

All project code can be found at the following github link:

<https://github.com/ludan1214/java-fsd-phase2/tree/main/LoginRegistration>

## 1. Project Setup

### ① Setting up the MySQL server

Docker was used to start up a MySQL server container on the localhost located at

<http://localhost:3306>

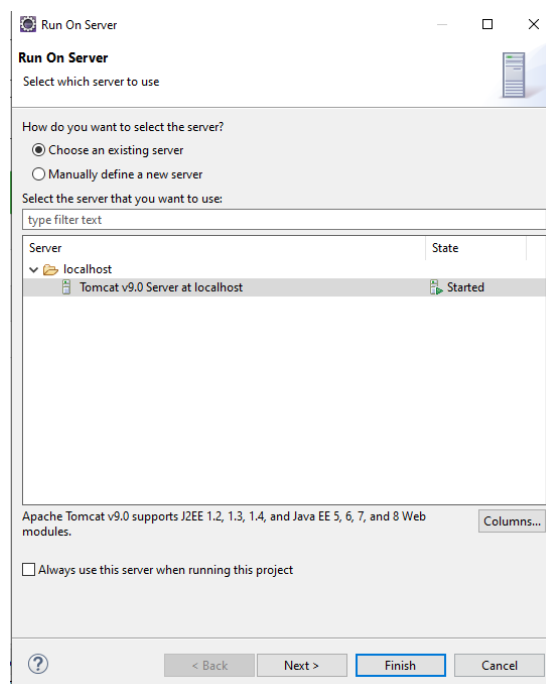
- `docker pull mysql/mysql-server:latest`
- `docker run -e MYSQL_ROOT_PASSWORD=password -d -p 3308:3306 mysql`

### ② Setting up the database

A table that contains the users' credentials is created.

- `CREATE TABLE IF NOT EXISTS `phase2db`.`user_tbl` (  
    `id` INT NOT NULL AUTO_INCREMENT,  
    `firstname` VARCHAR(45) NULL,  
    `lastname` VARCHAR(45) NULL,  
    `username` VARCHAR(45) NULL,  
    `password` VARCHAR(45) NULL,  
    PRIMARY KEY (`id`),  
    UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)  
ENGINE = InnoDB`

### ③ Setting up the Apache Web Server



## 2. Project Overview

In terms of language technologies, the LoginRegistration system uses Hibernate, Java Servlets, and JSP to accomplish its functions.

1. The User class is mapped to the database with Hibernate.

It contains the following info:

1. First Name
2. Last Name
3. User Name
4. Password

```
@javax.persistence.Entity(name = "user_tbl")
public class User {
    private long id;
    private String firstname;
    private String lastname;
    private String username;
    private String password;

    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }

    @Column(name = "firstname")
    public String getFirstname() {
        return firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    @Column(name = "lastname")
    public String getLastname() {
        return lastname;
    }
}
```

```

    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    @Column(name = "username")
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @Column(name = "password")
    public String getPassword() {
        return password;
    }

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

```

2. The UserManager class implements the credential validation by using hibernate to make queries to the database. Instances of this class are created in the servlets to accomplish this. Primary functions include verifying the login credentials and registering a new user to the database.

```

public class UserManager {

    private static StandardServiceRegistry registry;

    private static SessionFactory sessionFactory;

    public UserManager() {
        setup();
    }

    protected void setup() {
        try {
            // Creating a registry
            registry = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

            // Create the MetadataSources

```

```

MetadataSources sources = new MetadataSources(registry);

// Create the Metadata
Metadata metadata = sources.getMetadataBuilder().build();

// Create SessionFactory
sessionFactory = metadata.getSessionFactoryBuilder().build();

} catch (Exception e) {
    e.printStackTrace();
    if (registry != null) {
        StandardServiceRegistryBuilder.destroy(registry);
    }
}
}

protected void exit() {
    // code to close Hibernate Session factory
    sessionFactory.close();
}

protected boolean verifyCredentials(String username) {
    Session session = sessionFactory.openSession();
    Criteria criteria = session.createCriteria(User.class);
    User user = (User) criteria.add(Restrictions.eq("username", username))
        .uniqueResult();
    session.close();
    if (user != null) {
        return false;
    }
    return true;
}

protected boolean register(String first, String last, String username, String pass) {
    User user = new User();
    if (!verifyCredentials(username)) {
        return false;
    }
    user.setFirstname(first);

```

```

        user.setLastname(last);
        user.setUsername(username);
        user.setPassword(pass);

        Session session = sessionFactory.openSession();
        session.beginTransaction();

        session.save(user);

        session.getTransaction().commit();
        session.close();
        return true;
    }

    @protected User login(String username, String pass) {
        Session session = sessionFactory.openSession();
        Criteria criteria = session.createCriteria(User.class);
        User user = (User) criteria.add(Restrictions.eq("username",
username)).add(Restrictions.eq("password",pass))
            .uniqueResult();
        session.close();
        return user;
    }
}

```

### 3. There are four servlets.

#### 1. SendLoginDetails

Handles the HTML POST request from the login.jsp webpage.

If the credentials are found in the database, an HttpSession will be started, otherwise an error message is displayed

#### 2. SendRegisterDetails

Handles the HTML POST request from the register.jsp webpage,

If the username exists in the database, an error message will appear.

#### 3. Dashboard

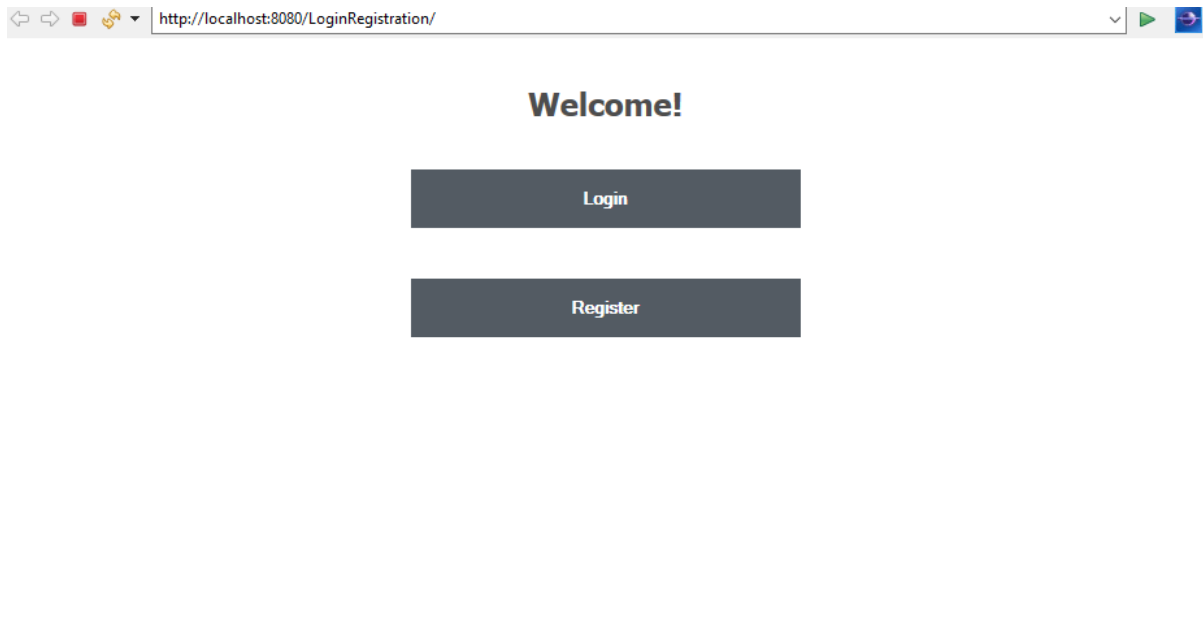
An endpoint only accessible if the user has logged in with valid credentials and started an HttpSession, otherwise the user is redirected to the login page.

#### 4. Logout

This destroys the current HttpSession and returns the user to the index.jsp page where they can decide to login or register.

### 3. Project Demo

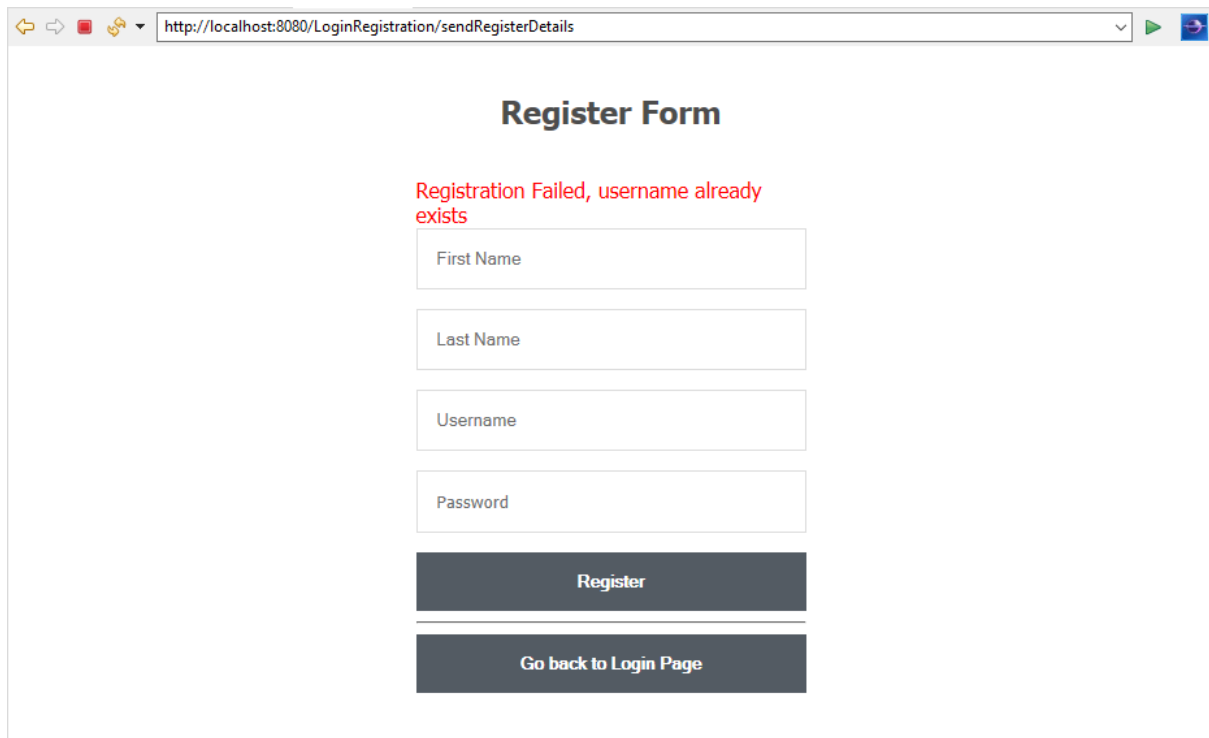
When visiting the homepage the user is met with the option to login or register an account.



Clicking the Register button will bring the user to the registration form

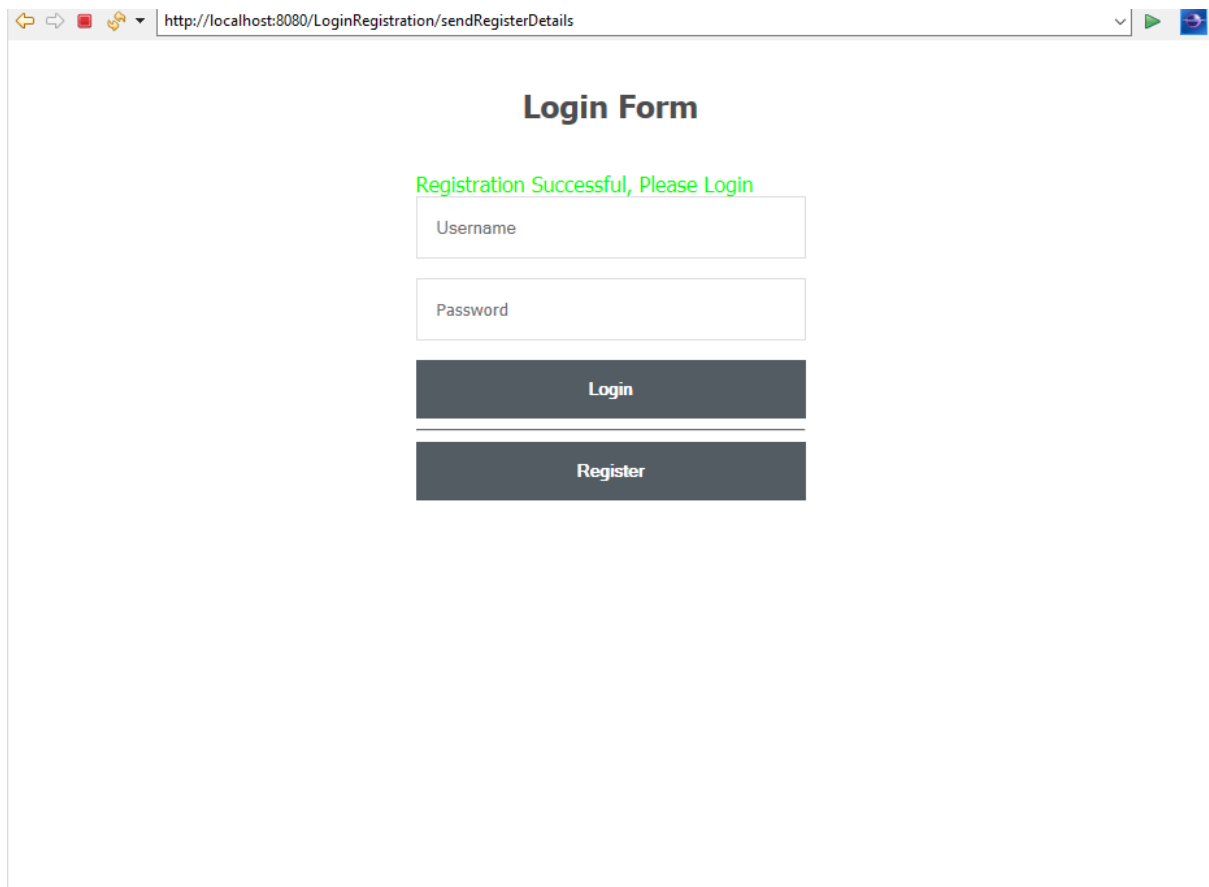
A screenshot of a web browser window showing the 'register.jsp' page. The page has a white background and a dark grey header bar. Below the header, the text 'Register Form' is displayed in a bold, black font. The form consists of four input fields stacked vertically: the first contains 'Luc', the second 'Le', the third 'lucle', and the fourth is a password field represented by ten black dots. Below the input fields are two dark grey rectangular buttons with white text: 'Register' and 'Go back to Login Page', stacked vertically. The browser's address bar shows 'http://localhost:8080/LoginRegistration/register.jsp'.

If the user already exists we will see an error



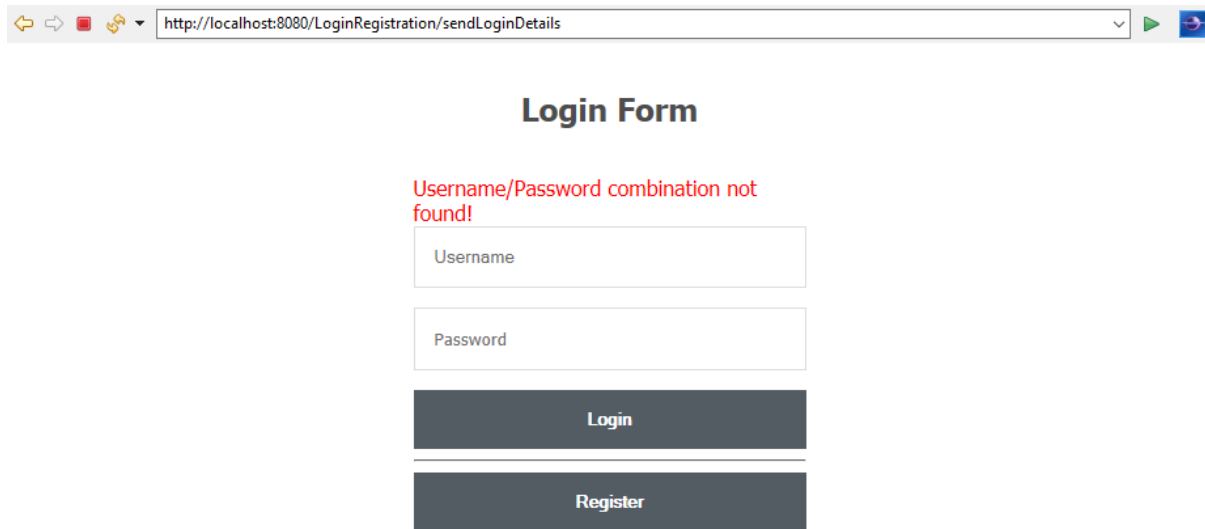
A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/LoginRegistration/sendRegisterDetails`. The page title is "Register Form". Below the title, a red error message reads "Registration Failed, username already exists". The form contains four input fields: "First Name", "Last Name", "Username", and "Password". Below these fields are two dark grey buttons: "Register" and "Go back to Login Page".

Otherwise if successful we will be redirected to the login page and a success message is displayed.



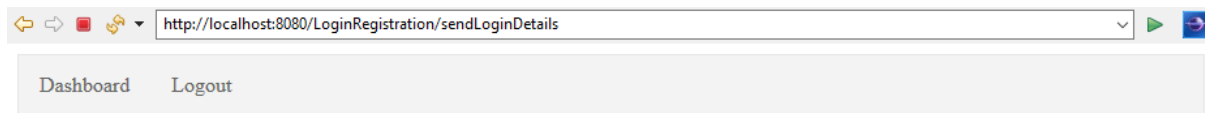
A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/LoginRegistration/sendRegisterDetails`. The page title is "Login Form". Below the title, a green success message reads "Registration Successful, Please Login". The form contains two input fields: "Username" and "Password". Below these fields are two dark grey buttons: "Login" and "Register".

If we enter incorrect credentials on the login form, an error message is displayed.



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/LoginRegistration/sendLoginDetails`. The page title is "Login Form". Below the title, there is a red error message: "Username/Password combination not found!". Underneath the error message, there are two input fields: "Username" and "Password". Below these fields are two buttons: "Login" and "Register".

Otherwise if the credentials are correct, we will be redirected to the dashboard



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/LoginRegistration/sendLoginDetails`. The page has a header bar with two links: "Dashboard" and "Logout". Below the header bar, the text "Hello, Luc!" is displayed.

**Hello, Luc!**

The name "Luc" is retrieved from the User object passed in via the "user" attribute in the SendLoginRequest servlet, the object is then retrieved and used in the Dashboard.jsp file, a `getFirstName()` call on this User object is used to retrieve the first name.



Finally if the logout button is selected the user is logged out and sent to the index page. And a message is displayed.



## Welcome!

You have been logged out

Login

Register

If the user tries to access the dashboard again the user is redirected to the login page and an error message is displayed.



## Login Form

You must log in!

Username

Password

Login

Register