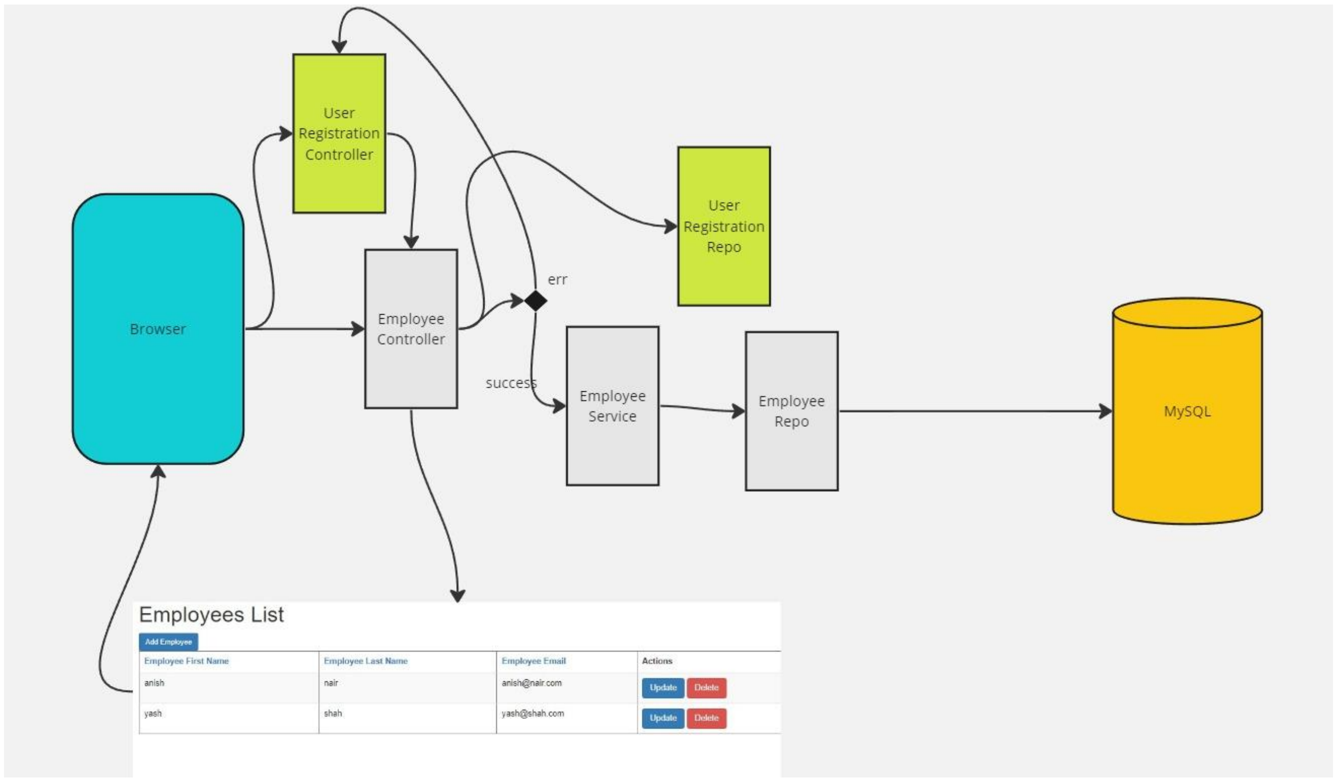
Employee Management System

– B035 B042 B050

Project Overview

The Employee Management System provides an easy-to-use web interface for managing employee information. The system allows users to add, edit, delete, and view employee information in real-time. The system uses Spring Boot, Spring Security, Thymeleaf, and MySQL database to provide a robust and scalable solution.

Modules



Employees List

Employee First Name	Employee Last Name	Employee Email	Actions
anish	nair	anish@nair.com	Update Delete
yash	shah	yash@shah.com	Update Delete

1. **Employee Management Module**: This module provides the core functionality for managing employee information. The module uses Spring Boot and Thymeleaf to create a dynamic web application that allows users to perform CRUD operations on employee information. The module uses a MySQL database to store employee information.
2. **Registration and Login Module**: This module provides user authentication and authorization functionality. The module uses Spring Security to create a secure login system

that allows users to log in and access the Employee Management Module. The module uses a MySQL database to store user information.

Dependencies

The following dependencies are used in the project:

3. **Spring Boot Starter Web**: Provides framework for building web applications.
4. **Spring Boot Starter Thymeleaf**: Provides integration with Thymeleaf template engine.
5. **Spring Boot Starter Data JPA**: Provides integration with JPA and Spring Data.
6. **Spring Boot Starter Security**: Provides integration with Spring Security.
7. **MySQL Connector/J**: Provides database connection to MySQL.
8. **Spring Boot Starter Test**: Provides support for testing.
9. **Log4j2**: Provides logging framework.

Employee Management Module

EmployeeController.java
This code snippet shows the EmployeeController class that provides the REST API for managing employee information.

```
@RestController
@RequestMapping("/api/employees")
public class EmployeeController {

    private EmployeeService employeeService;

    @Autowired
    public EmployeeController(EmployeeService employeeService) {
        this.employeeService = employeeService;
     }

     @GetMapping
     public List<Employee> getAllEmployees() {
         return employeeService.getAllEmployees();
```

```java
    }

    @GetMapping("/{id}")
    public Employee getEmployeeById(@PathVariable Long id) {
        return employeeService.getEmployeeById(id);
    }

    @PostMapping
    public Employee createEmployee(@RequestBody Employee employee) {
        return employeeService.createEmployee(employee);
    }

    @PutMapping("/{id}")
    public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee employee) {
        return employeeService.updateEmployee(id, employee);
    }

    @DeleteMapping("/{id}")
    public void deleteEmployee(@PathVariable Long id) {
        employeeService.deleteEmployee(id);
    }
}
```

Registration and Login Module

UserController.java

This code snippet shows the UserController class that provides the REST API for managing user information.

```java
@RestController
@RequestMapping("/api/users")
```

```java
public class UserController {

    private UserService userService;

    @Autowired
    public UserController(UserService userService) {
        this.userService = userService;
    }

    @PostMapping
    public User createUser(@RequestBody User user) {
        return userService.createUser(user);
    }

    @GetMapping("/{id}")
    public User getUserById(@PathVariable Long id) {
        return userService.getUserById(id);
    }

    @PutMapping("/{id}")
    public User updateUser(@PathVariable Long id, @RequestBody User user) {
        return userService.updateUser(id, user);
    }

    @DeleteMapping("/{id}")
    public void deleteUser(@PathVariable Long id) {
        userService.deleteUser(id);
    }

    @PostMapping("/login")
    public ResponseEntity<UserDto> loginUser(@RequestBody UserDto userDto) {
        UserDto user = userService.loginUser(userDto);
        return ResponseEntity.ok(user);
    }
}
```

SecurityConfig.java
This code snippet shows the SecurityConfig class that provides the configuration for Spring Security.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private UserService userService;

    @Autowired
    public SecurityConfig(UserService userService) {
        this.userService = userService;
     }

     @Override

The Employee Management    protected void configure(HttpSecurity http) throws Exception {
          http.authorizeRequests()
                  .antMatchers("/api/users/login System you have described").permitAll()
                  .anyRequest().authenticated()
                  .and()
              . is a comprehformLogin()
                  .loginPage("/login")
                  .defaultSuccessUrl("/")
                  .permitAll()
                  .and()
              .logout()
                  .ensive web applicationlogoutUrl("/logout")
                  .deleteCookies("JSESSIONID")
                  .invalidateHttpSession(true that allows users to manage employee information)
                  .permitAll()
                  .and()
              .csrf()
                  .csrfTokenRepository(CookieC.ThesrfTokenRepository.withHttpOnlyFalse())
                  .and()
```

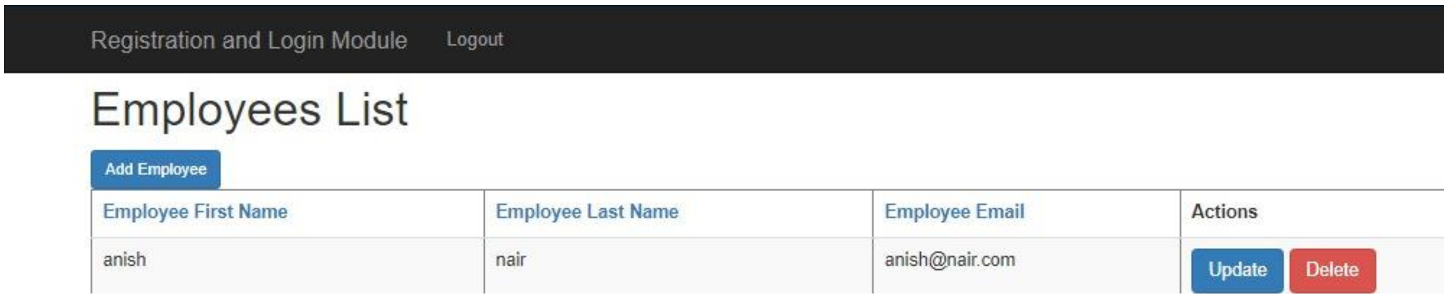
```
            .exceptionHandling()
                .authenticationEntryPoint(new Http system uses Spring Boot, Spring Security401AuthenticationEntryPoint("Unauthorized"))
                .and()
            .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and()
            .addFilter(new JwtAuthenticationFilter(, Thymeleaf, and MySQL toauthenticationManager()))
            .addFilter(new JwtAuthorizationFilter(authenticationManager(), userService));
 provide a robust    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authenticationProvider and scalable solution. = new DaoAuthenticationProvider();
        authenticationProvider.setUserDetailsService(userService);
        authenticationProvider.setPassword The system isEncoder(passwordEncoder());
        return authenticationProvider;
    }

    @Bean
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return new Provider divided into two mainManager(Arrays.asList(authenticationProvider()));
    }
}
```

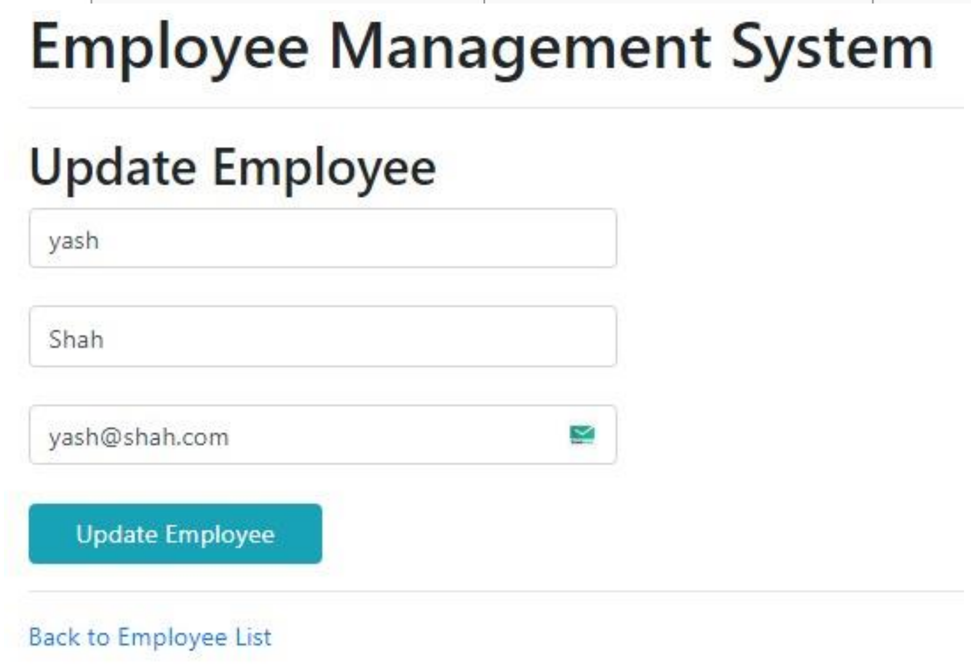
Working of the project


Registration and Login Module Logout

Employees List

Add Employee

Employee First Name	Employee Last Name	Employee Email	Actions
anish	nair	anish@nair.com	Update Delete


Employee Management System

Update Employee

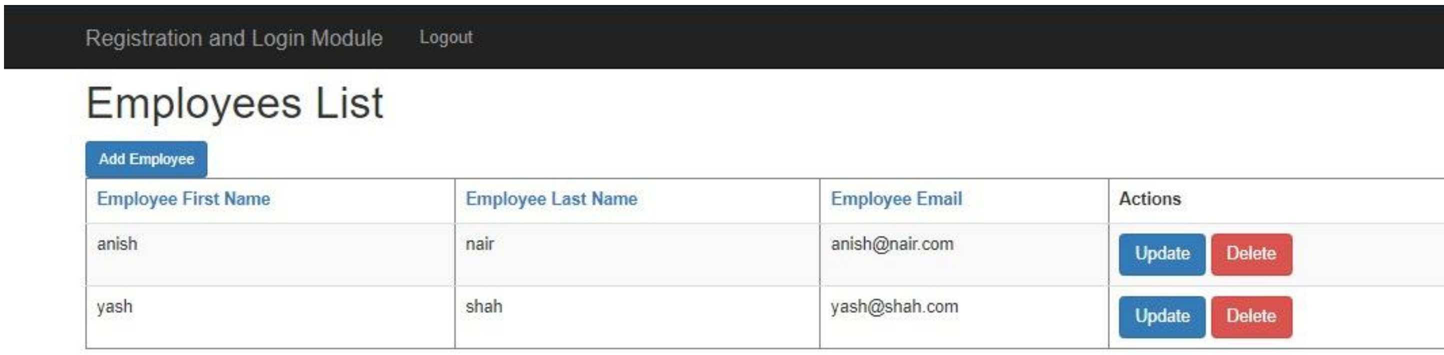
yash

Shah

yash@shah.com

Update Employee

Back to Employee List


Registration and Login Module Logout

Employees List

Add Employee

Employee First Name	Employee Last Name	Employee Email	Actions
anish	nair	anish@nair.com	Update Delete
yash	shah	yash@shah.com	Update Delete

Conclusion

The Employee Management System is a robust and scalable solution for managing employee information. The system uses Spring Boot, Spring Security, Thym The Employee Management Module provides the core functionality for managing leaf, and MySQL database to provide a fast and reliable solution. The system is easy to use and can be customized to meet specific business requirements. The project employee information is well-designed and follows best practices for software development. The project is a notable example of how to use Spring Boot and related technologies to build web applications. while the Registration and Login Module provides user authentication and authorization functionality. The system uses a MySQL database to store employee and user information. The code snippets provided in the document demonstrate how the Employee Management Module and UserController class are implemented. Overall, the project provides a comprehensive overview of the Employee Management System and its implementation using the specified technologies.