Battleships Application

Exam Preparation [Spring Fundamentals]

The video game market is quite oversaturated but there is always room for another clone of one of the most famous game series. A group of friends decided to try to develop this game, but they need your help to implement it. The idea is clear but due to the lack of technical knowledge from your colleagues you will have to create the business logic for this project.

1. Database Requirements
The **Database** of the Battleships application needs to support **3 entities**:

User
- **Id** – Accepts **UUID-String** or **Long** values
- **Username**
  - The **length** of the **values** should be **between 3** and **10** characters long (both numbers are **INCLUSIVE**)
  - The values should be **unique** in the database
- **Full Name**
  - The **length** of the **values** should be **between 5** and **20** characters long (both numbers are **INCLUSIVE**)
- **Password**
  - The **length** of the **values** should be more than **3** characters long (**INCLUSIVE**)
- **Email**
  - The values should contain a **'@'** symbol)
 - The values should be **unique** in the database

Ship
- **Id** – Accepts **UUID-String** or **Long** values
- **Name**
  - The **length** of the **values** should be **between 2** and **10** characters long (both numbers are **INCLUSIVE**)
  - The values should be **unique** in the database
- **Health**
  - The **values** should be **positive numbers**
- **Power**
  - The **values** should be **positive numbers**
- **Created**
  - The **values** should **not** be **future dates**
- **Category**
  - A **relationship** with **the Categories Entity**
- **User**
  - A user that **owns the ship**

Category
- **Id** – Accepts **UUID-String** or **Long** values
- **Name**
  - An option between **BATTLE**, **CARGO** and **PATROL**
  - The values should be **unique** in the database
- **Description**

- A very long and detailed description of the category
- Can accept **null values**

Nullable/Empty/Blank values are not allowed unless explicitly mentioned. Implement the entities with the **correct data types** and implement the **repositories** for them. Here is the ER Diagram.
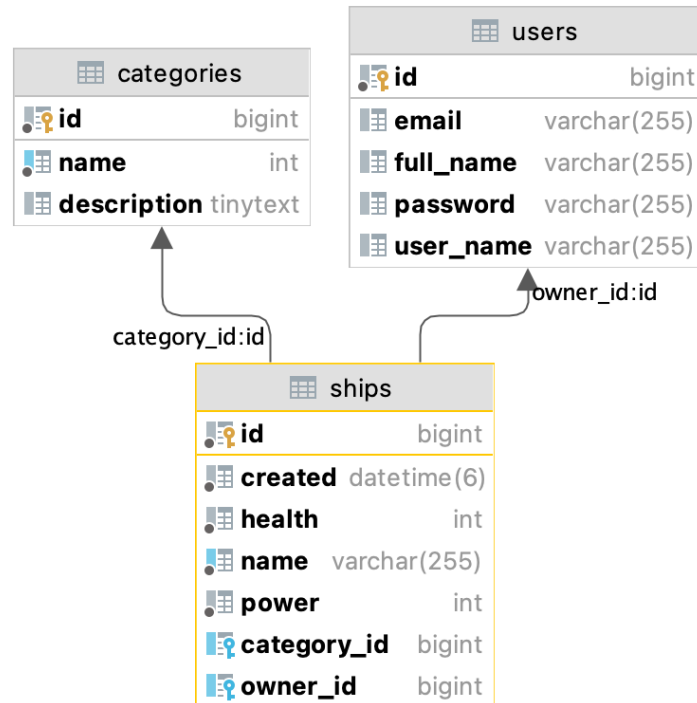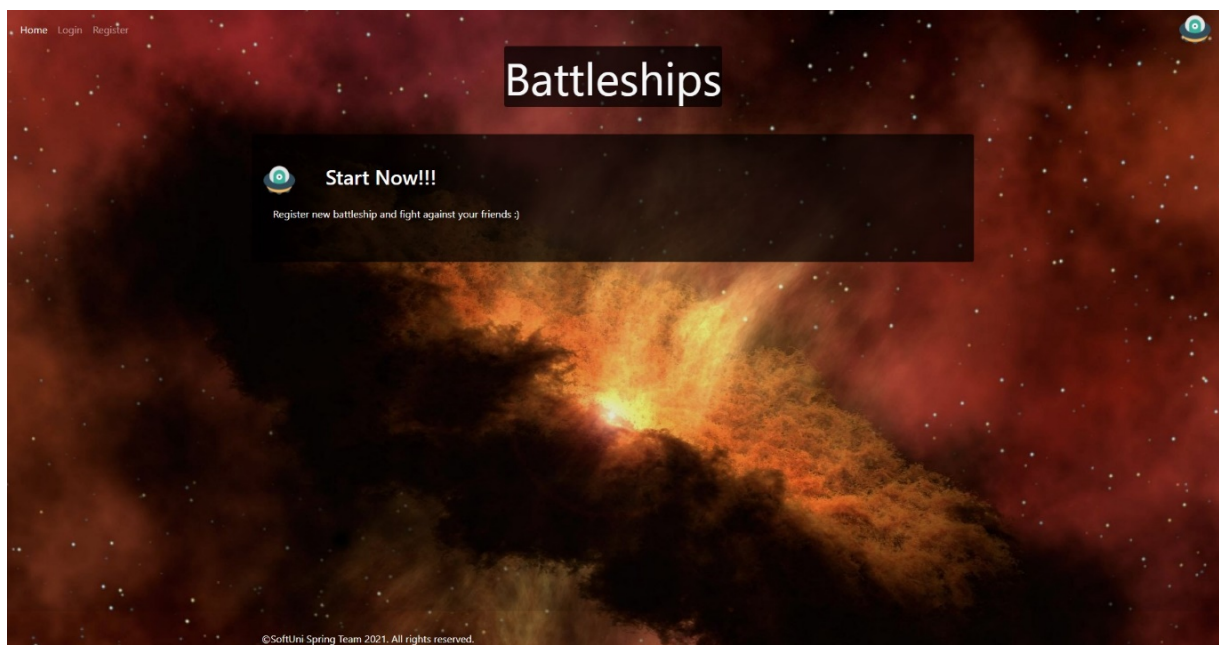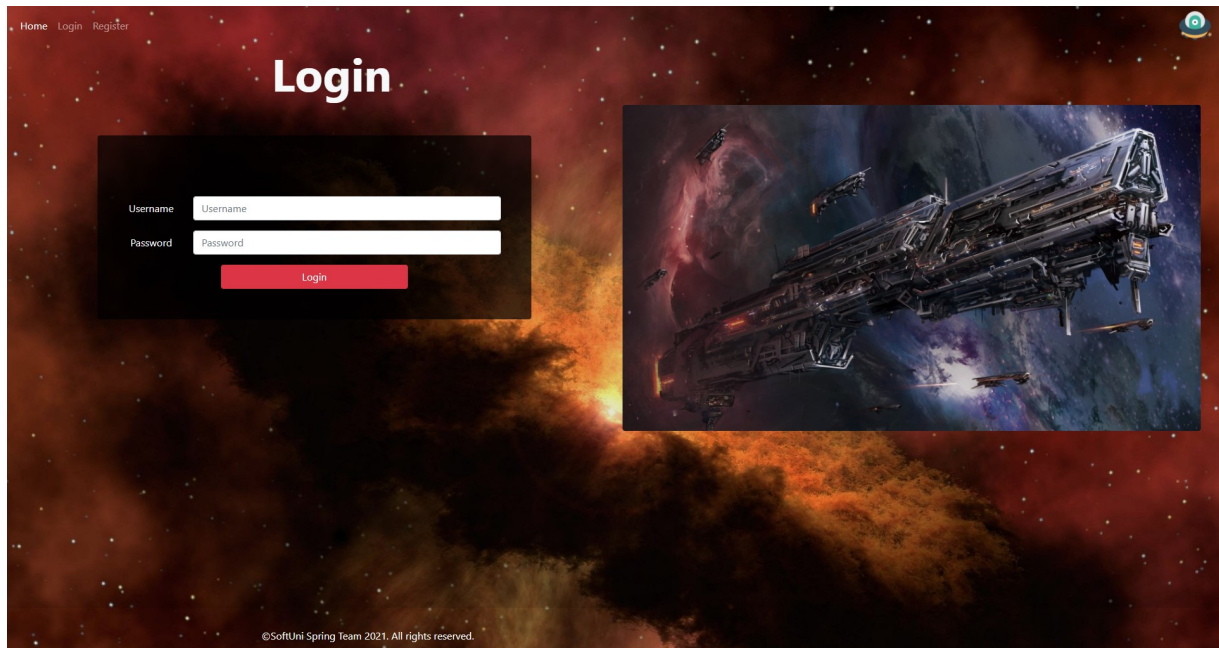

2. Initialize categories
- Implement a method that checks (when app started) if the database does not have any category and initialize them
  - You are free to do this in some different ways
  - You can skip the description if you want

3. Page Requirements
Index Page (logged out user)

Login Page (logged out user)

Login Page validations

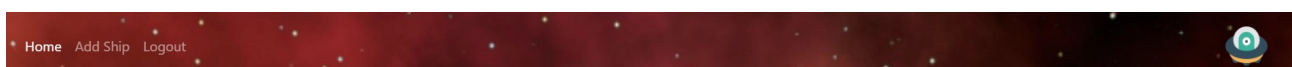**Registration Page (logged out user) / Registration Page validations**

Navigation (Guest user)
- Note: can access only **Index**, **Login**, **Registration** pages.
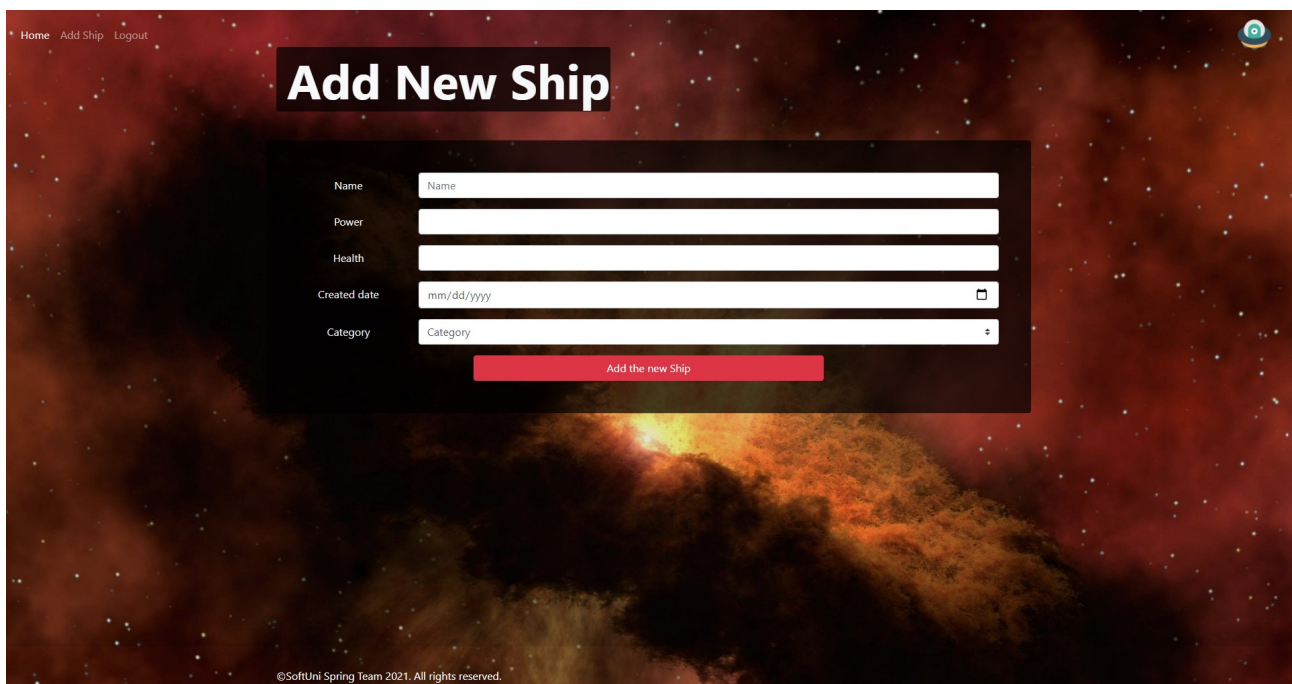
Navigation (Registered user)
- Note: can access only **Home**, **Add Ship**, **Logout** pages.

Add Ship

Add Ship validation

Home Page

NOTES:
- You must select **one** of the ships that are **owned** by the **current user**
- You must select **one** of the ships that are **owned** by **other users**
- In the last section you should list all the ships **ordered by** their status (name, health, power) in the following format: "**| name | health | power |**"
- When pressing the fire button, the **attacker hits** the defender and **reduces** his health by the value of the attacker's power. The **defender also hits** the attacker and **reduces** the attacker's health by the value of defender's power. If the defender's or the attacker's healths are **less than or equal to 0**, **remove** their ship from the database. After the attack, the application must redirect again to the home page.

The templates have been given to you in the application skeleton, so make sure you implement the pages correctly.

- The templates should look **EXACTLY** as shown above.

- The templates do **NOT require additional CSS** for you to write. Only the provided **bootstrap** and **CSS** are enough.

4. Functional Requirements
- The Functionality Requirements describe the functionality that the **application** must support.
- The **application** should provide **Guest** (not logged in) users with the functionality **login**, **register** and view the **Index** page.
- The **application** should provide **Authenticated** (logged in) users with the functionality to **logout**, **add a ship**, view **home** page and ready to **attack** other ships.
- In the **Battleships Application**, the navigation bar should redirect to the appropriate URL depending on if the user is logged in.
- The **application** should provide the **functionality** for **adding ships** with **categories** and **users**. Also, the ships should **fire** at other ships and **remove** them from the database when their health is **lower than** or **equal** to 0.
- The **Attack** button **creates** the **attack** to the defender and **redirects** to the home page.
- The **application** should **store** its **data** into a **MySQL** database.

5. Security Requirements
The Security Requirements are mainly access requirements. Configurations about which users can access specific functionalities and pages.
- **Guest** (not logged in) users can access:
  - **Index** page;
  - **Login** page;
  - **Register** page.
- **Users** (logged in) can access:
  - **Home** page;
  - **Add Ship** page;
  - **Logout** functionality.

6. Scoring
Database – 10 points
Pages – 25 points
Functionality – 35 points
Security – 5 points
Validations – 15 points
Code Quality – 10 points