Implement Spring Security with Authentication

SpringSecurityMwApplication.java

```
package com.example.SpringSecurityMW;


import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication

public class SpringSecurityMwApplication {


	public static void main(String[] args) {

		SpringApplication.run(SpringSecurityMwApplication.class, args);

	}


}
```

SecurityConfiguration.java

```
package com.example.SpringSecurityMW.Configurations;


import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManag
erBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
```

```
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerA
dapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

   @Autowired
   UserDetailsService userDetailsService;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {
      auth.userDetailsService(userDetailsService);
   }

   @Override
   protected void configure(HttpSecurity http) throws Exception {
      http.authorizeRequests()
            .antMatchers("/admin").hasRole("ADMIN")
            .antMatchers("/user").hasAnyRole("ADMIN", "USER")
            .antMatchers("/").permitAll()
            .and().formLogin()
      .defaultSuccessUrl("/user")
      .failureUrl("/login?error=true")
      .permitAll();
   }

   @Bean
```

```
    public PasswordEncoder getPasswordEncoder() {
        return NoOpPasswordEncoder.getInstance();
    }
}
```

LoginController.java

```
package com.example.SpringSecurityMW.controllers;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class LoginController {

            @GetMapping("/")
              public String index() {
                  return ("<h1>Home Page</h1><br><a href=\"/login\">Go to login page</a>");
              }
              @GetMapping("/user")
              public String user() {
                  return ("<h1>Welcome User</h1><br><a href=\"/logout\">Logout</a>");
              }
}
```

MyUsersDetails.java

```
package com.example.SpringSecurityMW.Entities;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Arrays;
import java.util.Collection;
import java.util.List;
import java.util.stream.Collectors;

public class MyUserDetails implements UserDetails {

	private static final long serialVersionUID = 1L;

    private String userName;
    private String password;
    private boolean active;
    private List<GrantedAuthority> authorities;

    public MyUserDetails(User user) {
        this.userName = user.getUserName();
        this.password = user.getPassword();
        this.active = user.isActive();
        this.authorities = Arrays.stream(user.getRole().split(","))
                    .map(SimpleGrantedAuthority::new)
                    .collect(Collectors.toList());
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return authorities;
    }
```

```
@Override
public String getPassword() {
   return password;
}

@Override
public String getUsername() {
   return userName;
}

@Override
public boolean isAccountNonExpired() {
   return true;
}

@Override
public boolean isAccountNonLocked() {
   return true;
}

@Override
public boolean isCredentialsNonExpired() {
   return true;
}

@Override
public boolean isEnabled() {
   return active;
}
```

```
}
```

User.java

```
package com.example.SpringSecurityMW.Entities;

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

@Entity
@Table(name = "springsecurityuser")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String userName;
    private String password;
    private boolean active;
    private String role;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

```
public String getUserName() {
    return userName;
}

public void setUserName(String userName) {
    this.userName = userName;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public boolean isActive() {
    return active;
}

public void setActive(boolean active) {
    this.active = active;
}

public String getRole() {
    return role;
}

public void setRole(String roles) {
        this.role = roles;
    }
}
```

UserRepository.java

```
package com.example.SpringSecurityMW.Repositories;

import java.util.Optional;

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

import com.example.SpringSecurityMW.Entities.User;

public interface UserRepository extends JpaRepository<User, Integer> {
    Optional<User> findByUserName(String userName);
}
```

MyUserDetailsService.java

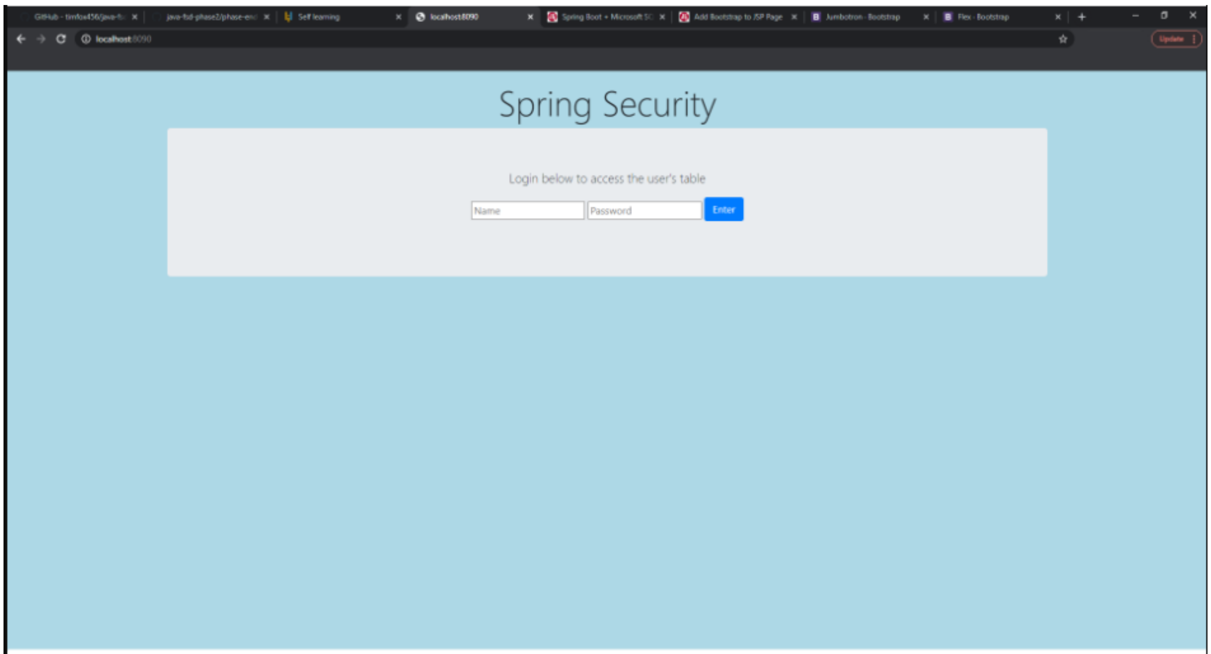
```
package com.example.SpringSecurityMW.Services;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import com.example.SpringSecurityMW.Entities.MyUserDetails;
import com.example.SpringSecurityMW.Entities.User;
import com.example.SpringSecurityMW.Repositories.UserRepository;


@Service
public class MyUserDetailsService implements UserDetailsService{
        @Autowired
    UserRepository userRepository;


        @Override
    public UserDetails loadUserByUsername(String userName) throws
UsernameNotFoundException {
        Optional<User> user = userRepository.findByUserName(userName);


        user.orElseThrow(() -> new UsernameNotFoundException("Not found: " + userName));


        return user.map(MyUserDetails::new).get();
    }
}
```

Application.properties:

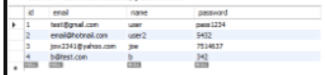
```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.naming-strategy=org.hibernate.cfg.ImprovedNamingStrategy
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/adding
spring.datasource.username=root
spring.datasource.password=rthLogan123*



logging.level.org.springframework.web: DEBUG
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
server.port=8090
```

server.error.whitelabel.enabled=false

Output Screenshot:

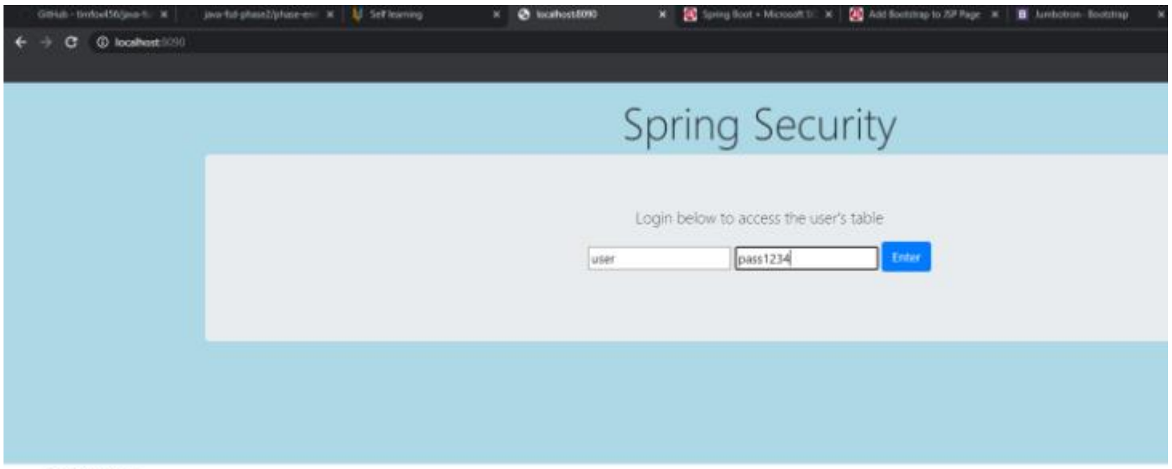


List of valid usernames/passwords

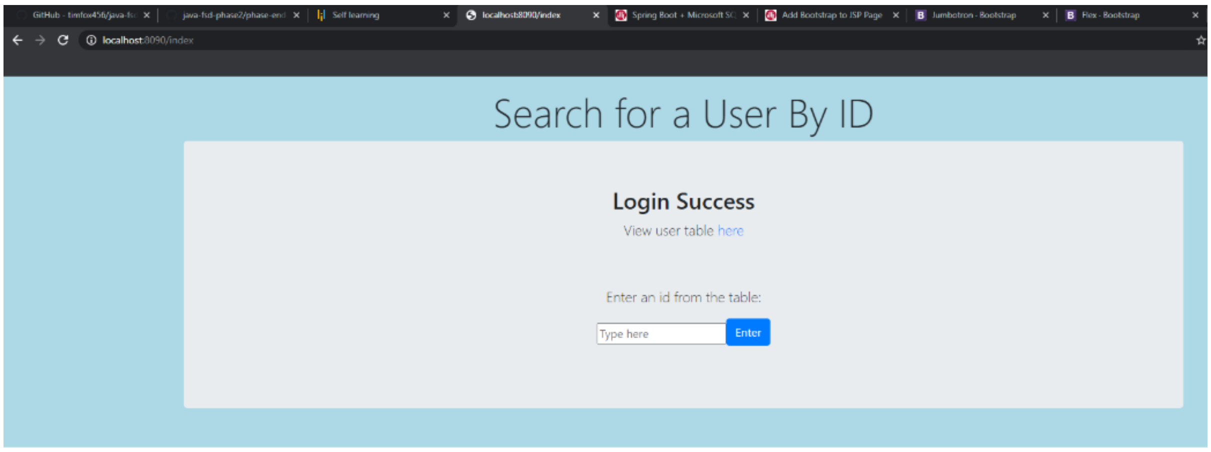


id	email	name	password
1	test@gmail.com	user	pass1234
2	email@hotmail.com	user2	5432
3	jow2341@yahoo.com	joe	7514637
4	b@test.com	b	342
NULL	NULL	NULL	NULL

Logging in with valid credentials



Successful login

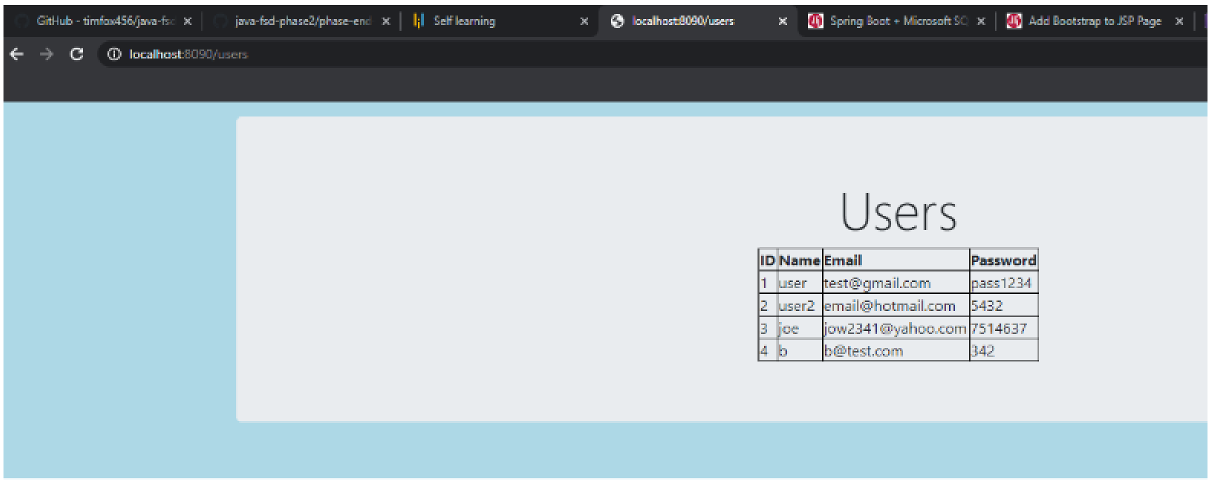

Search for a User By ID

Login Success

View user table here

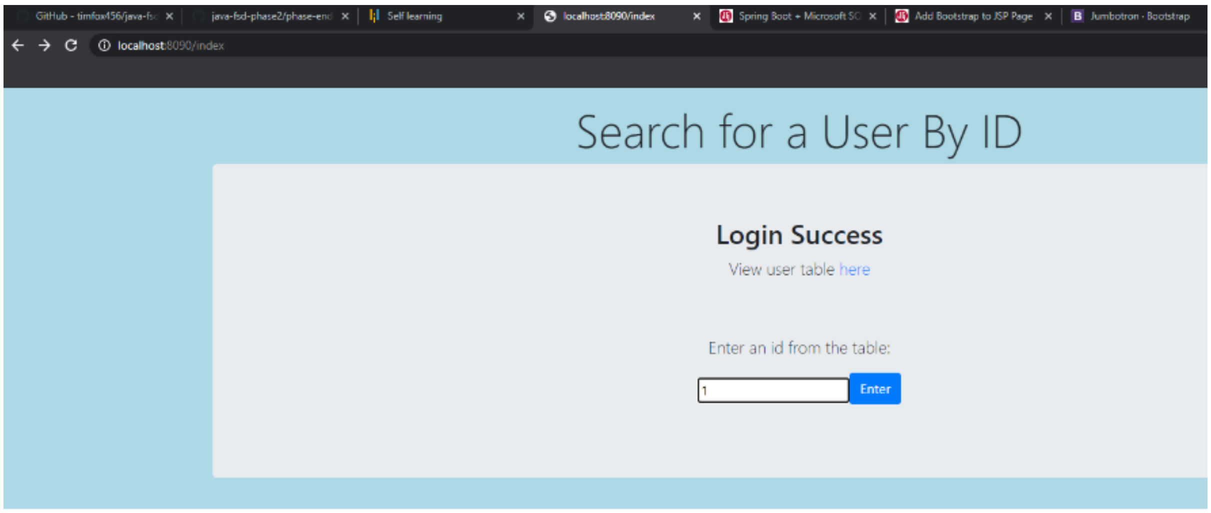
Enter an id from the table:

Prints out table from database of all users


Users

ID	Name	Email	Password
1	user	test@gmail.com	pass1234
2	user2	email@hotmail.com	5432
3	joe	jow2341@yahoo.com	7514637
4	b	b@test.com	342

Edit user with id 1


Search for a User By ID

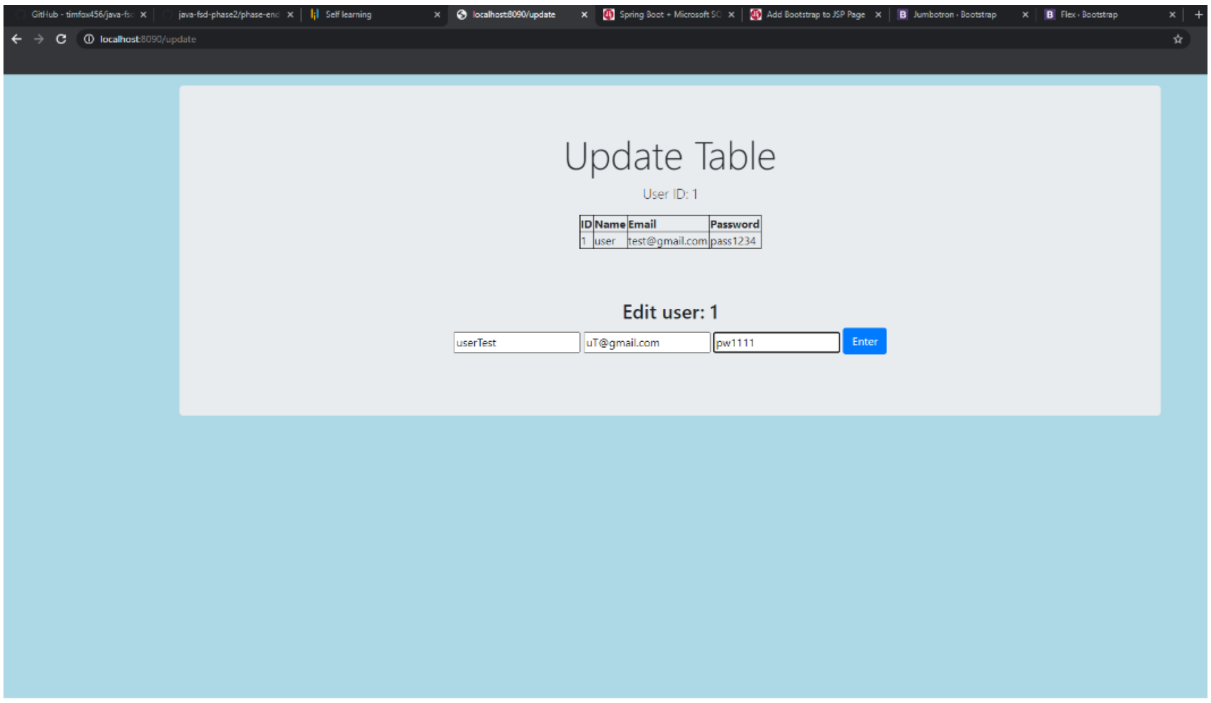
Login Success

View user table here

Enter an id from the table:

1

change user 1's information



Update Table

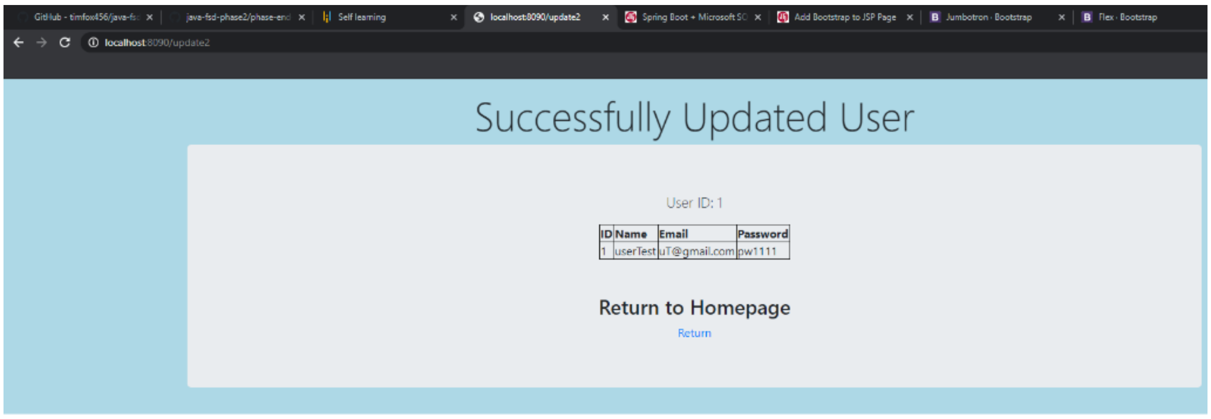
User ID: 1

ID	Name	Email	Password
1	user	test@gmail.com	pass1234

Edit user: 1

userTest | uT@gmail.com | pw1111

User 1 is updated in the table



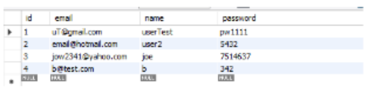
Successfully Updated User

User ID: 1

ID	Name	Email	Password
1	userTest	uT@gmail.com	pw1111

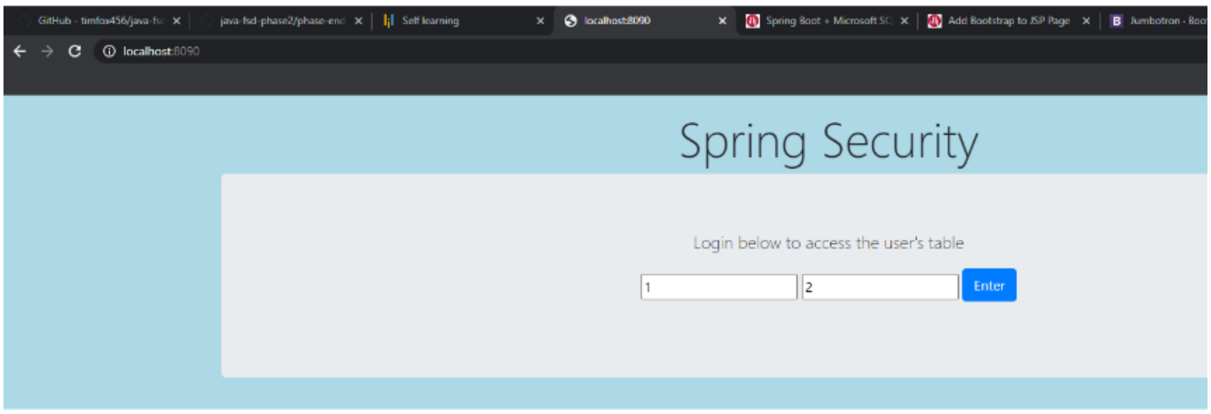
Return to Homepage

MySQL table also updated



id	email	name	password
1	uT@gmail.com	userTest	pw1111
2	email@hotmail.com	user2	5432
3	jow2341@yahoo.com	joe	7514637
4	b@test.com	b	342
NULL	NULL	NULL	NULL

Invalid Credentials

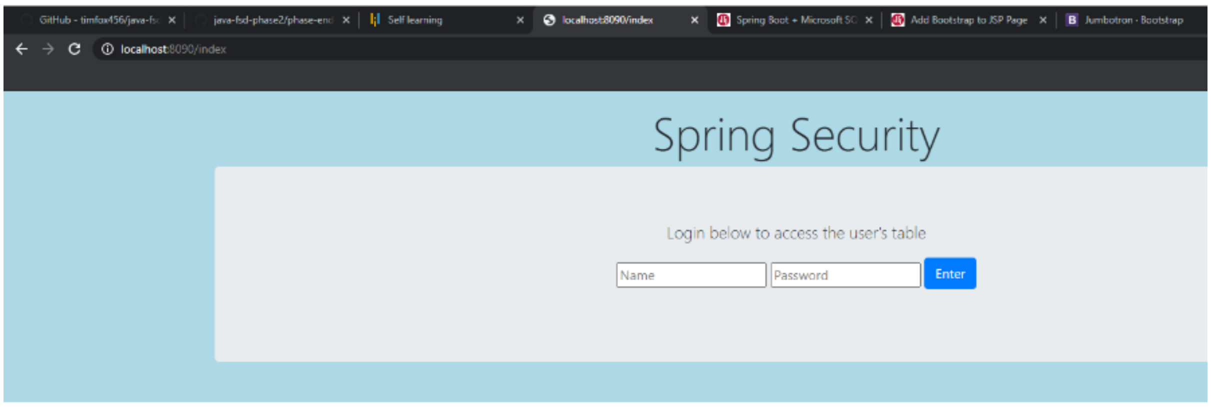

GitHub - timfox456/java-fsc
java-fsd-phase2/phase-end
Self learning
localhost:8090
Spring Boot + Microsoft SC
Add Bootstrap to JSP Page
Jumbotron - Boo
localhost:8090

Spring Security

Login below to access the user's table

1 | 2 | Enter

user is redirected to home page (notice address bar says index)


GitHub - timfox456/java-fsc
java-fsd-phase2/phase-end
Self learning
localhost:8090/index
Spring Boot + Microsoft SC
Add Bootstrap to JSP Page
Jumbotron - Bootstrap
localhost:8090/index

Spring Security

Login below to access the user's table

Name | Password | Enter