



SKILL  LYNC

Internship Project On

REGISTRATION AND LOGIN

PRESENTED BY,
SHARATH P

ABSTRACT

Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by pivotal team and is used to build stand-alone and production ready spring applications. Let us try to learn how to develop a Java web application from scratch with the essential features: user registration, login, logout and view users list – with user information stored in MySQL database. we will go through the process of coding a Spring Boot project.

SOFTWARE TECHNOLOGIES

- Java Eclipse IDE
- Java Development Kit (JDK)
- MySQL Community server and MySQL Workbench
- Spring Web MVC for the web layer
- Spring Data JPA with Hibernate framework or the data access layer
- Spring Security for authentication, login and logout
- Thymeleaf as template engine
- HTML 5 and Bootstrap 4 for responsive user interface
- MySQL database



HARDWARE USED

- RAM 16 GB
- Processor 64-bit, 2 core.
- Hard drive space required 2GB for installation

What is Micro Service?

Micro Service is an architecture that allows the developers to develop and deploy services independently. Each service running has its own process and this achieves the lightweight model to support business applications.

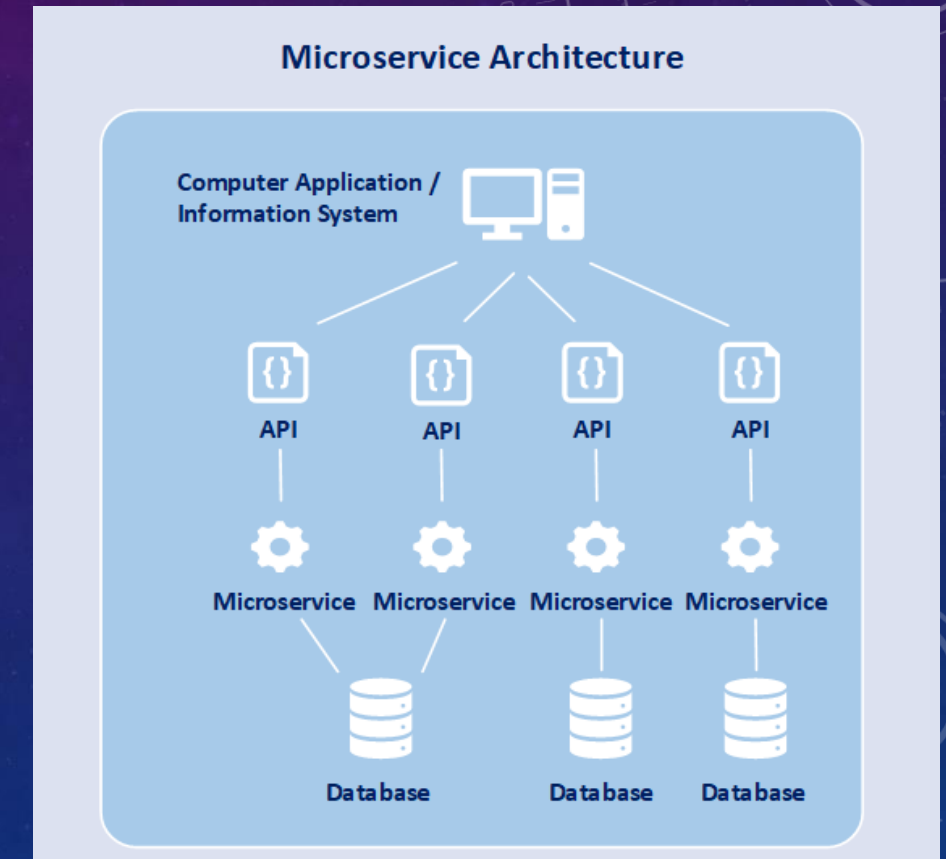
Advantages:

Micro services offers the following advantages to its developers –

- Easy deployment
- Simple scalability
- Compatible with Containers
- Minimum configuration
- Lesser production time

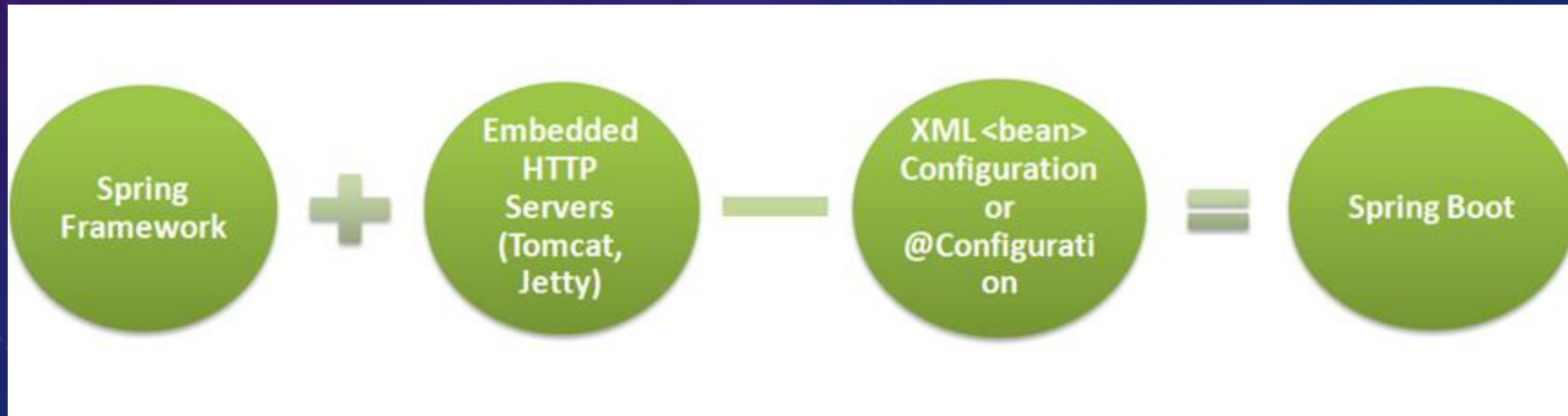
Disadvantages:

- Difficult to manage a large number of services.
- The developer needs to solve the problem, such as network latency and load balancing.
- Complex testing over a distributed environment.



What is Spring Boot?

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can **just run**. You can get started with minimum configurations without the need for an entire Spring configuration setup.



Advantages:

Spring Boot offers the following advantages to its developers –

- Easy to understand and develop spring applications
- Increases productivity
- Reduces the development time

Goals:

Spring Boot is designed with the following goals –

- To avoid complex XML configuration in Spring
- To develop a production ready Spring applications in an easier way
- To reduce the development time and run the application independently
- Offer an easier way of getting started with the application

Why Spring Boot?

You can choose Spring Boot because of the features and benefits it offers as given here –

- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides a powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto configured; no manual configurations are needed.
- It offers annotation-based spring application
- Eases dependency management
- It includes Embedded Servlet Container

How does it work?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using **@EnableAutoConfiguration** annotation. For example, if MySQL database is on your classpath, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

The entry point of the spring boot application is the class contains **@SpringBootApplication** annotation and the main method.

Spring Boot automatically scans all the components included in the project by using **@ComponentScan** annotation.

Examples:

Look at the following Spring Boot starters explained below for a better understanding –

- **Spring Boot Starter Actuator dependency** is used to monitor and manage your

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>
```

- **Spring Boot Starter Security dependency** is used for Spring Security.
- **Spring Boot Starter web dependency** is used to write a Rest Endpoints.
- **Spring Boot Starter Test dependency** is used for writing Test cases.

Auto Configuration:

Spring Boot Auto Configuration automatically configures your Spring application based on the JAR dependencies you added in the project. For example, if MySQL database is on your class path, but you have not configured any database connection, then Spring Boot auto configures an in-memory database.

For this purpose, you need to add **@EnableAutoConfiguration** annotation or **@SpringBootApplication** annotation to your main class file. Then, your Spring Boot application will be automatically configured.

Observe the following code for a better understanding –

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
@EnableAutoConfiguration
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```


Spring Boot Application:

The entry point of the Spring Boot Application is the class contains **@SpringBootApplication** annotation. This class should have the main method to run the Spring Boot application. **@SpringBootApplication** annotation includes Auto-Configuration, Component Scan, and Spring Boot Configuration.

If you added **@SpringBootApplication** annotation to the class, you do not need to add the **@EnableAutoConfiguration**, **@ComponentScan** and **@SpringBootConfiguration** annotation. The **@SpringBootApplication** annotation includes all other annotations.

Observe the following code for a better understanding –

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```


Spring Boot - Bootstrapping

- **Spring_INITIALIZER:**

One of the ways to Bootstrapping a Spring Boot application is by using Spring_INITIALIZER. To do this, you will have to visit the Spring_INITIALIZER web page www.start.spring.io and choose your Build, Spring Boot Version and platform. Also, you need to provide a Group, Artifact and required dependencies to run the application.

Once you provided the Group, Artifact, Dependencies, Build Project, Platform and Version, click **Generate Project** button. The zip file will download and the files will be extracted.

- Maven
- Class Path Dependencies
- Main Method

STEPS FOR CREATE THE PROJECT :

1. Create Spring Boot Project and Configure Dependencies:

In Spring Tool Suite, create a new Spring Starter project with type Maven and language Java. And choose these dependencies: Spring Web, Thymeleaf, Spring Data JPA, MySQL Driver, Spring Security and Spring Boot DevTools – so the XML code for these dependencies in the pom.xml file.

2. Create Database and Configure Data Source:

- Use MySQL Workbench or MySQL Command Line Client program to create a new database named springdb(you can choose any name you want):
- Then open the Spring Boot configuration file application.properties under /src/main/resources directory. Enter the following properties for configuring a data source that will be used by Spring Data JPA:
- Note that we set the spring.jpa.hibernate.ddl-auto to create in order to let Hibernate create the tables when we run a unit test in the next section. Other properties are self-explanatory.

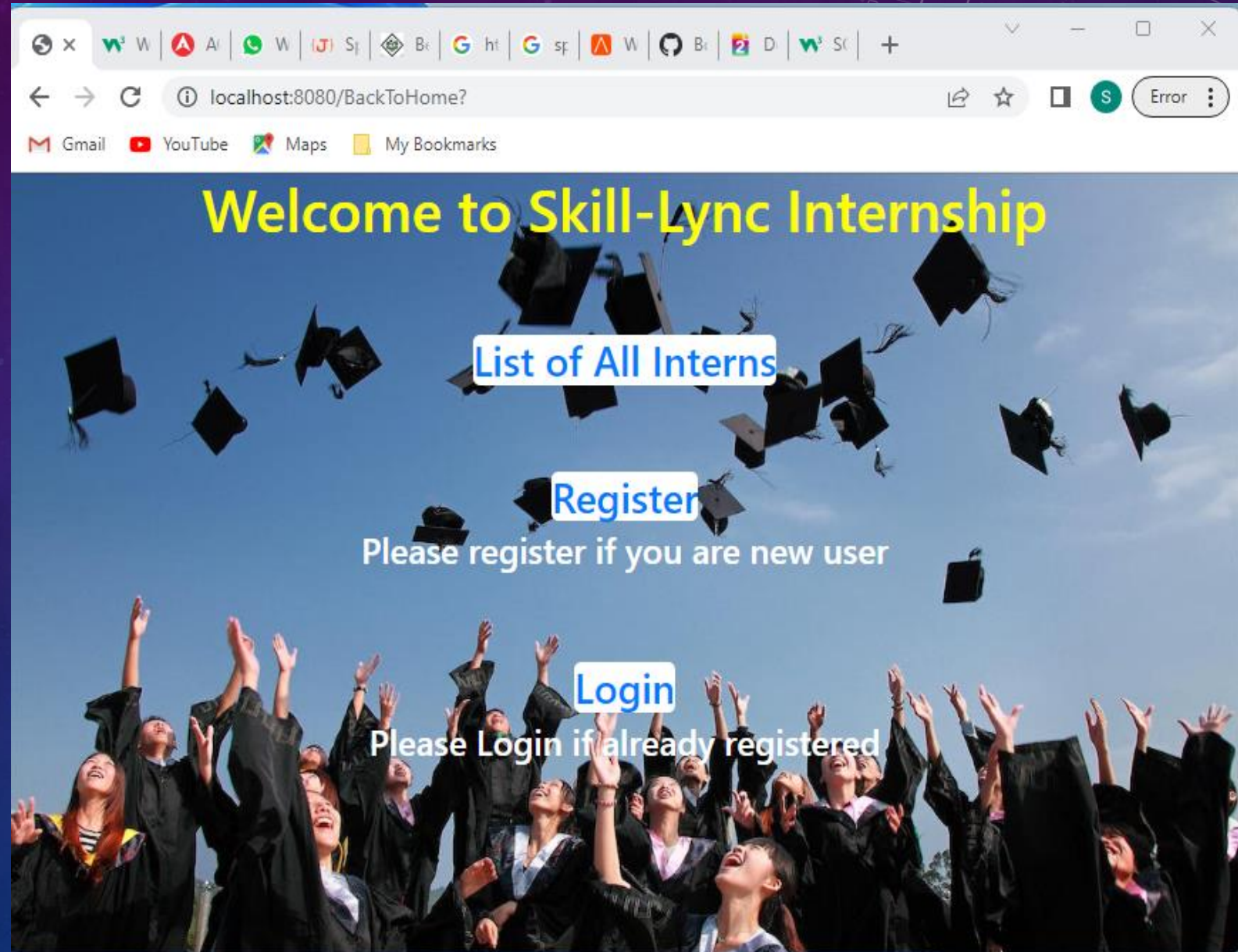
3. Code Entity Class and Repository Interface

- Next, create a new Java class named `User` to map with the corresponding `users` table (not yet created) in the database.
- As you can see, the user information consists of ID, email, password, first name and last name. Here I use common annotations from JPA.
- The setters and getters are now shown for brevity, so be sure you generate those methods as well.
- Next, create a new interface named `UserRepository` to act as a Spring Data JPA repository
- This interface is a subtype of `JpaRepository` which defines common persistence operations (including CRUD) and the implementation will be generated at runtime by Spring Data JPA.

4. Code Controller class and Home Page

- Next, let's create a Spring MVC controller class named `AppController`, with the first handler method to show the home page.
- Under `/src/main/resources/templates` directory, create a new HTML file named `index.html`.
- As you can see, in this webpage we use Bootstrap and JQuery from Webjars, so you must add the following dependencies for the project:
- You can also notice Thymeleaf is used to generate the URLs properly.

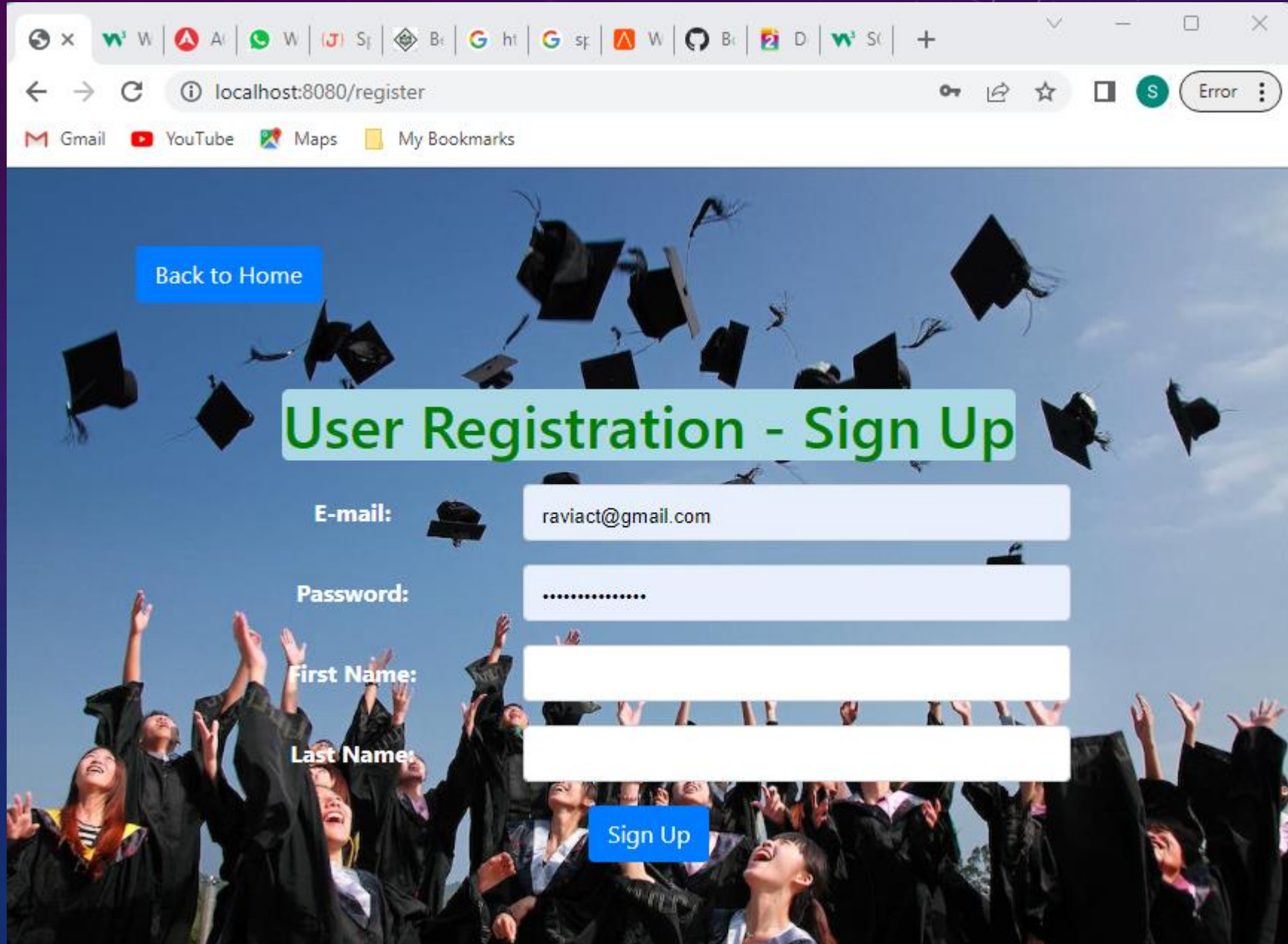
- Now, you can run this Spring Boot Project (using eclipse IDE), and access the web application at this URL `http://localhost:8080`, you should see the homepage appears as shown below:
- You see, the home page shows 3 links List of Users, Register and Login. You will learn how to implement each function in the next few minutes.



5. Implement User Registration feature

- Add a new handler method in the controller class to show the user registration form (sign up).
- This handler method will be executed when a user clicks the Register hyperlink in the homepage.
- And write code for the user registration page as follows (create signup_form.html file):

- Click Register link in the homepage, you should see the registration page appears like this:
- It looks very nice, because of Bootstrap and HTML 5. You can also notice with HTML 5, the browser provides validation for input fields so you don't have to use JavaScript for that.
- Next, code a handler method in the controller class to process registration.



Back to Home

User Registration - Sign Up

E-mail: raviact@gmail.com

Password:

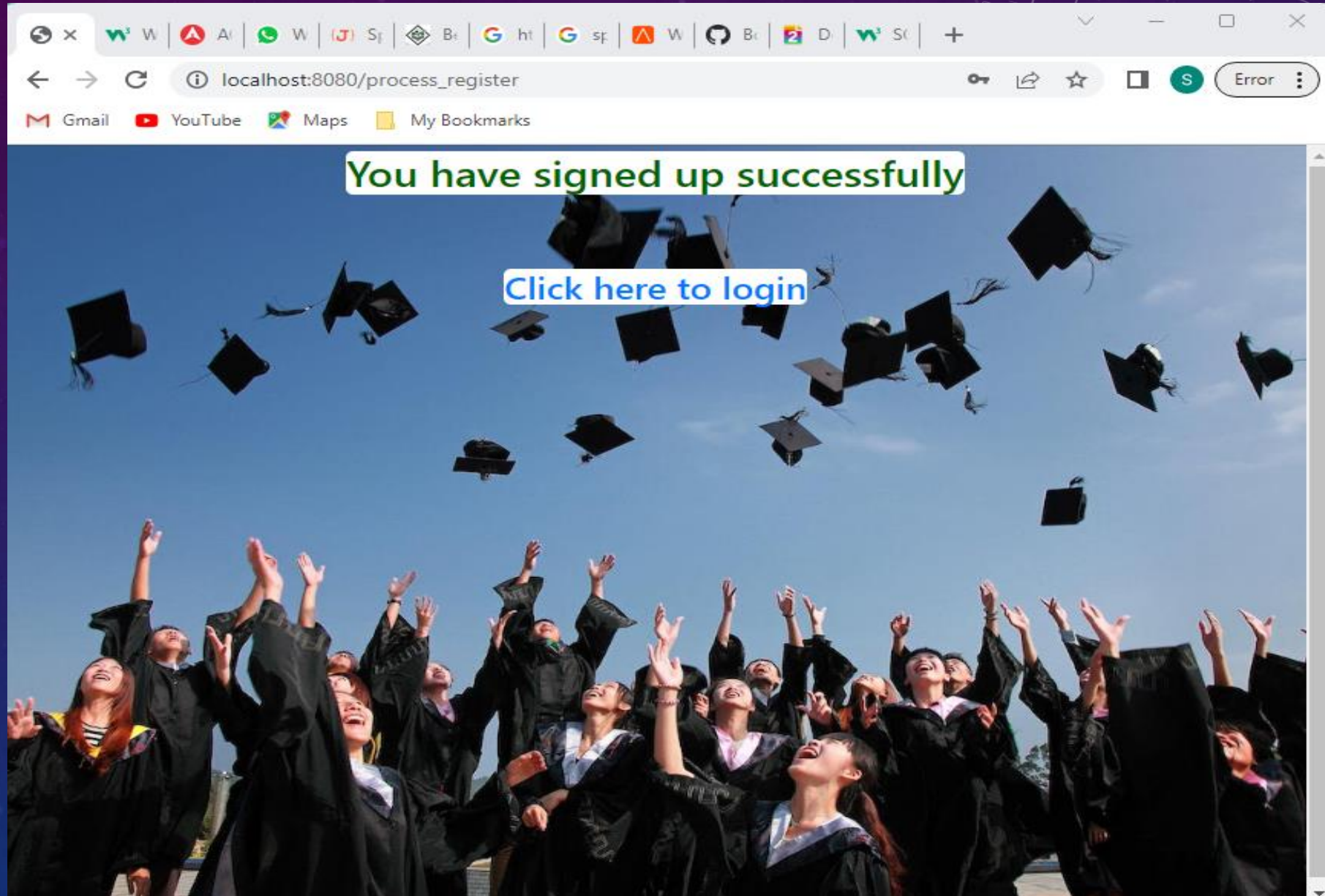
First Name:

Last Name:

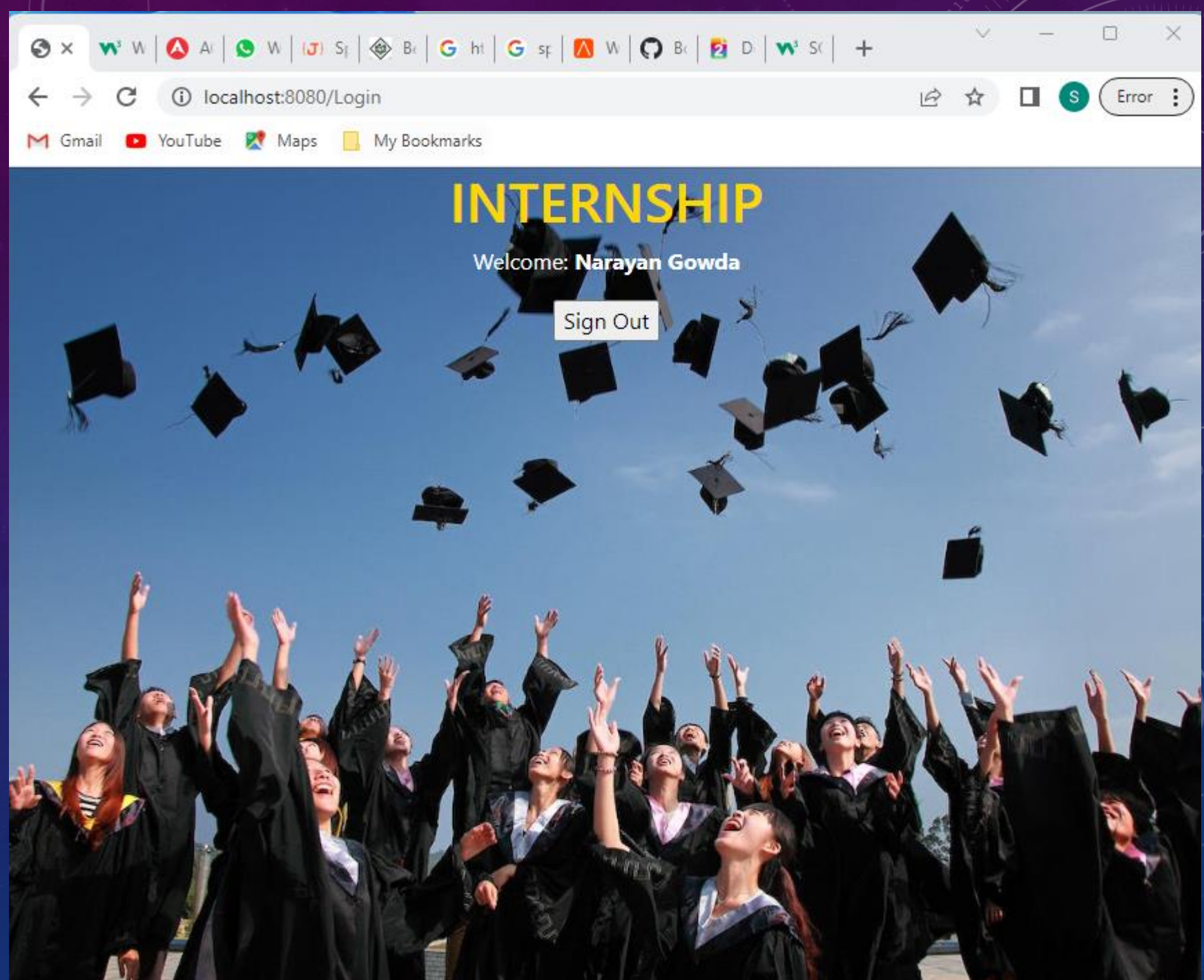
Sign Up

- As you can see, here we use `BCryptPasswordEncoder` to encode the user's password so the password itself is not stored in database (for better security) – only the hash value of the password is stored.
- After a `User` object is persisted into the database, it returns a logical view name `register_success`, so we need to create the corresponding HTML page.

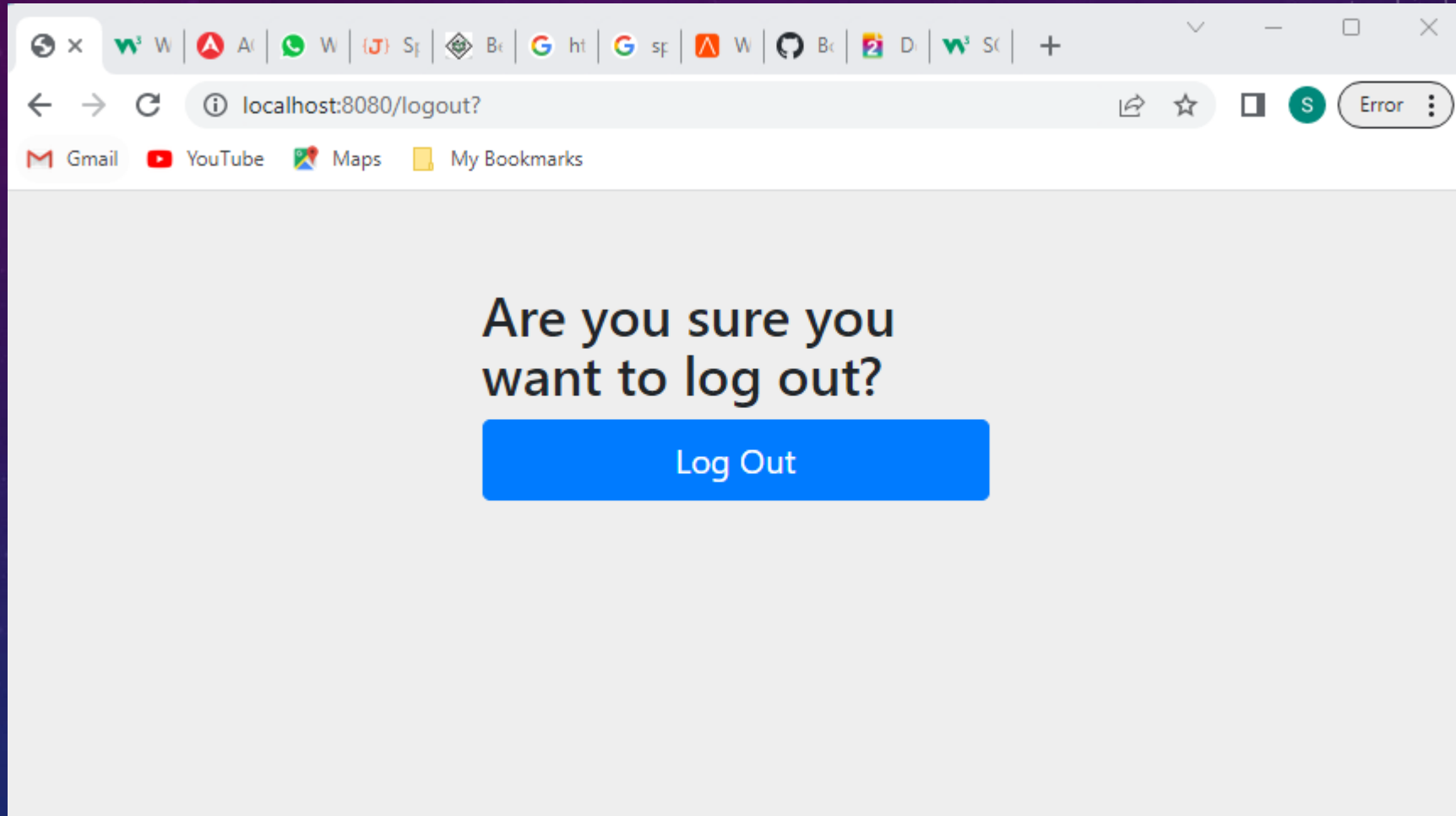
- This page simply displays the successful message after a user has registered, as shown below:
- Now you can test the user registration feature and verify result in the database (note that the password should be encoded).
- If we click the login link it will map to the individual user page.



- Let test adding more users and you will the list will contain more items. Click the Sign Out button, the application will log the user out and show the homepage.



- Once after clicking on Sign-out button you will redirect to the page as shown below:

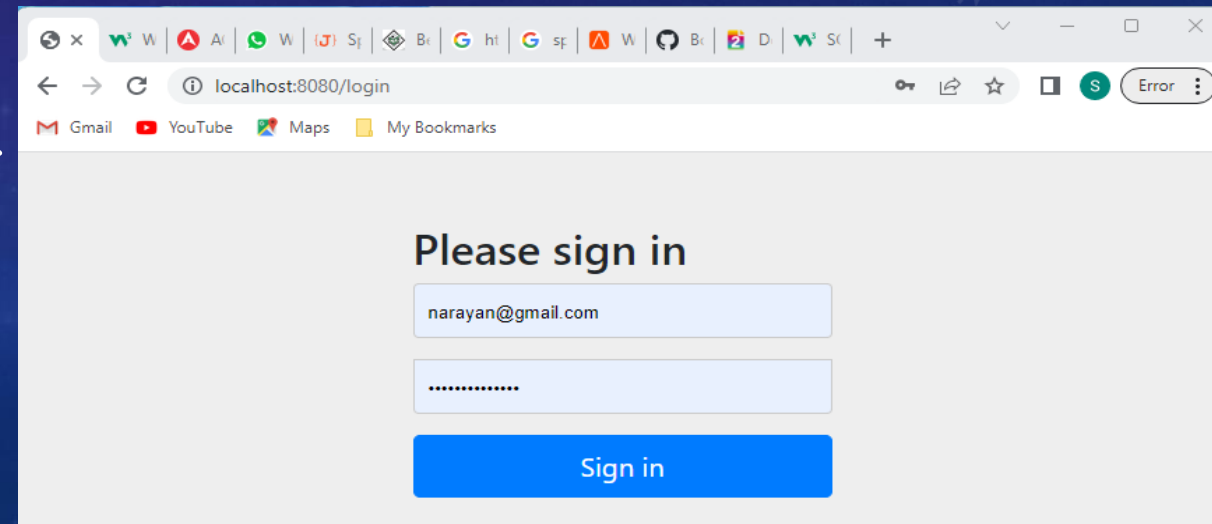


6. Code Custom UserDetails and UserDetailsService Classes

- Next, in order to implement authentication (login) feature, we need to create a class of subtype UserDetails (defined by Spring Security) to represent an authentication user.
- Spring Security will invoke methods in this class during the authentication process.
- And next, to tell Spring Security how to look up the user information, we need to code a class that implements the UserDetailsService interface.
- As you can see, Spring Security will invoke the loadUserByUsername() method to authenticate the user, and if successful, a new object of type CustomUserDetails object is created to represent the authenticated user.
- Also remember to update the UserRepository interface.
- Suppose that the email column is unique in the users table, so we define the findByEmail() method that returns a single User object based on email (no two users having the same email).

7. Configure Spring Security for Authentication (Login)

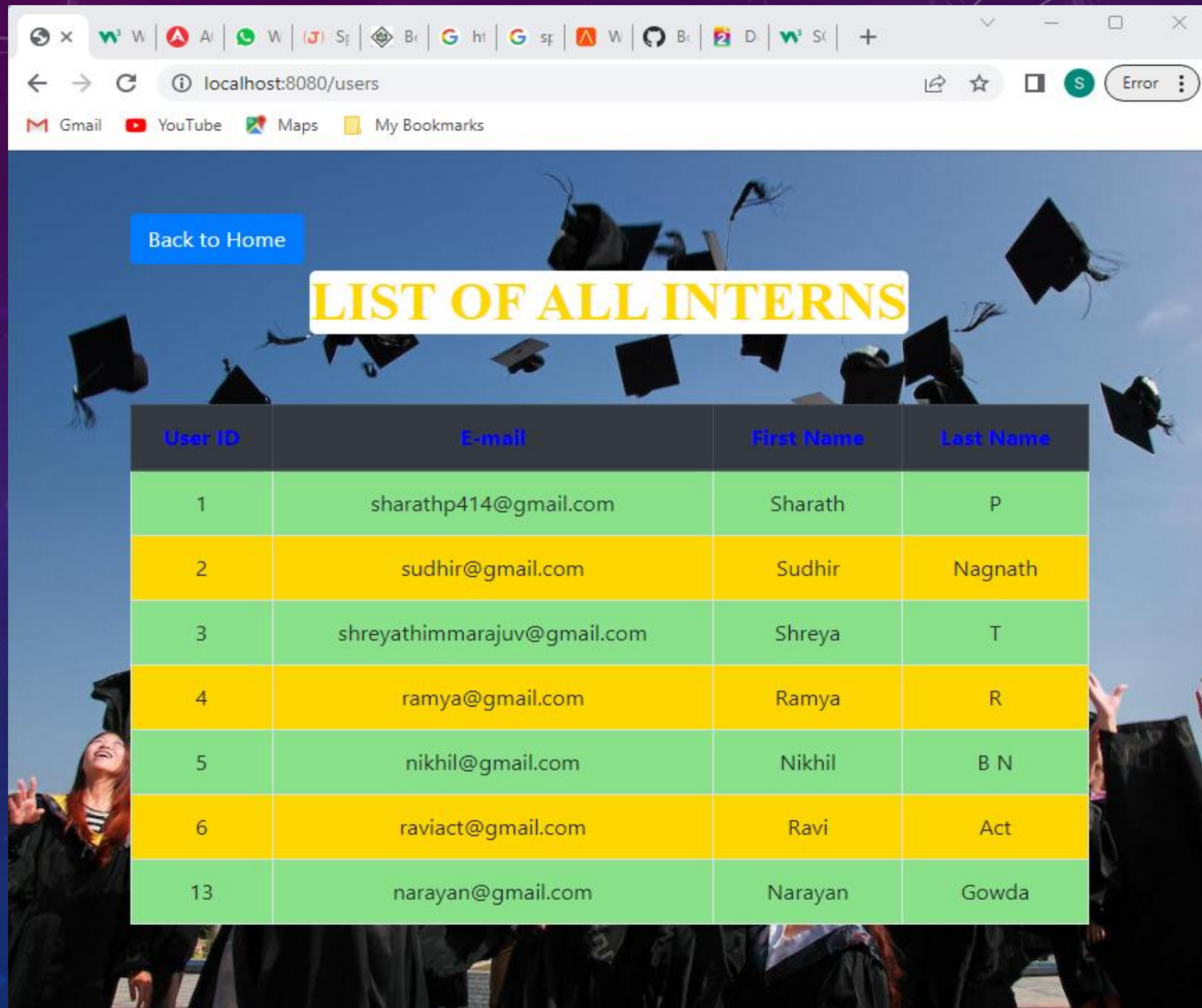
- Next, create a new Java class for configuring Spring Security.
- Here in the configure() method, a user must login to see the list users page (URL /users) and other pages do not require authentication. We also configure the default login page (generated by Spring Security) with the parameter name of the username field is email and the default success URL is /users – that means after successful login, the user will be redirected to the list users page.
- Now you can test the login function. Go to the homepage and click Login link, you should see the default login page appears as follows:
- Enter the username (email) and password of the user you have registered previously and click Sign in. You should see an error page because the list users page has not been implemented.



7. Code List Users Page and Logout

- Next, we're going to implement the list users and logout features. Update the controller class to have the following handler method:
- Here, you can see we call the `findAll()` method on the `UserRepository` but we didn't write that method. It is defined by the Spring Data JPA's `JpaRepository` interface.
- And create the `users.html` file with the following code:

- This page consists of two parts: the first part shows the user's full name with Logout button; and the second one lists all users in the database. It should look something like this:
- You can also use back to home button to go back to home page.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/users'. The page features a blue header with a 'Back to Home' button and a large yellow banner with the text 'LIST OF ALL INTERNS'. Below the banner is a table with four columns: User ID, E-mail, First Name, and Last Name. The table contains eight rows of data, alternating between green and yellow background colors for each row.

User ID	E-mail	First Name	Last Name
1	sharathp414@gmail.com	Sharath	P
2	sudhir@gmail.com	Sudhir	Nagnath
3	shreyathimmarajuv@gmail.com	Shreya	T
4	ramya@gmail.com	Ramya	R
5	nikhil@gmail.com	Nikhil	B N
6	raviact@gmail.com	Ravi	Act
13	narayan@gmail.com	Narayan	Gowda

CONCLUSION

This project is just to understand how to use spring boot to develop the simple features like user registration, login, logout/sign-out and view users list. And how we can connect to MySQL database using SQL driver dependencies, data access with the help of Spring data JPA. I also learnt how the authentication can be performed using Spring security, Thymeleaf, HTML 5 and Bootstrap 4 for responsive user interface.

Thank you