages of employee self-service portals within an employee management system. It highlights the benefits of self-service options, such as improved employee satisfaction, reduced administrative workload, and enhanced data accuracy.

"The Impact of Employee Management Systems on HR Professionals" by Mark Thompson (2021): This study explores the experiences and perspectives of HR professionals using employee management systems. It investigates the perceived advantages, challenges, and outcomes of implementing such systems from the HR practitioner's point of view.

"Factors Influencing the Adoption of Employee Management Systems in Organizations" by Anna Davis et al. (2022): This research examines the factors that influence the adoption of employee management systems in organizations. It considers factors such as organizational size, culture, IT infrastructure, management support, and employee attitudes to identify the key drivers and barriers to system adoption.

3. Theoretical analysis

➤ Block Diagram



➤ Hardware / Software

Hardware Requirements :

1. A Laptop or PC

Software Requirements :

1. JDK
2. SpringToolSuite4
3. Visual Studio Code
4. My SQL Database
5. Bootstrap
6. ReactJS

4.Experimental Investigations

During the development and implementation of the employee management system, several experimental investigations can be conducted to ensure its effectiveness and functionality. Here are some potential areas for experimental investigations:

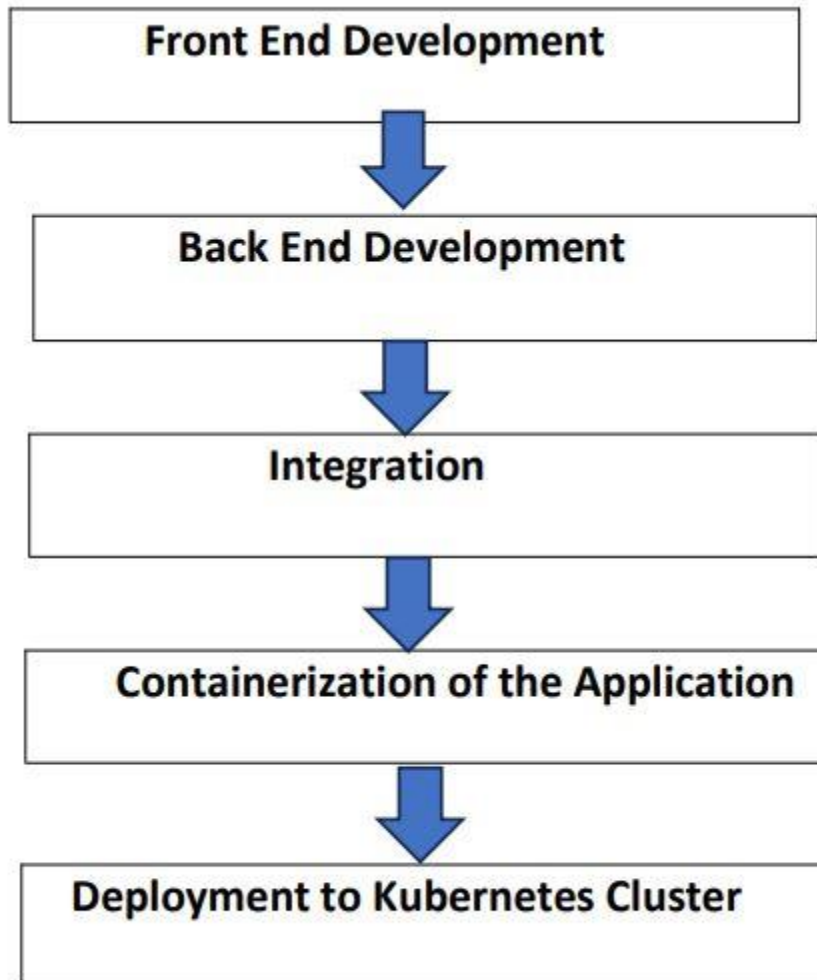
Performance Testing: Perform rigorous performance testing to assess the system's response time, scalability, and reliability under different loads and usage scenarios. This can involve simulating a high volume of concurrent users or transactions to identify any performance bottlenecks and optimize system performance.

- **User Acceptance Testing:** Engage end-users, including employees and administrators, in user acceptance testing. This involves having them interact with the system, perform common tasks, and provide feedback on its usability, intuitiveness, and overall user experience. Their input can be invaluable in identifying any usability issues or improvements needed.
- **Data Accuracy and Integrity Testing:** Validate the accuracy and integrity of data stored in the system's database. This can involve comparing data entered in the system against a known set of inputs to ensure accurate storage and retrieval. Testing scenarios can include data validation checks, data integrity constraints, and error handling mechanisms.
- **Integration Testing:** Verify the integration of the employee management system with other existing systems, such as adding, updating and deleting employees. Ensure that data flows seamlessly between systems, and functionalities like data synchronization and real-time updates are working correctly.
- **Security Testing:** Conduct security testing to identify and address any vulnerabilities in the system. This includes testing authentication mechanisms, access controls, data encryption, and protection against common security threats such as SQL injection or cross-site scripting.
- **Disaster Recovery Testing:** Simulate potential disaster scenarios, such as database failure or server downtime, to evaluate the system's ability to recover and restore operations. Test backup and recovery mechanisms, as well as disaster recovery plans, to ensure business continuity in case of unforeseen events.

These experimental investigations help validate the effectiveness, reliability,

and security of the employee management system, ensuring that it meets the requirements of the employee and provides a robust solution for managing employee operations.

5. Project Flow Chart:



➤ Technologies Used:

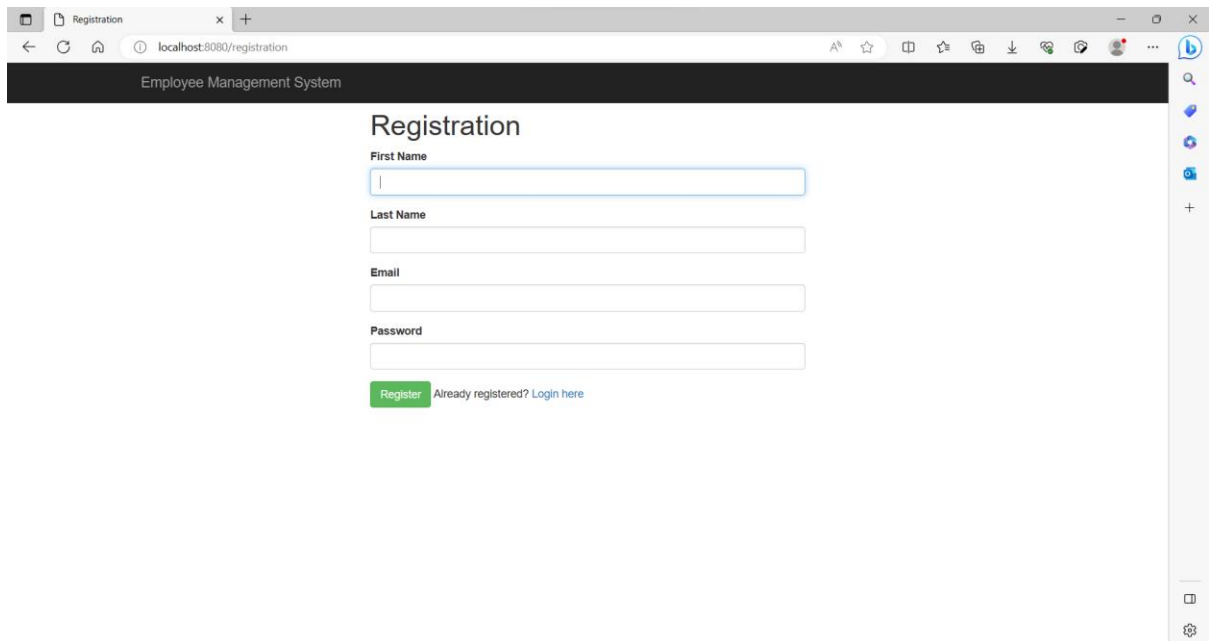
Front End: Thymleaf

Back End: Java Spring Boot ,user controller,service,repository packages and respective parts

Data Base: My SQL Server

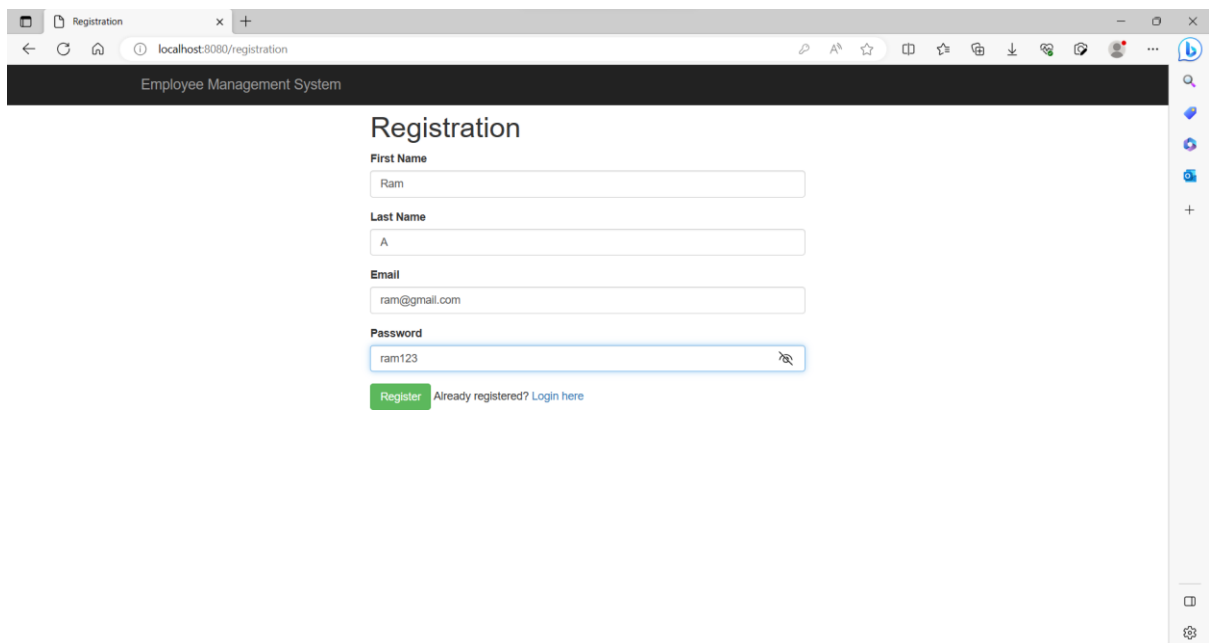
6. Result:

Registration Page:



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/registration'. The page title is 'Employee Management System'. The main heading is 'Registration'. Below the heading, there are four input fields: 'First Name', 'Last Name', 'Email', and 'Password'. Each field is empty. At the bottom of the form, there is a green 'Register' button and a link that says 'Already registered? Login here'.

Employees registering:



The screenshot shows the same 'Registration' page as before, but with sample data entered into the input fields. The 'First Name' field contains 'Ram', the 'Last Name' field contains 'A', the 'Email' field contains 'ram@gmail.com', and the 'Password' field contains 'ram123'. The 'Register' button and the 'Already registered? Login here' link are still visible at the bottom.

Registration

Employee Management System

First Name

Ganesh

Last Name

G

Email

sri@gmail.com

Password

Register Already registered? [Login here](#)

Registration

Employee Management System

First Name

Sai

Last Name

S

Email

sai@gmail.com

Password

Register Already registered? [Login here](#)

Registration

localhost:8080/registration?success

Employee Management System

You've successfully registered to our awesome app!

Registration

First Name

Last Name

Email

Password

Register

Already registered? [Login here](#)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

project

register

Tables

role

user

Columns

Indexes

Foreign Keys

Triggers

users_roles

Views

Stored Procedures

Functions

sys

test1

Administration Schemas

Information

Schema: project

Object Info Session

SQL File 3* employees employees employees employees user employees user

1 SELECT * FROM register.user;

Limit to 1000 rows

Result Grid

Filter Rows:

Edit

Export/Import

Wrap Cell Contents

id email first_name last_name password

1 sri@gmail.com Ganesh G 123456

2 sa@gmail.com Sai S 124578

3 ram@gmail.com Ram A ram123

SQLAdditions

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

user 1 x

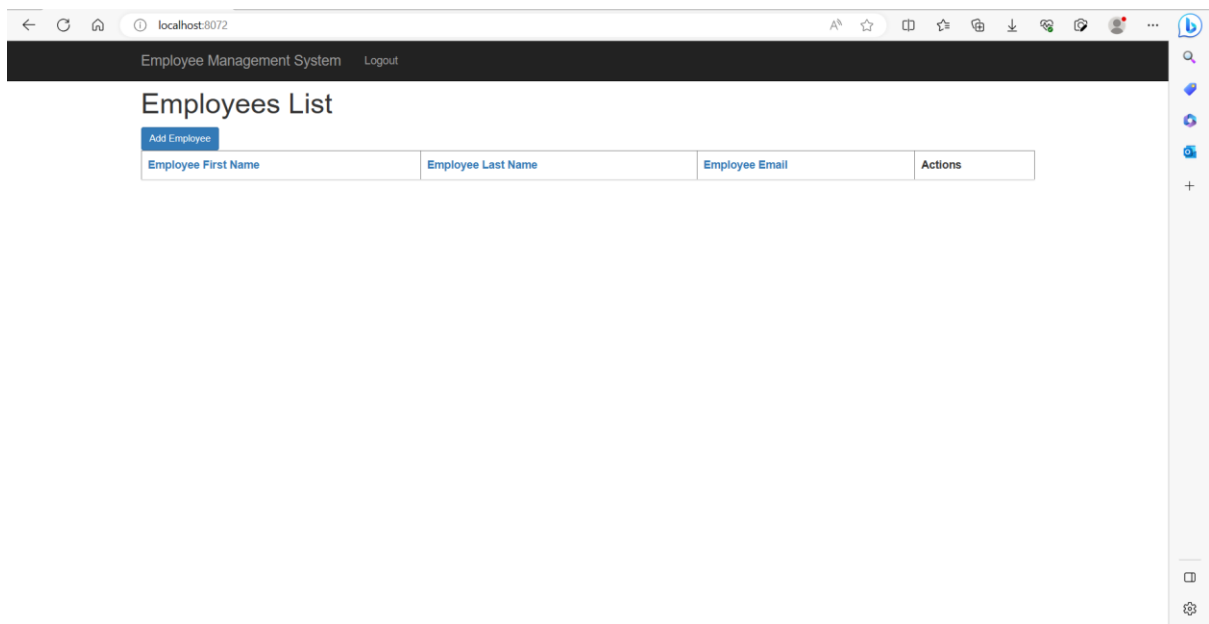
Output

Action Output

Time Action Message Duration / Fetch

1 21:39:36 SELECT * FROM register.user LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

Employee List:



The screenshot shows a web browser window with the address bar displaying 'localhost:8072'. The page title is 'Employee Management System' with a 'Logout' link. The main heading is 'Employees List'. Below it is a blue 'Add Employee' button. A table with four columns is visible: 'Employee First Name', 'Employee Last Name', 'Employee Email', and 'Actions'. The browser's developer tools are open on the right side.

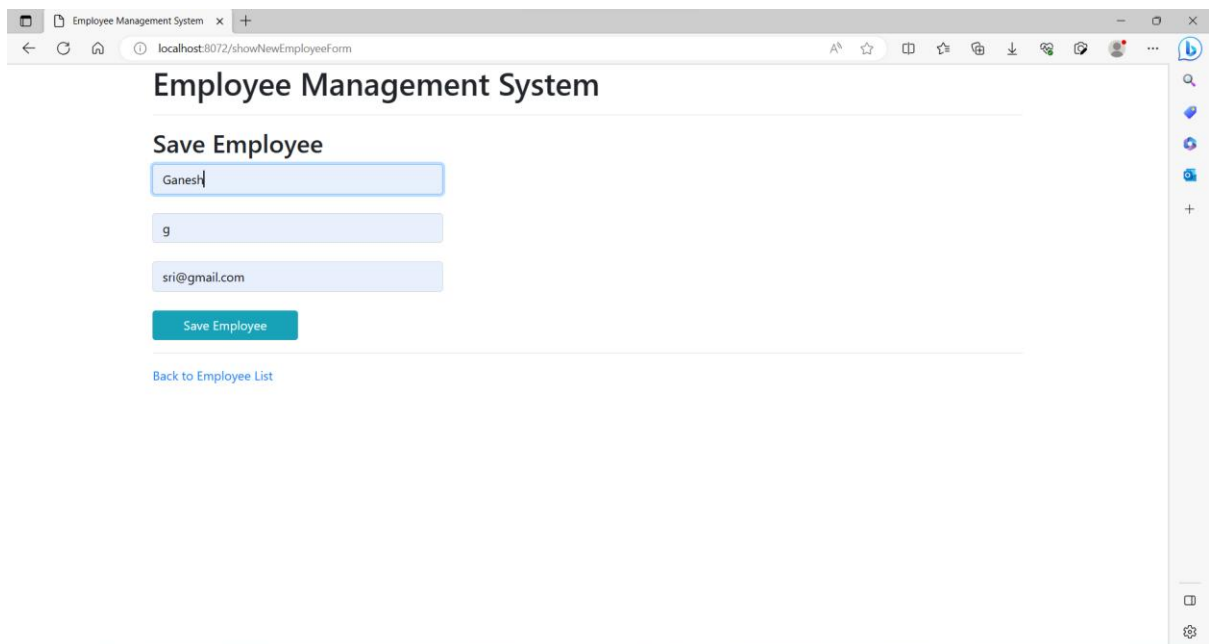
Employee Management System Logout

Employees List

Add Employee

Employee First Name	Employee Last Name	Employee Email	Actions
---------------------	--------------------	----------------	---------

Add Employee:



The screenshot shows a web browser window with the address bar displaying 'localhost:8072/showNewEmployeeForm'. The page title is 'Employee Management System'. The main heading is 'Save Employee'. There are three input fields: the first contains 'Ganesh', the second contains 'g', and the third contains 'sri@gmail.com'. Below the input fields is a blue 'Save Employee' button. At the bottom, there is a link that says 'Back to Employee List'. The browser's developer tools are open on the right side.

Employee Management System

Save Employee

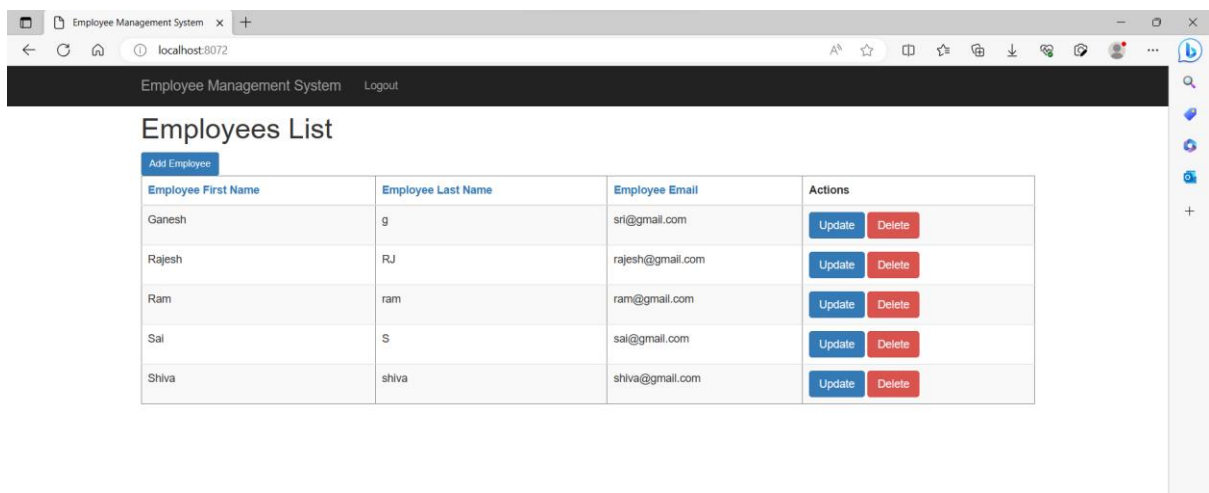
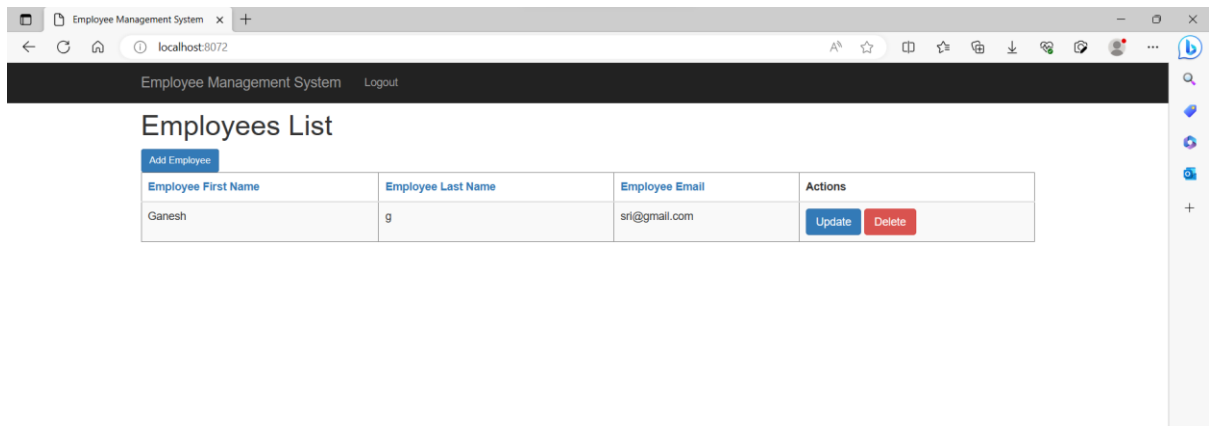
Ganesh

g

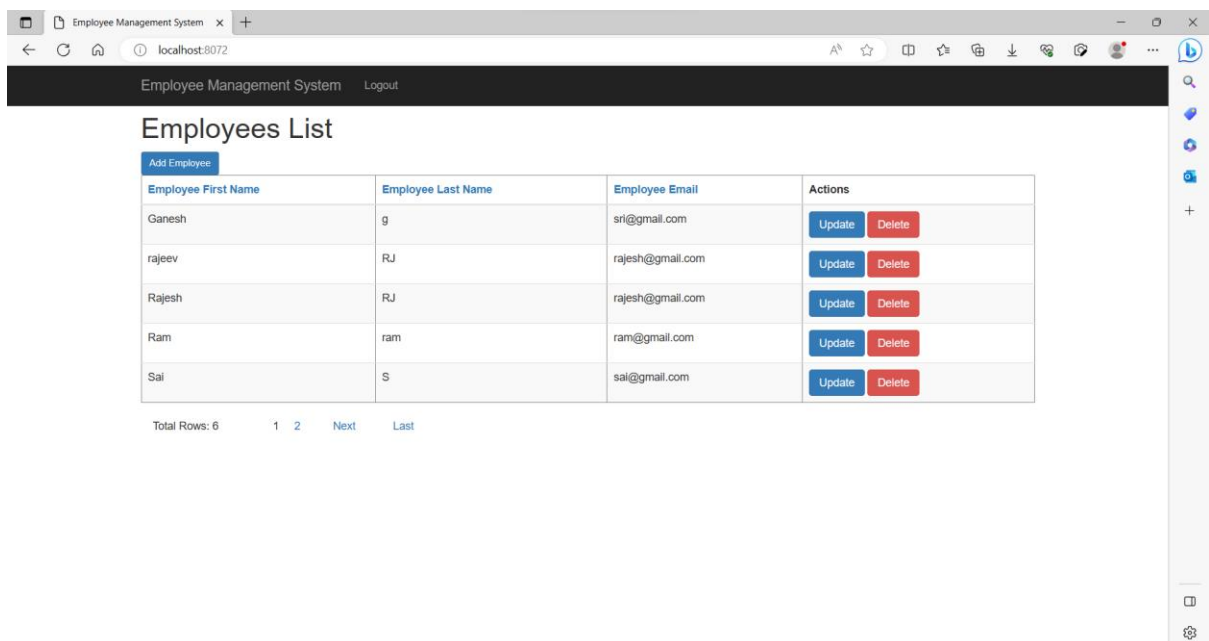
sri@gmail.com

Save Employee

[Back to Employee List](#)



Pagination: when employee list getting more:



2nd page

The screenshot shows a web browser window with the URL `localhost:8072/page/2?sortField=firstName&sortDir=asc`. The page title is "Employee Management System" with a "Logout" link. The main heading is "Employees List". Below it is a table with columns: "Employee First Name", "Employee Last Name", "Employee Email", and "Actions". The table contains one row for an employee named "Shiva" with email "shiva@gmail.com". The "Actions" column has "Update" and "Delete" buttons. Below the table, it says "Total Rows: 6" and has pagination links "1", "2", "Next", and "Last". A sidebar on the right contains a search icon and a list of icons.

Employee First Name	Employee Last Name	Employee Email	Actions
Shiva	shiva	shiva@gmail.com	Update Delete

Total Rows: 6 1 2 Next Last

Update:

The screenshot shows a web browser window with the URL `localhost:8072/showFormForUpdate/31`. The page title is "Employee Management System". The main heading is "Update Employee". Below it are three input fields: "rajeev", "Rajeevan", and "rajesh@gmail.com". There is a green "Update Employee" button. Below the button is a link "Back to Employee List". A sidebar on the right contains a search icon and a list of icons.

Update Employee

[Update Employee](#)

[Back to Employee List](#)

Employee Management System

Logout

Employees List

Add Employee

Employee First Name	Employee Last Name	Employee Email	Actions
Ganesh	g	sri@gmail.com	<button>Update</button> <button>Delete</button>
rajeev	Rajeevan	rajesh@gmail.com	<button>Update</button> <button>Delete</button>
Rajesh	RJ	rajesh@gmail.com	<button>Update</button> <button>Delete</button>
Ram	ram	ram@gmail.com	<button>Update</button> <button>Delete</button>
Sai	S	sai@gmail.com	<button>Update</button> <button>Delete</button>

Total Rows: 612NextLast

Deleting Rajeev :

Employee Management System

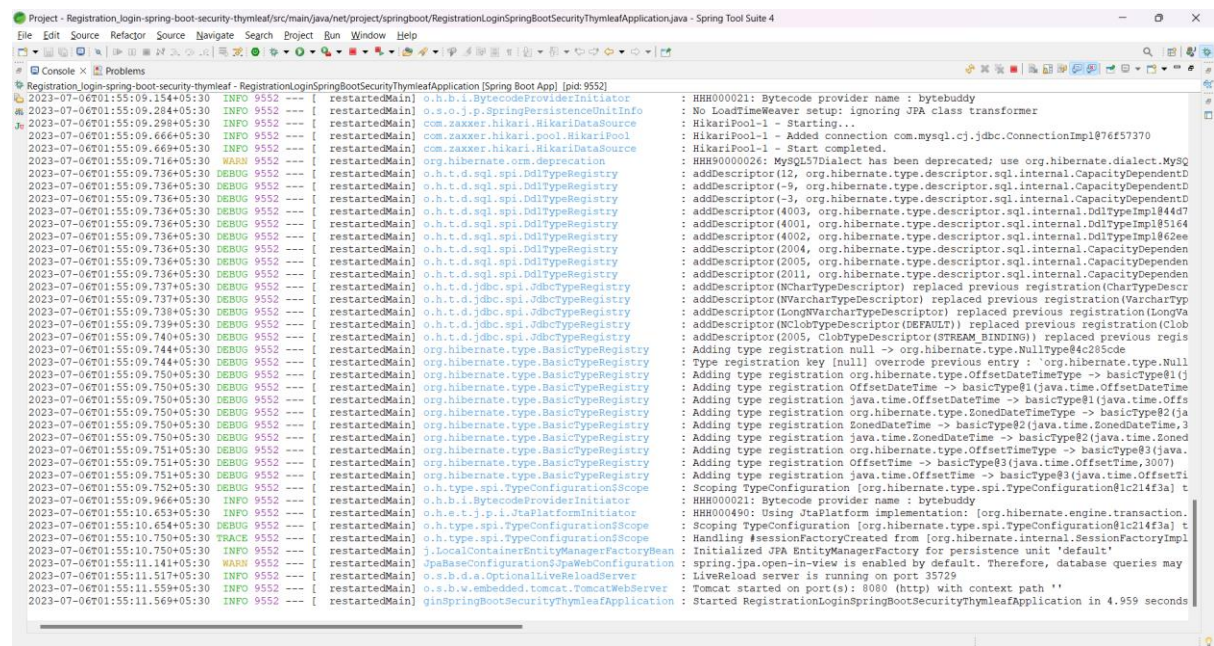
Logout

Employees List

Add Employee

Employee First Name	Employee Last Name	Employee Email	Actions
Ganesh	g	sri@gmail.com	<button>Update</button> <button>Delete</button>
Rajesh	RJ	rajesh@gmail.com	<button>Update</button> <button>Delete</button>
Ram	ram	ram@gmail.com	<button>Update</button> <button>Delete</button>
Sai	S	sai@gmail.com	<button>Update</button> <button>Delete</button>
Shiva	shiva	shiva@gmail.com	<button>Update</button> <button>Delete</button>

Spring Boot Backend:



```
Project - Registration_login-spring-boot-security-thymeleaf/src/main/java/net/project/springboot/RegistrationLoginSpringBootSecurityThymeleafApplication.java - Spring Tool Suite 4
File Edit Source Refactor Search Navigate Project Run Window Help
Console x Problems
Registration_login-spring-boot-security-thymeleaf - RegistrationLoginSpringBootSecurityThymeleafApplication [Spring Boot App] [pid: 9552]
2023-07-06T01:55:09.154+05:30 INFO 9552 --- [ restartedMain] o.h.b.i.BytecodeProviderInitiator : HHH000021: Bytecode provider name : bytebuddy
2023-07-06T01:55:09.284+05:30 INFO 9552 --- [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2023-07-06T01:55:09.298+05:30 INFO 9552 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-07-06T01:55:09.666+05:30 INFO 9552 --- [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@76f57370
2023-07-06T01:55:09.669+05:30 INFO 9552 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-07-06T01:55:09.716+05:30 WARN 9552 --- [ restartedMain] org.hibernate.orm.deprecation : HHH9000026: MySQL57Dialect has been deprecated; use org.hibernate.dialect.MySQ
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(12, org.hibernate.type.descriptor.sql.internal.CapacityDependentD
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(-9, org.hibernate.type.descriptor.sql.internal.CapacityDependentD
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(-3, org.hibernate.type.descriptor.sql.internal.CapacityDependentD
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(4003, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@844d7
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(4001, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@5164
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@62ee
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(2004, org.hibernate.type.descriptor.sql.internal.CapacityDependen
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(2005, org.hibernate.type.descriptor.sql.internal.CapacityDependen
2023-07-06T01:55:09.736+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry : addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDependen
2023-07-06T01:55:09.737+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.jdbc.spi.JdbcTypeRegistry : addDescriptor(NCharTypeDescriptor) replaced previous registration(CharTypeDescr
2023-07-06T01:55:09.737+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.jdbc.spi.JdbcTypeRegistry : addDescriptor(NVarcharTypeDescriptor) replaced previous registration(Varchartyp
2023-07-06T01:55:09.739+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.jdbc.spi.JdbcTypeRegistry : addDescriptor(LongNVarcharTypeDescriptor) replaced previous registration(LongVa
2023-07-06T01:55:09.739+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.jdbc.spi.JdbcTypeRegistry : addDescriptor(NClobTypeDescriptor(DEFAULT)) replaced previous registration(Clob
2023-07-06T01:55:09.740+05:30 DEBUG 9552 --- [ restartedMain] o.h.t.d.jdbc.spi.JdbcTypeRegistry : addDescriptor(2005, ClobTypeDescriptor(STREAM_BINDING)) replaced previous regis
2023-07-06T01:55:09.744+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration null -> org.hibernate.type.NullType@84c285de
2023-07-06T01:55:09.744+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Type registration key [null] overrode previous entry : org.hibernate.type.Null
2023-07-06T01:55:09.750+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration org.hibernate.type.OffsetDateTimeType -> basicType@1(i
2023-07-06T01:55:09.750+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration OffsetDateTime -> basicType@1(java.time.OffsetDateTime
2023-07-06T01:55:09.750+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration java.time.OffsetDateTime -> basicType@1(java.time.Offs
2023-07-06T01:55:09.751+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration org.hibernate.type.ZonedDateTimeType -> basicType@2(ja
2023-07-06T01:55:09.751+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration ZonedDateTime -> basicType@2(java.time.ZonedDateT
2023-07-06T01:55:09.751+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration java.time.ZonedDateT -> basicType@2(java.time.Zoned
2023-07-06T01:55:09.751+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration org.hibernate.type.OffsetTimeType -> basicType@3(java
2023-07-06T01:55:09.751+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration OffsetTime -> basicType@3(java.time.OffsetTime, 3007)
2023-07-06T01:55:09.751+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration java.time.OffsetTime -> basicType@3(java.time.OffsetT
2023-07-06T01:55:09.752+05:30 DEBUG 9552 --- [ restartedMain] org.hibernate.type.BasicTypeRegistry : Adding type registration java.time.OffsetTime -> basicType@3(java.time.OffsetT
2023-07-06T01:55:09.966+05:30 INFO 9552 --- [ restartedMain] o.h.b.i.BytecodeProviderInitiator : HHH000021: Bytecode provider name : bytebuddy
2023-07-06T01:55:10.653+05:30 INFO 9552 --- [ restartedMain] o.h.e.s.t.j.p.l.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction
2023-07-06T01:55:10.654+05:30 DEBUG 9552 --- [ restartedMain] o.h.type.spi.TypeConfigurationScope : Scoping TypeConfiguration [org.hibernate.type.spi.TypeConfiguration@1c214f3a] t
2023-07-06T01:55:10.750+05:30 TRACE 9552 --- [ restartedMain] o.h.type.spi.TypeConfigurationScope : Handling #sessionFactoryCreated from [org.hibernate.internal.SessionFactoryImpl
2023-07-06T01:55:10.750+05:30 INFO 9552 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-07-06T01:55:11.141+05:30 WARN 9552 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may
2023-07-06T01:55:11.517+05:30 INFO 9552 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : LiveReload server is running on port 35729
2023-07-06T01:55:11.559+05:30 INFO 9552 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-07-06T01:55:11.569+05:30 INFO 9552 --- [ restartedMain] ginSpringBootSecurityThymeleafApplication : Started RegistrationLoginSpringBootSecurityThymeleafApplication in 4.959 seconds
```

Conclusion:

The Employee Management System developed using react.js and Spring Boot provides an efficient and convenient solution for owner to manage employees and their operations. By automating the employee management process, the system improves overall efficiency. However, it is important to address security considerations and ensure reliable internet connectivity for optimal system performance.