# Workshop

# Judge v2

Judge V2 is an application, which organizes the exercises in the Java DB course. In the courses, each homework is required to be in a zip format, which will be evaluated by your peers. The application will have an admin functionality. Only the administrators can add new exercises and give out admin roles to other users. The rest of the users can submit solutions to the specific exercise, as well as evaluate the solutions of your peers.

### Role

Create a Role class, which holds the following properties:

* **id** – UUID **String or Long**
* **name** – a **String**.
  + **USER and ADMIN** (add the two roles manually, when create DB)

### User

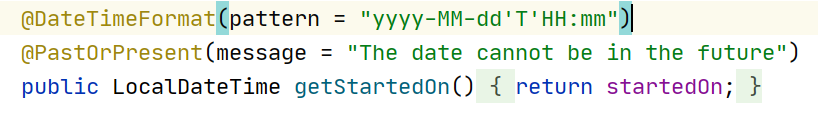
Create a User class, which holds the following properties:

* **id** – UUID **String or Long**
* username – a **String**.
  + username length must be minimum two characters**!**
* **password** – a **String**
  + password length must be minimum three characters**!**
* **email** – a **String**
  + email must contains '@'
* **git** – a **String**
  + git must be a valid github address in pattern: https:/github.com/{username}/SpringTestData/…
* **role** – a **Role**
  + on register every be user.

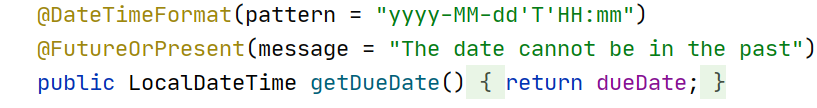
### Exercise

Create an Exercise class, which holds the following properties:

* **id** – UUID **String or Long**
* name – a **String**.
  + name length must be minimum two characters**!**
* **startedOn** – a **Date and Time**
  + startedOn cannot be in the future
    - Little Hint



* **dueDate** – a **Date and Time**
  + dueDate cannot be in the past
    - Little Hint



### Homework

Create a User class, which holds the following properties:

* **id** – UUID **String or Long**
* addedOn – a **Date and Time**.
  + auto set time and date now
* **gitAddress** – a **String**
  + gitAddress must be a valid github address in pattern: https:/github.com/{username}/{homeworkExample}/
* **author** – a **User**
  + logged in user
* **exercise** – an **Exercise**
  + exercise that owns it

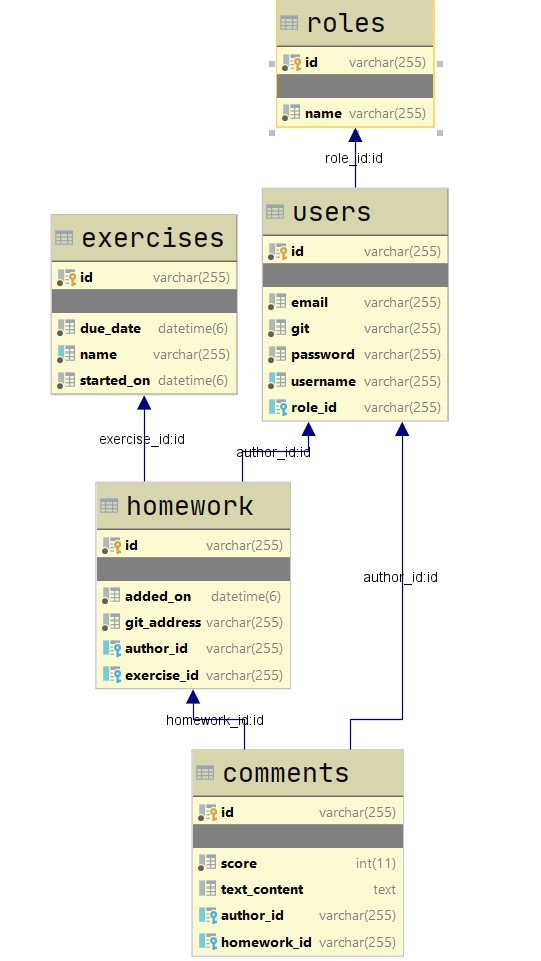
### Comment

Create a User class, which holds the following properties:

* **id** – UUID **String or Long**
* score – an **Integer**.
* **textContent** – a very long **String**
* **author** – a **User**
* **homework** – a **Homework**

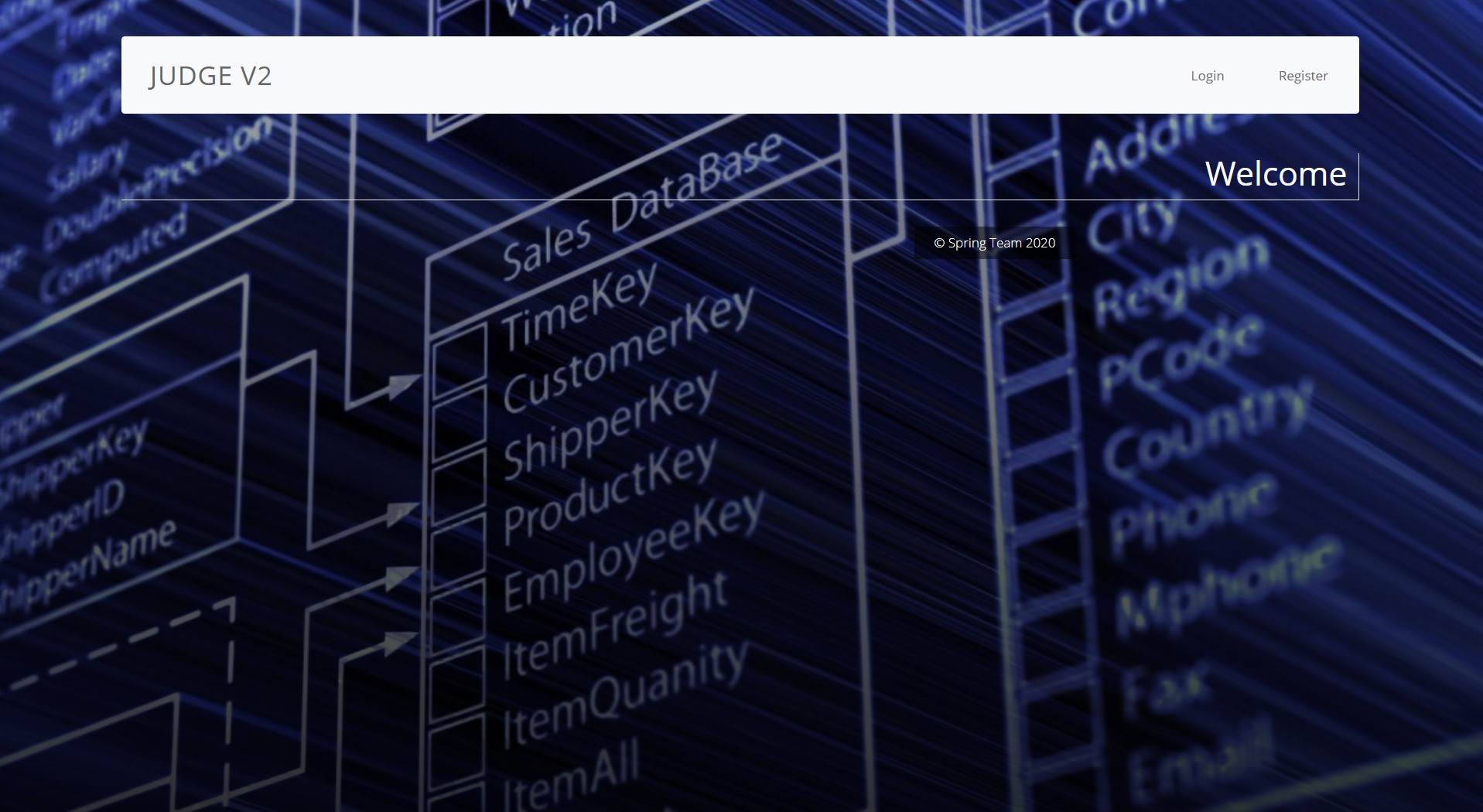
You can change the database and entities so far as the project works according to the task.

**Example for ER Diagram**



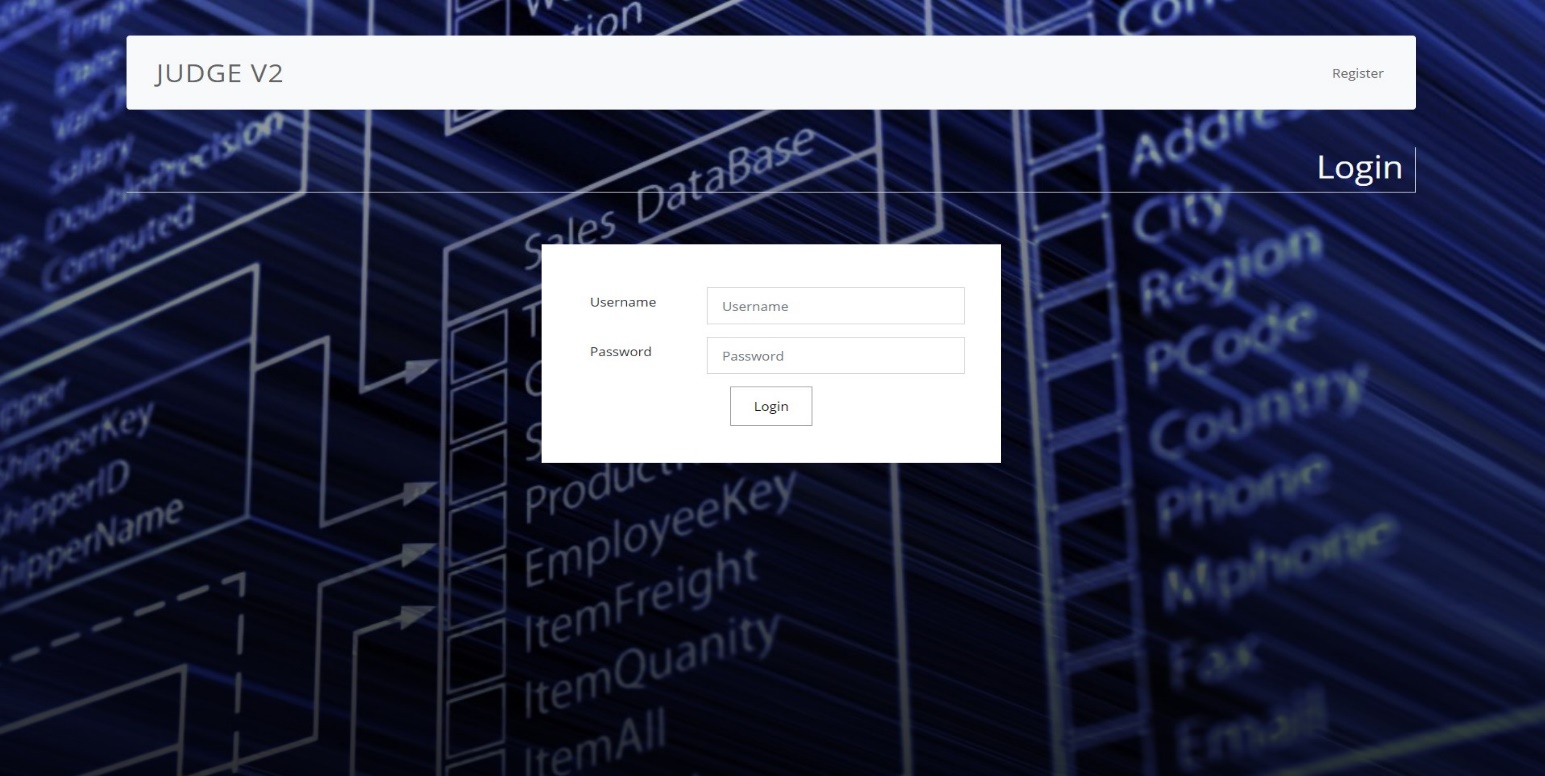
## index - route ("/")

* It should support only a **GET** request.
* It should return the following HTML page, upon a **GET** request.

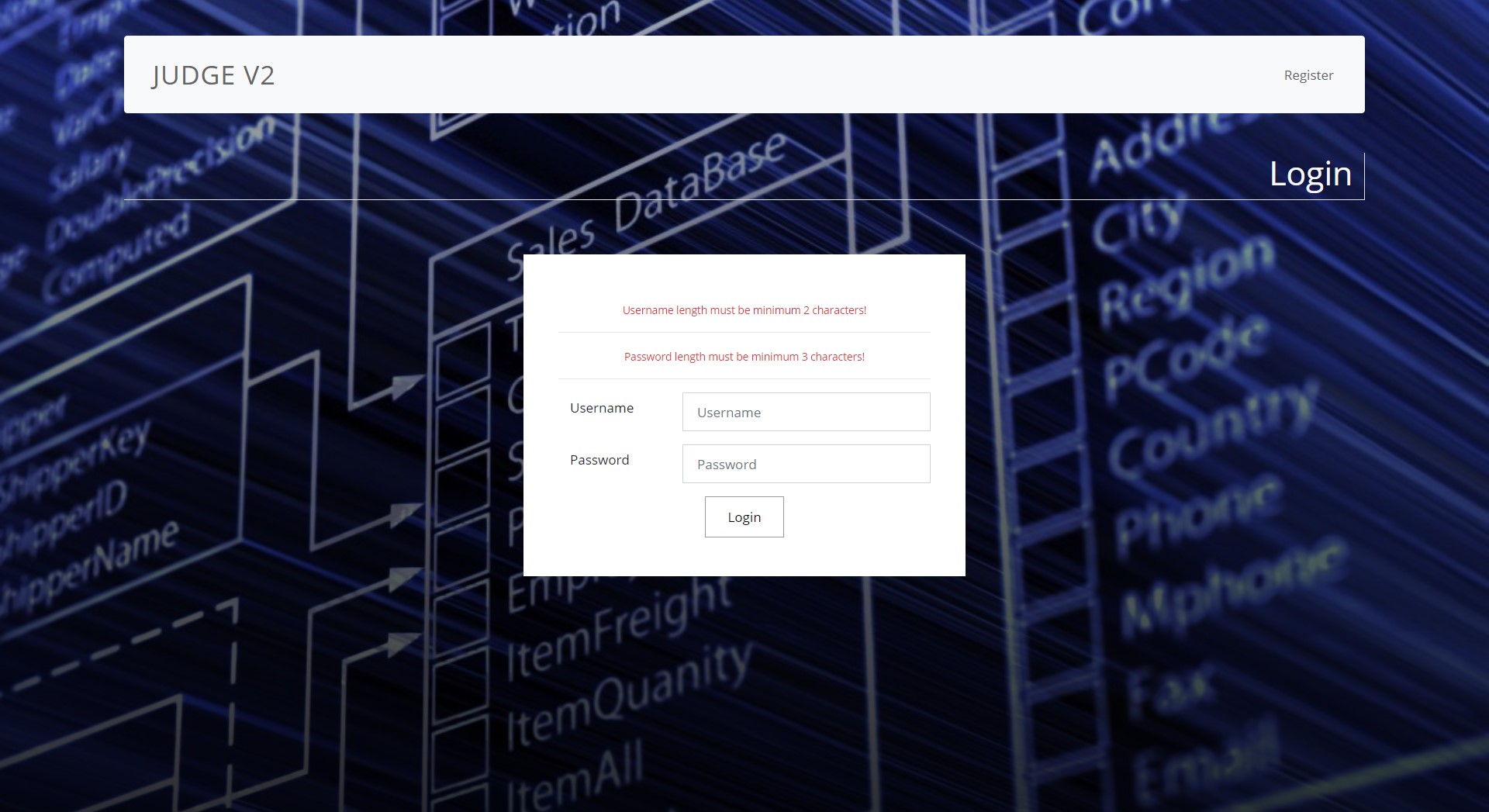


## Login page - route ("/users/login").

* It should support **GET** & **POST** requests.
* It should return the following HTML page, upon a **GET** request.
* When login successfully redirect to "**/home**".

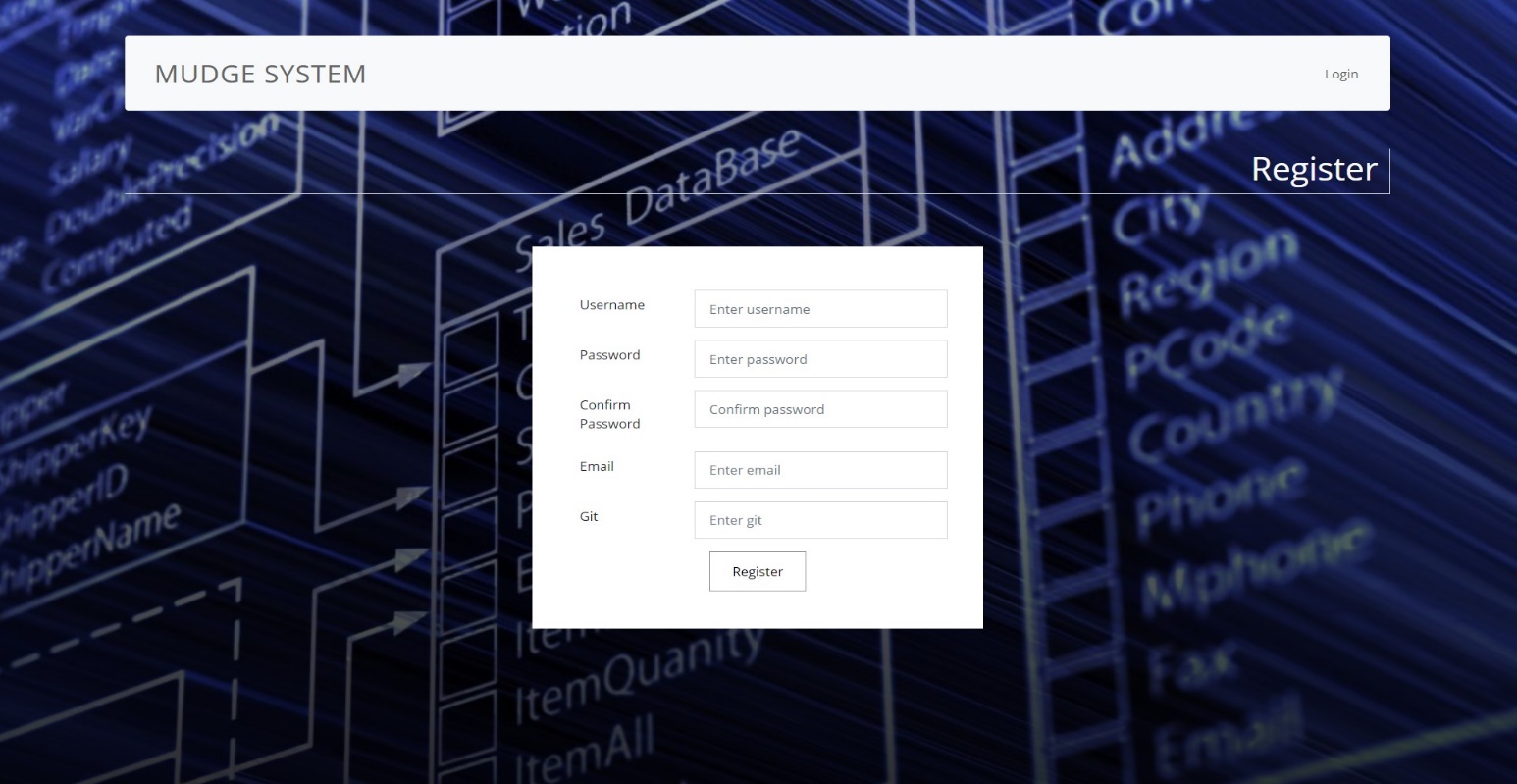


* When the users enter the wrong username or password, you must show the appropriate error message.

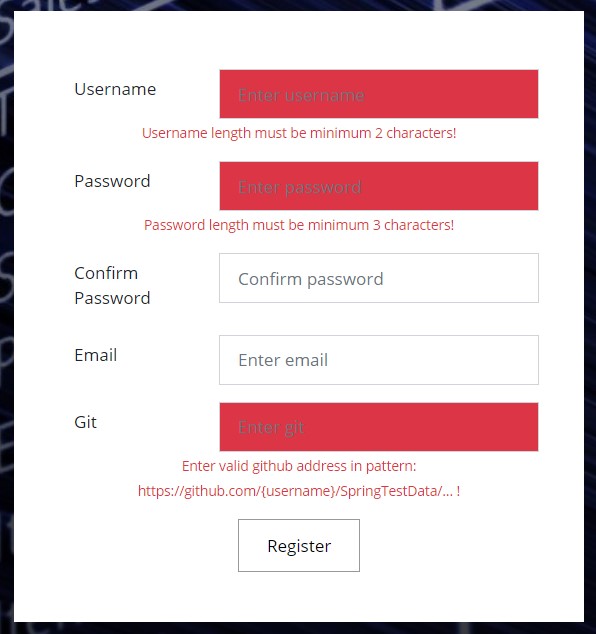


## Register - route ("/users/register").

* It should support **GET** & **POST** requests.
* It should return the following HTML page, upon a **GET** request.
* When register successfully redirect to "**/users/login**".

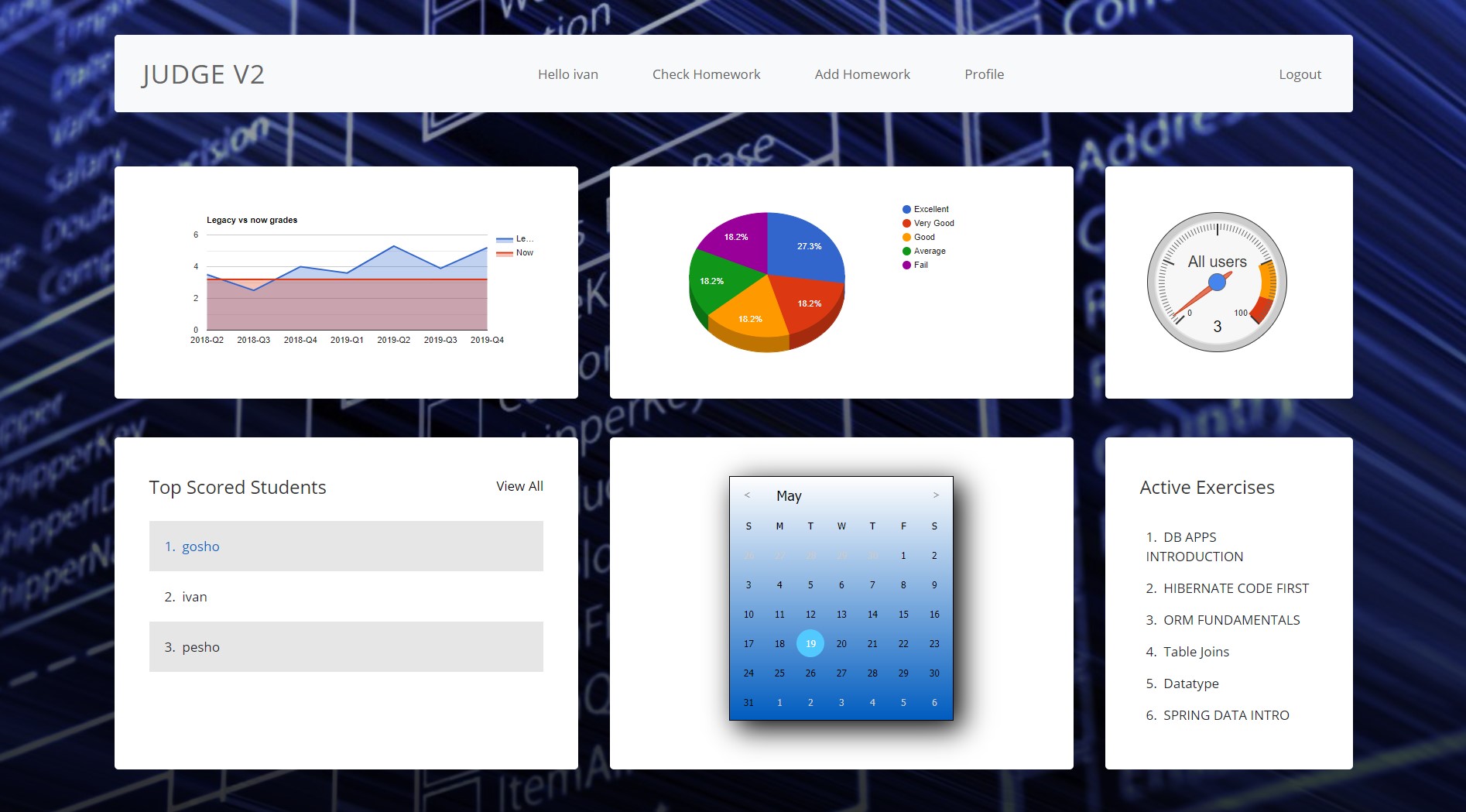


* When the users enter the wrong username or password, you must show the appropriate error message.



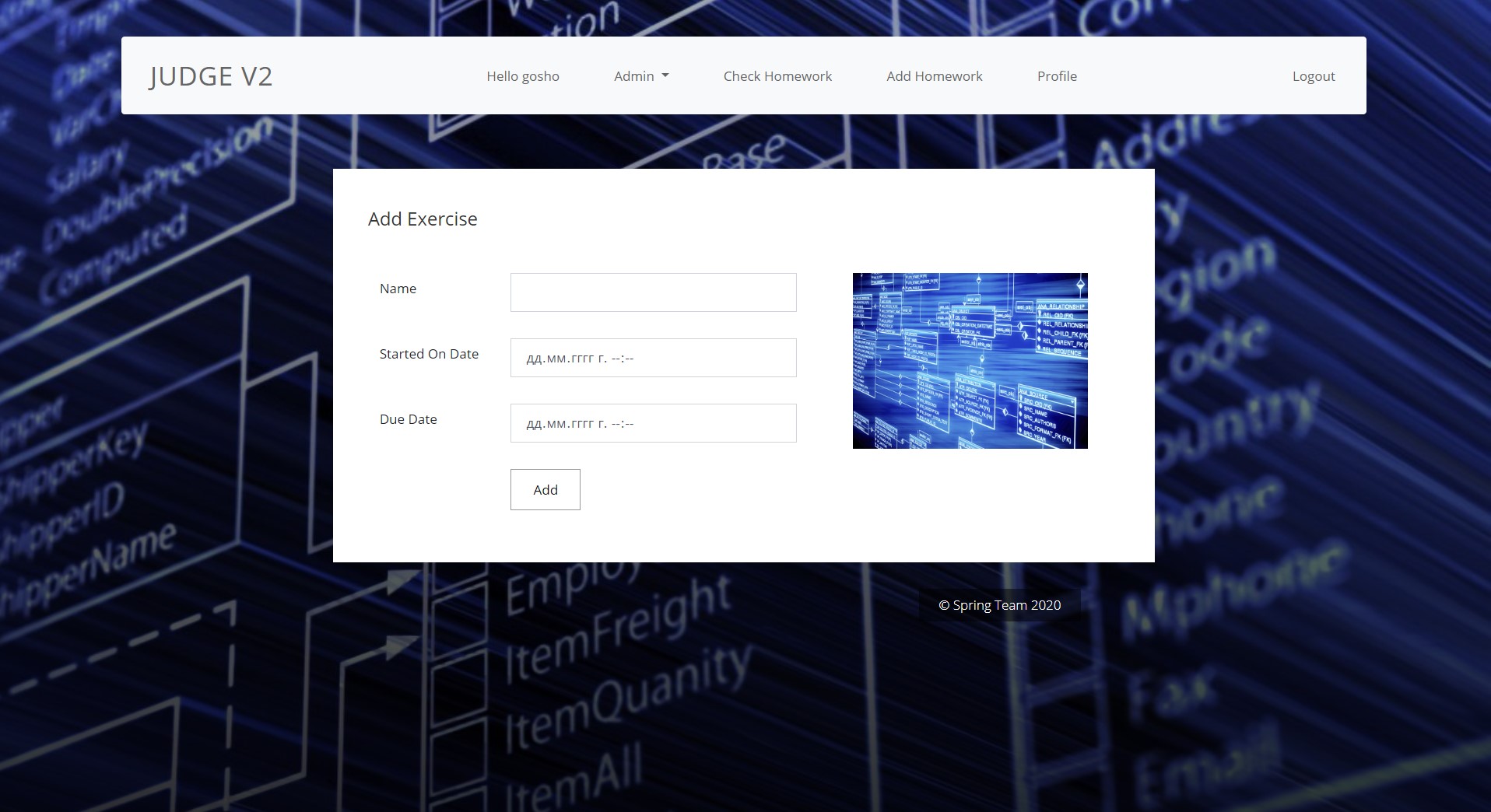
## Home page - route ("/home").

* It should support a **GET** request.
* It should return the following HTML page, upon a GET request.
* For now, just create home page for the logged in user, who are not admins.
* Later we will explain all for this page in details, for now just show it to the user on this route.
* Guest users cannot access it. They will redirect to login page.

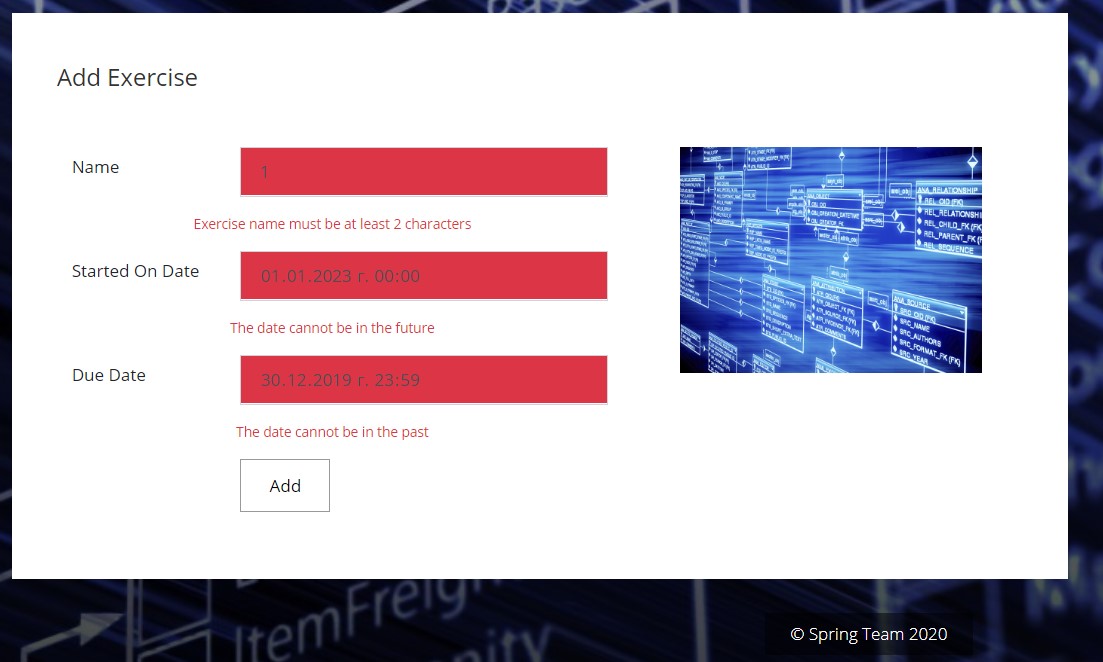


## Add Exercise - route ("/exercises/add").

* Only admins have permission to add exercises. When a user tries to access it, we must redirect it to home page.



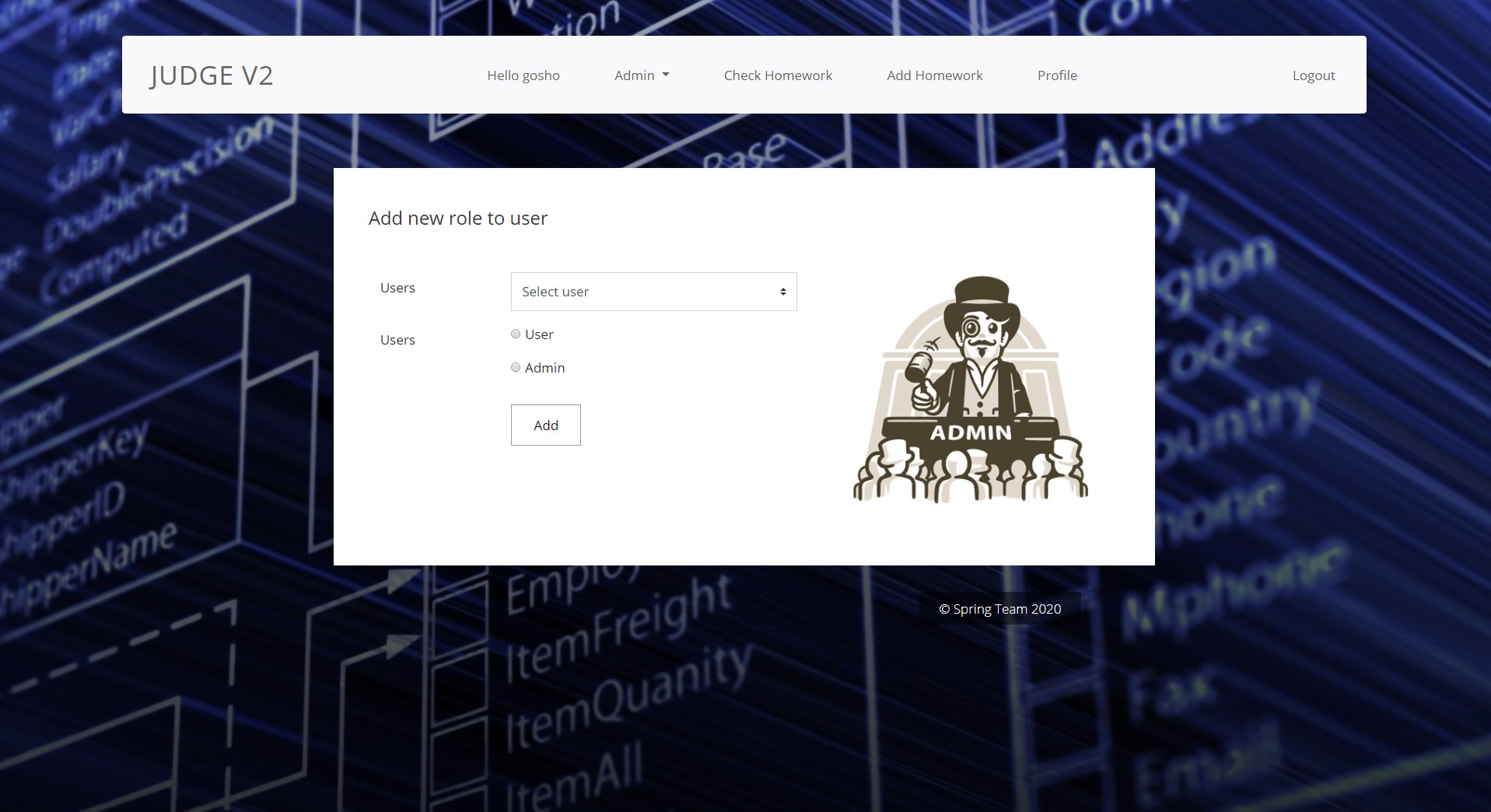
* When the admin tries to add exercise with the wrong parameters, it should return the appropriate error message



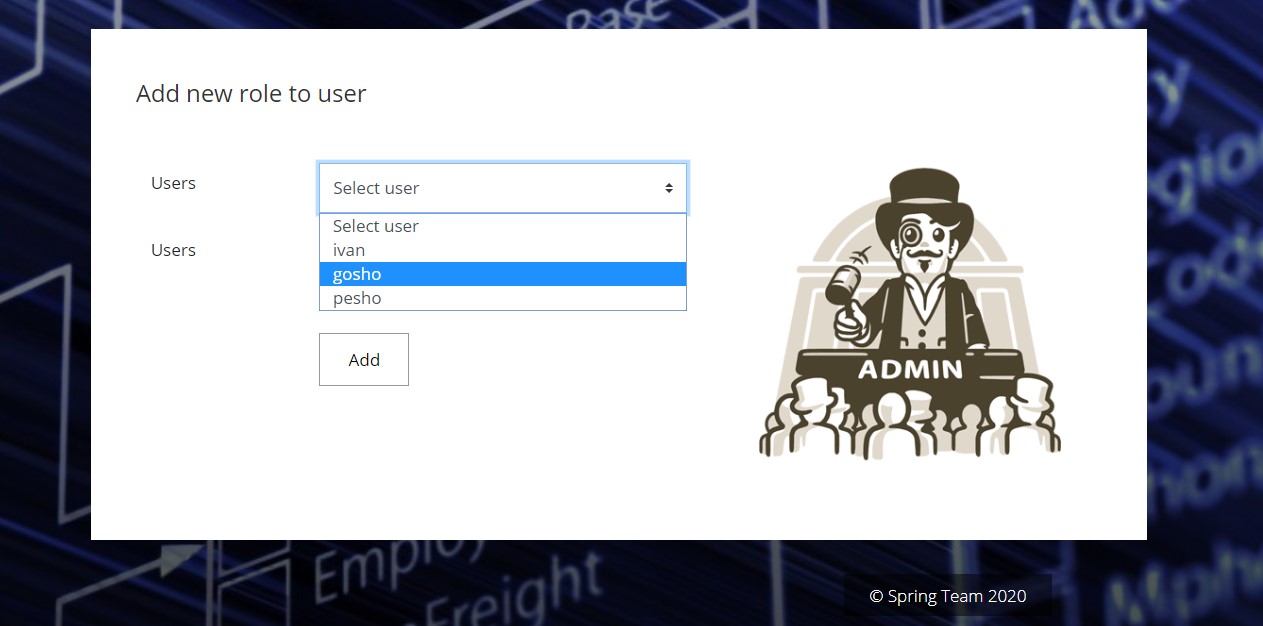
* We must add some Exercises before adding the homework, because they depend on it.

## Add role - route ("/roles/add").

* It should support a **GET** and **POST** request.
* It should return the following HTML page, upon a **GET** request.
* This functionality is only available for the **ADMINS**. If a **USER** tries to access it, we will redirect him to home page.

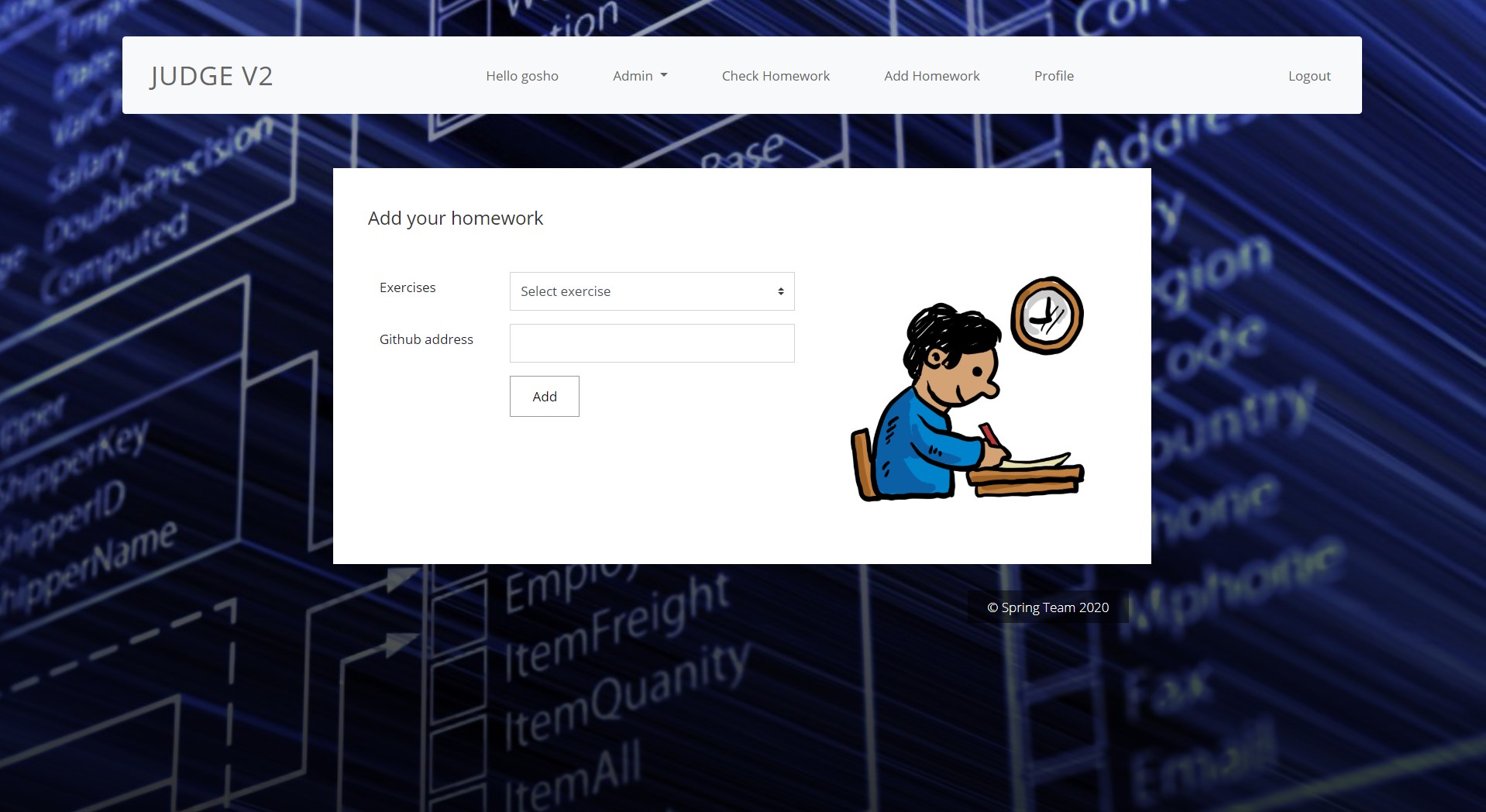


* In a dropdown form, you must show all the registered users in the application.

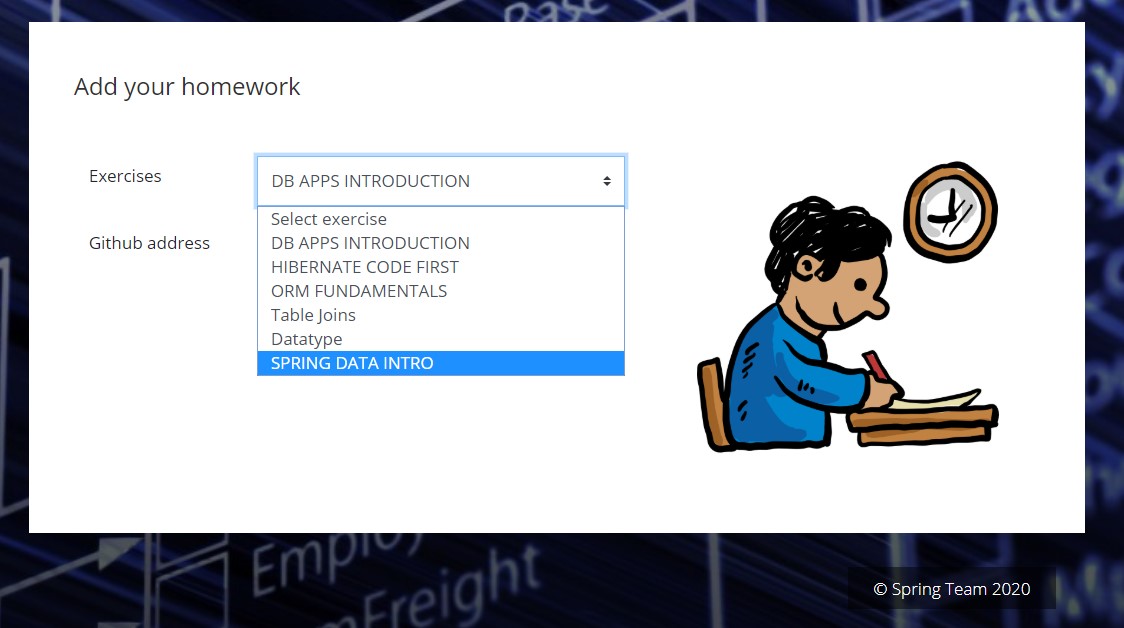


## Add homework - route ("/homework/add").

* It should support a **GET** and **POST** request.
* It should return the following HTML page, upon a **GET** request.
* This functionality is available for all users.

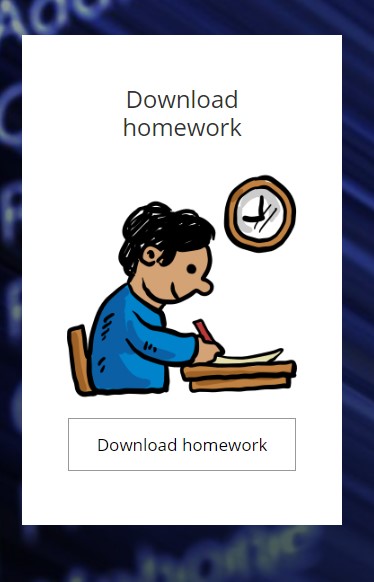


* In a dropdown form, you must show all the registered exercises in application. That's why you need to add some exercises first.

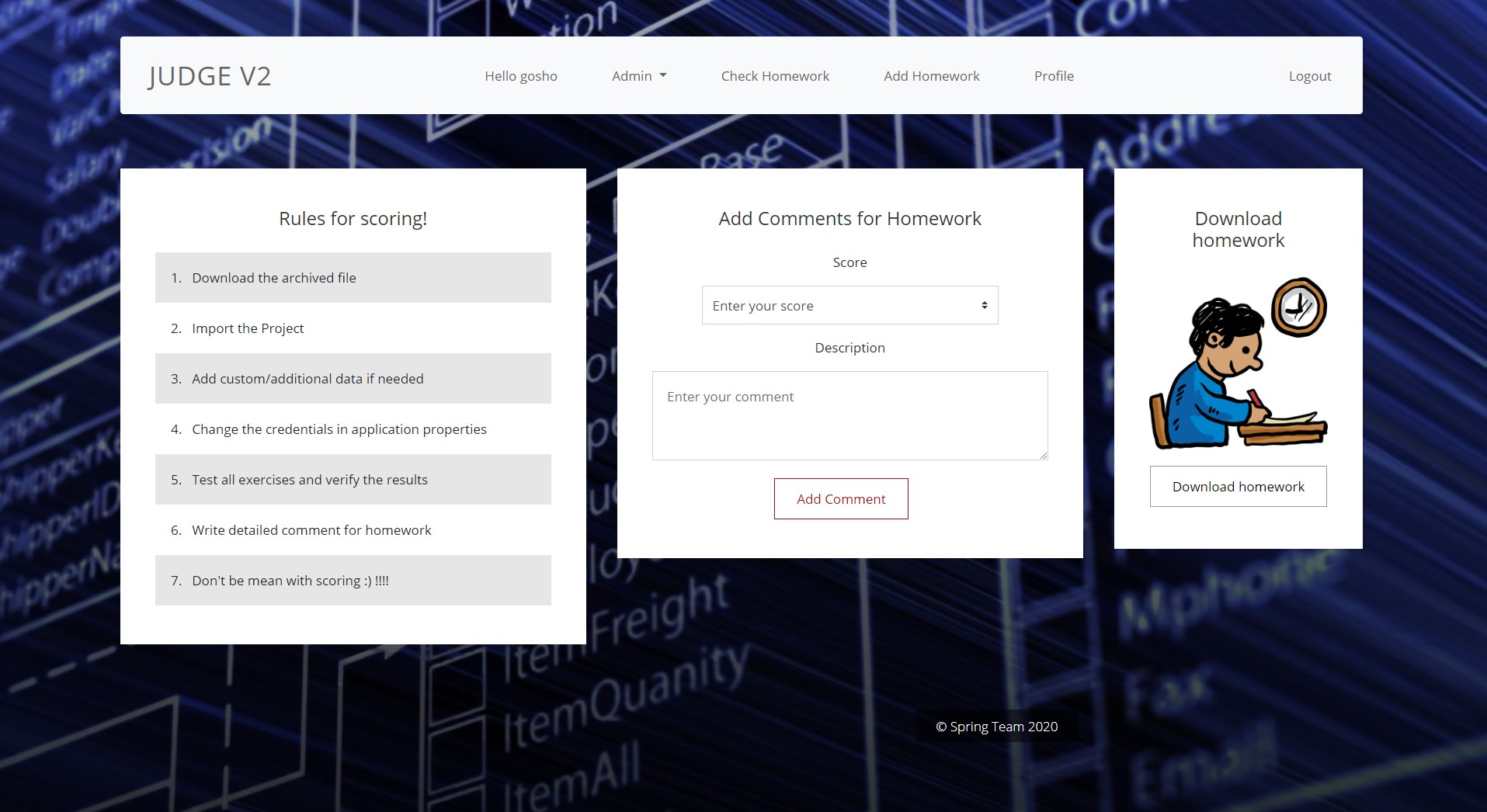


## Check homework - route ("/homework/check").

* It should support a **GET** and **POST** request.
* It should return the following HTML page, upon a **GET** request.
* This functionality is available for all users.
* First the checker must download homework. This homework must be random from the database, or choose one that have lower count of comments.

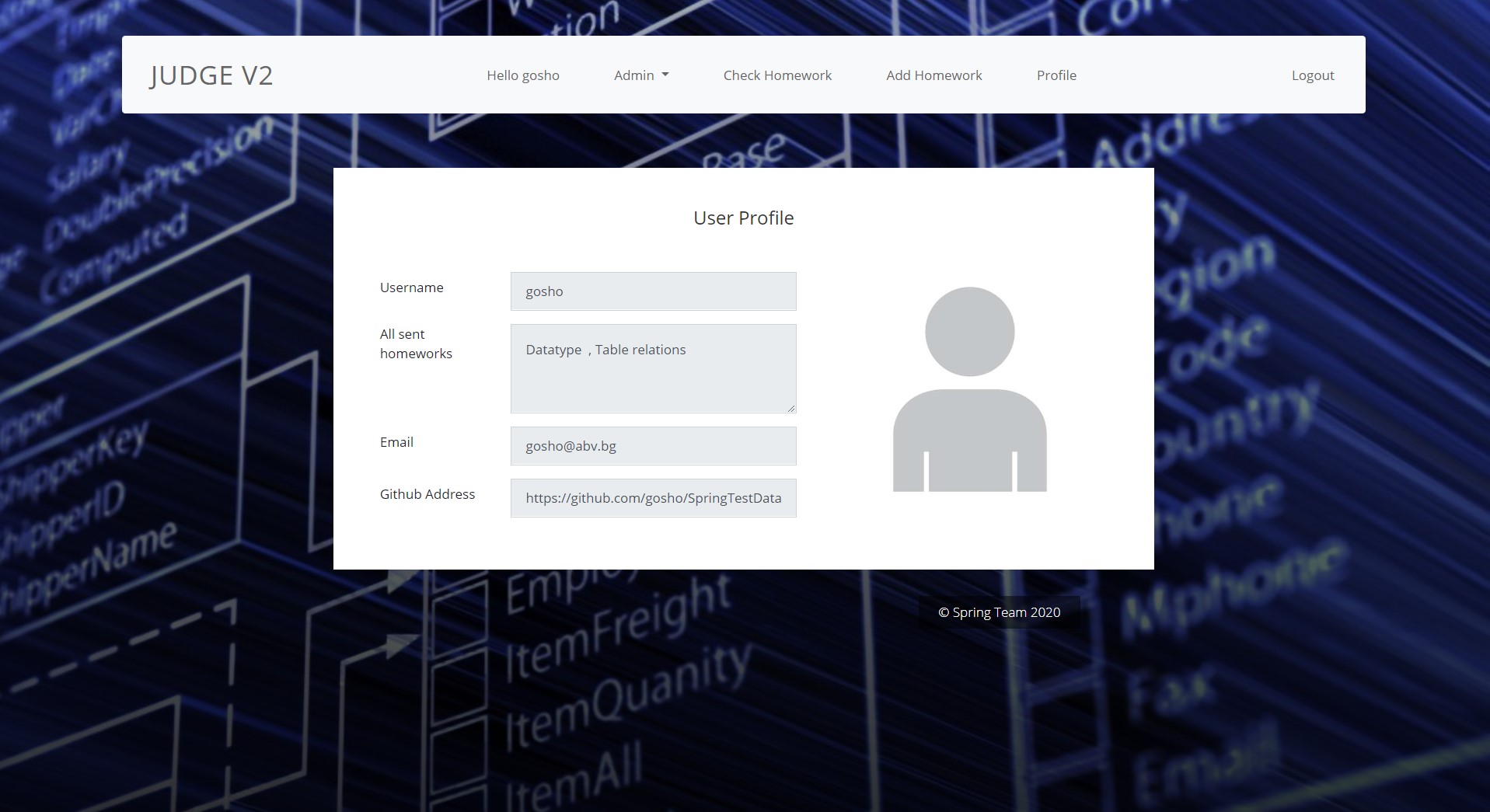


* When he checks the homework, mark his grade and comment for all.



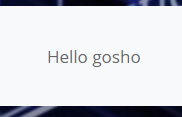
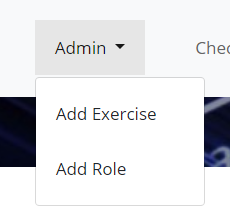
## Profile homework - route ("/users/profile/{id}").

* It should support only a **GET** request.



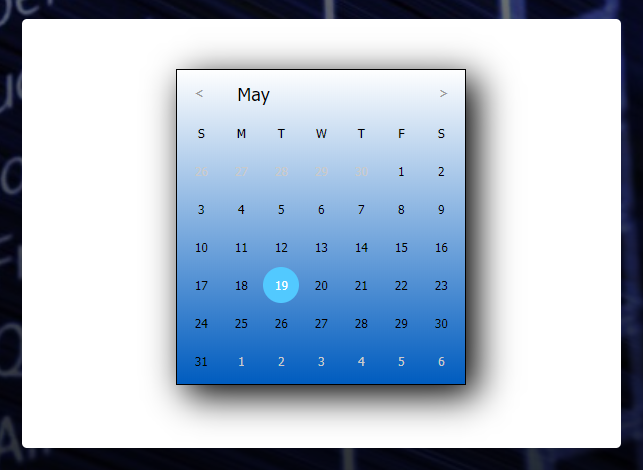
## The navbar

Let explain all navbar tabs.

* The logged in user must be greeted   
    
  - 
* If the user is an ADMIN, show this dropdown  
    
   
* All other tabs are links to the corresponding pages.

## Home page – part one.

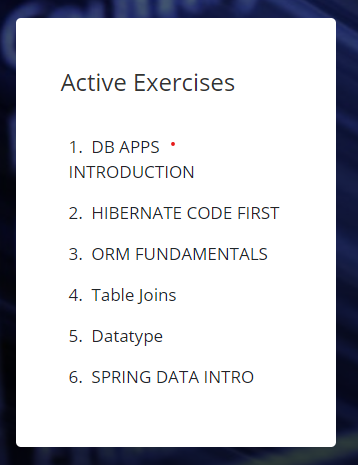
1. Let's start with the calendar. The calendar is created thanks to a JS script in the head element.



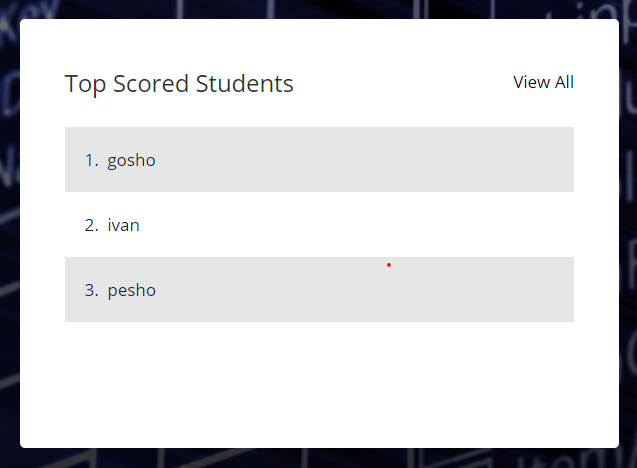


* This script gives you a working calendar. Later, in JS lectures, you will be overviewing scripts in details.

1. The next tab is "**Active Exercises**". Here you must show all the exercises in the database



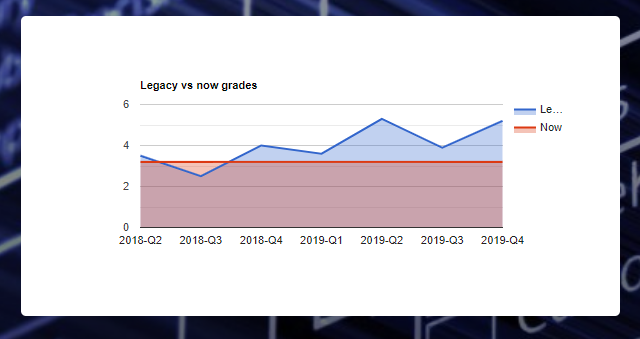
1. In the "**Top Scored Students**". You must show all the students, ordered by their average grades.



## Home page – all attributes – part two.

In this part, we will see how easy to use [Google Charts](https://developers.google.com/chart) in your projects. Because for now, some of you are not very familiar with JavaScript, we use pre-made scripts that require only some variables and visualize the chart for us. Feel free to change everything, that you think is needed for the applications. Now we'll see all three chart scripts and explain them.

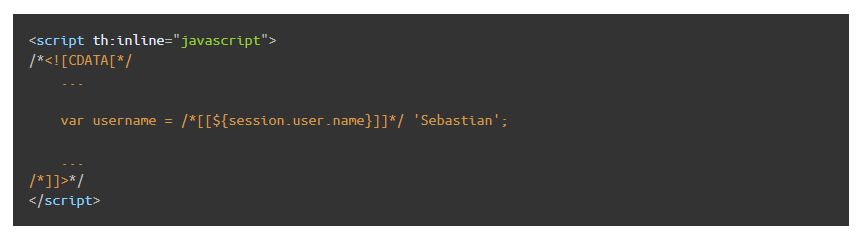
1. Legacy vs Now



* This is the script. You have it in HTML home page. The script is there, because in this way, you can easily pass parameters with the help of Thymeleaf without the need of additional JS coding skills and REST controllers.
* Here is the script and comment where you must pass parameter

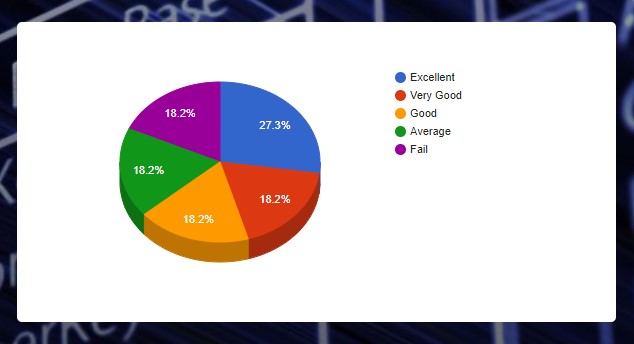


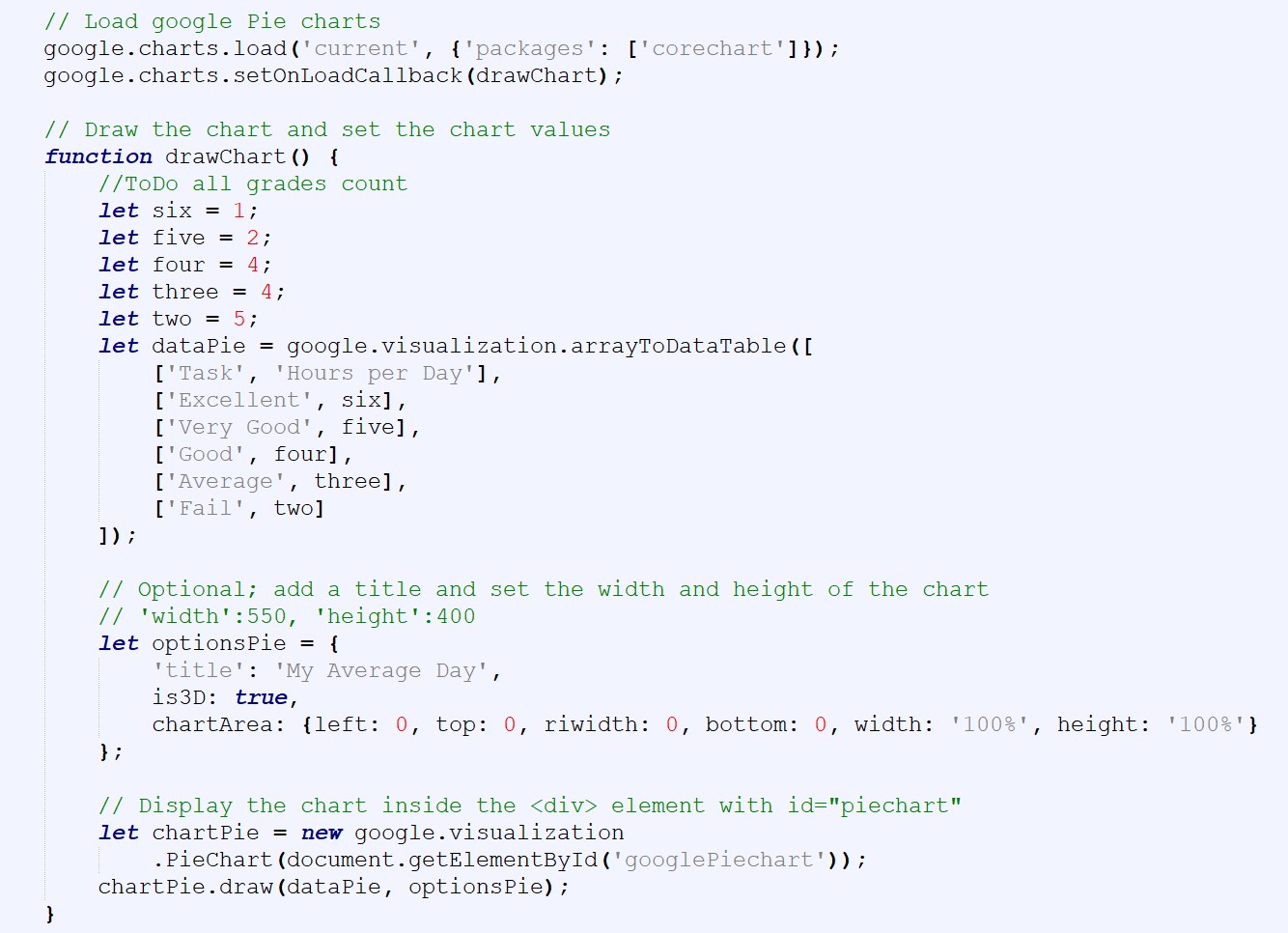
* Here is a little hint, how we can [pass parameter with Thymeleaf](https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#script-inlining-javascript-and-dart)



At the same way you can do it in other two scripts, or add new scripts from Google Charts.

Pie Chart





Gauge

