```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3
306/db_example
spring.datasource.username=user
spring.datasource.password=Password


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

```
package com.example.Authentication;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication
;

@SpringBootApplication
public class AuthenticationApplication {

    public static void main(String[] args) {

    SpringApplication.run(AuthenticationApplication.class,
args);
    }

}
```

```
package com.example.Authentication;


import com.example.Authentication.entities.User;
import
com.example.Authentication.repositories.UserRepository;
import com.example.Authentication.services.UserService;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assertions.assertEquals;


public class EntityTests {

	@Test
	public void WhenSetPassword_CheckGetPassword() {
		User testUser = new User();

		testUser.setPassword("mypassword");
		assertEquals(testUser.getPassword(),"mypassword");
	}

	@Test
	public void WhenSetName_CheckGetName() {
		User testUser = new User();

		testUser.setName("name");
		assertEquals(testUser.getName(),"name");
	}

	@Test
	public void WhenSetEmail_CheckGetEmail() {
		User testUser = new User();

		testUser.setEmail("email@email.com");

	assertEquals(testUser.getEmail(),"email@email.com");
	}
}
```

```
package com.example.Authentication;

import
com.example.Authentication.controllers.LoginController;
import com.example.Authentication.entities.User;
import com.example.Authentication.services.UserService;

import org.junit.jupiter.api.Test;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.web.server.LocalServerPort;
import org.springframework.test.web.servlet.MockMvc;
import
org.springframework.boot.test.autoconfigure.web.servlet.Auto
ConfigureMockMvc;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.assertj.core.api.Assertions.assertThat;
import static org.hamcrest.Matchers.containsString;
import static
org.springframework.test.web.servlet.request.MockMvcRequestB
uilders.get;
import static
org.springframework.test.web.servlet.result.MockMvcResultHan
dlers.print;
import static
org.springframework.test.web.servlet.result.MockMvcResultMat
chers.content;
import static
org.springframework.test.web.servlet.result.MockMvcResultMat
chers.status;

import java.util.Optional;


@SpringBootTest(webEnvironment =
SpringBootTest.WebEnvironment.RANDOM_PORT)
@AutoConfigureMockMvc
```

```
public class AuthenticationWebTests {



    @LocalServerPort
    private int port;

    @Autowired
    private MockMvc mockMvc;


    @Test
    public void shouldReturnDefaultMessage() throws
Exception {

this.mockMvc.perform(get("/")).andDo(print()).andExpect(stat
us().isOk());
    }




}
```

```
package com.example.Authentication;


import com.example.Authentication.entities.User;
import
com.example.Authentication.repositories.UserRepository;
import com.example.Authentication.services.UserService;
import org.junit.jupiter.api.Test;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaT
est;
import
org.springframework.boot.test.autoconfigure.orm.jpa.TestEnti
tyManager;
import org.springframework.boot.test.context.SpringBootTest;
import org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assertions.assertEquals;

import java.util.Optional;



@DataJpaTest

public class AuthenticationTests {

    @Autowired
    private TestEntityManager entityManager;

    @Autowired
    private UserRepository userRepository;


    @Test
    public void whenFindByName_thenReturnUser() {
        // given
        User dummyUser = new User();
        dummyUser.setName("Dummy");
        dummyUser.setEmail("test@test.com");
        dummyUser.setPassword("password");
        entityManager.persist(dummyUser);
        entityManager.flush();

        // when
        User found =
userRepository.findByName(dummyUser.getName());

        // then

        assertEquals(found.getName(), dummyUser.getName());
    }

    @Test
    public void whenFindById_thenReturnUser() {
     User dummyUser = new User();
        dummyUser.setName("Dummy");
        dummyUser.setEmail("test@test.com");
        dummyUser.setPassword("password");
        entityManager.persist(dummyUser);
        entityManager.flush();

        Optional<User> found =
userRepository.findById(dummyUser.getId());

        assertEquals(found.get().getName(),
dummyUser.getName());
    }

    @Test
    public void whenFindByName_thenReturnpassword() {
        // given

        User dummyUser = new User();
        dummyUser.setName("Dummy");
        dummyUser.setEmail("test@test.com");
        dummyUser.setPassword("password");
        entityManager.persist(dummyUser);
        entityManager.flush();

        // when
        User found =
userRepository.findByName(dummyUser.getName());

        // then

        assertEquals(found.getPassword(),
dummyUser.getPassword());
    }

    @Test
    public void whenFindById_thenReturnpassword() {
     User dummyUser = new User();
        dummyUser.setName("Dummy");
        dummyUser.setEmail("test@test.com");
        dummyUser.setPassword("password");
        entityManager.persist(dummyUser);
        entityManager.flush();

        Optional<User> found =
userRepository.findById(dummyUser.getId());

        assertEquals(found.get().getPassword(),
dummyUser.getPassword());
    }




}
```

```
package com.example.Authentication;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static
org.junit.jupiter.api.Assertions.assertNotEquals;

import java.util.Optional;

import org.junit.jupiter.api.Test;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import com.example.Authentication.entities.User;
import com.example.Authentication.services.UserService;

@SpringBootTest
public class ServiceTest {

     @Autowired
     private UserService service;

     @Test
    public void whenGetAllUsers_thenReturnCountNotZero() {
     Iterable<User> users = service.GetAllUsers();
     int count = 0;

     for(User user : users) {
          count++;
     }

     assertNotEquals(count, 0);
    }

     @Test
     public void whenGetUsersByName_thenReturnUser() {
          User users = service.GetUserByName("moss");

          assertEquals(users.getName(), "moss");
    }

    @Test
    public void whenGetUserById_thenReturnUser() {
        User users = service.GetUserById(1);

        assertEquals(users.getName(), "moss");
    }

    @Test
    public void whenUpdateUser_thenReturntheUpdate() {
        User users = service.GetUserByName("moss");
        users.setEmail("coolio@gmail.com");
        //Did it this way on purpose, so I did not get
persistence in my actual DB
        //System.out.println("------------------------------
----------" + users.getEmail());

        assertEquals(users.getEmail(), "coolio@gmail.com");

    }
}
```

```java
package com.example.Authentication;

import org.junit.jupiter.api.Test;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import static org.assertj.core.api.Assertions.assertThat;
import
com.example.Authentication.controllers.LoginController;

@SpringBootTest
class AuthenticationApplicationTests {

	@Autowired
	private LoginController controller;

	@Test
	void contextLoads() {
		assertThat(controller).isNotNull();
	}

}
```

```
package com.example.Authentication.services;

import java.util.Optional;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.Authentication.entities.User;
import
com.example.Authentication.exceptions.UserNotFoundException;
import
com.example.Authentication.repositories.UserRepository;




@Service
public class UserService {

     @Autowired
      private UserRepository userRepository;

     Logger logger =
LoggerFactory.getLogger(UserService.class);

    public Iterable<User> GetAllUsers()
    {
//       logger.info("in getallusers------------------------
--------------------------------");
        return userRepository.findAll();
    }


    public User GetUserByName(String name) {
        User foundUser = userRepository.findByName(name);
        return foundUser;
    }

    public User GetUserById(int id) {
     Optional<User> foundUser = userRepository.findById(id);


     //TODO: we need to decide how to handle a "Not Found"
condition

     if (!foundUser.isPresent()) {
         throw new UserNotFoundException();
     }

     return(foundUser.get());
    }

    public void UpdateUser(User usertoUpdate) {
     userRepository.save(usertoUpdate);
    }


}
```

```java
package com.example.Authentication.repositories;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.example.Authentication.entities.User;

@Repository
public interface UserRepository extends CrudRepository<User,
Integer> {

    public User findByName(String name);
//    public User findById(int id);
}
```

```java
package com.example.Authentication.exceptions;

public class UserNotFoundException extends RuntimeException
{
    private static final long serialVersionUID = 1L;
}
```

```java
package com.example.Authentication.entities;

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

@Entity // This tells Hibernate to make a table out of this
class
public class User {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;

    private String email;

    private String name;

    private String password;

    public String getPassword() {
        return password;
    }

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

    public Integer getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
     return (id.toString() + " " + name + " " + email + " " +
password);
    }

}
```

```
package com.example.Authentication.controllers;


import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import com.example.Authentication.entities.User;
import com.example.Authentication.services.UserService;


@Controller
public class LoginController {

     @Autowired
     private UserService serv;

    @GetMapping("/")
    public String showGreeting(ModelMap map) {
        return "index";
    }


    @GetMapping("/login")
    public String showLogin(ModelMap map) {
        return "login";
    }

    @PostMapping("/login")
    public String submitLogin(@RequestParam("username")
String username, @RequestParam("password") String password){

         String login = "";
         User person = serv.GetUserByName(username);
        //TODO:
         if(password.equals(person.getPassword())) {
              login = "Success";
         }else {
              login = "failed";
         }
        return login;



    }
}
```