Spring Fundamentals Exam

Battle Ships Application

Exam for the "Spring Fundamentals" course @ SoftUni.

The video game market is quite oversaturated but there is always room for another clone of one of the most famous movie 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 **Battle Ships** 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** must be **between 2** and **10** characters (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**

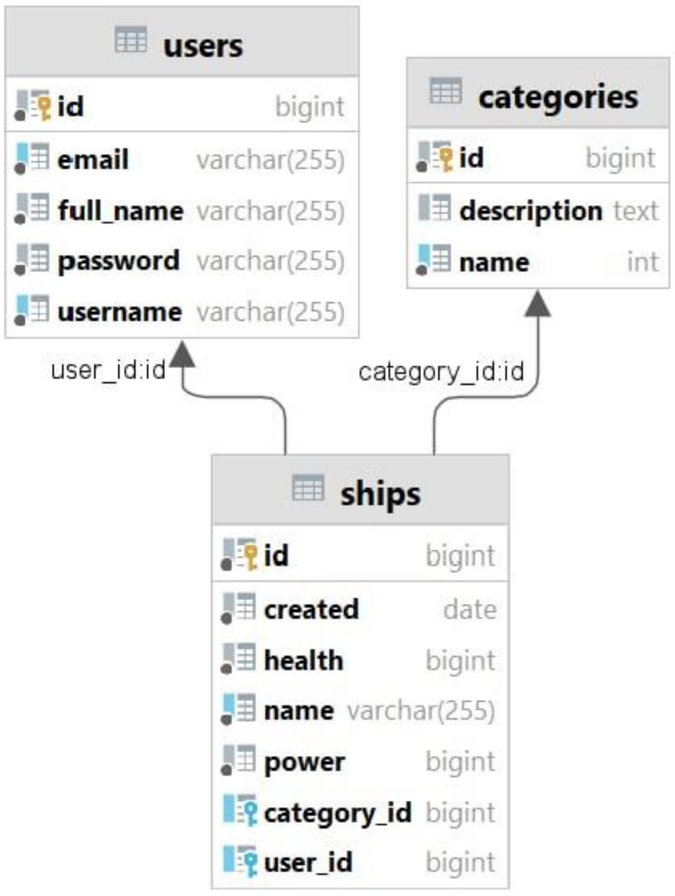
Category

- `Id` **– Accepts UUID-String or Long values**
- **Name**
  - An option between **BATTLE, CARGO**, **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 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:

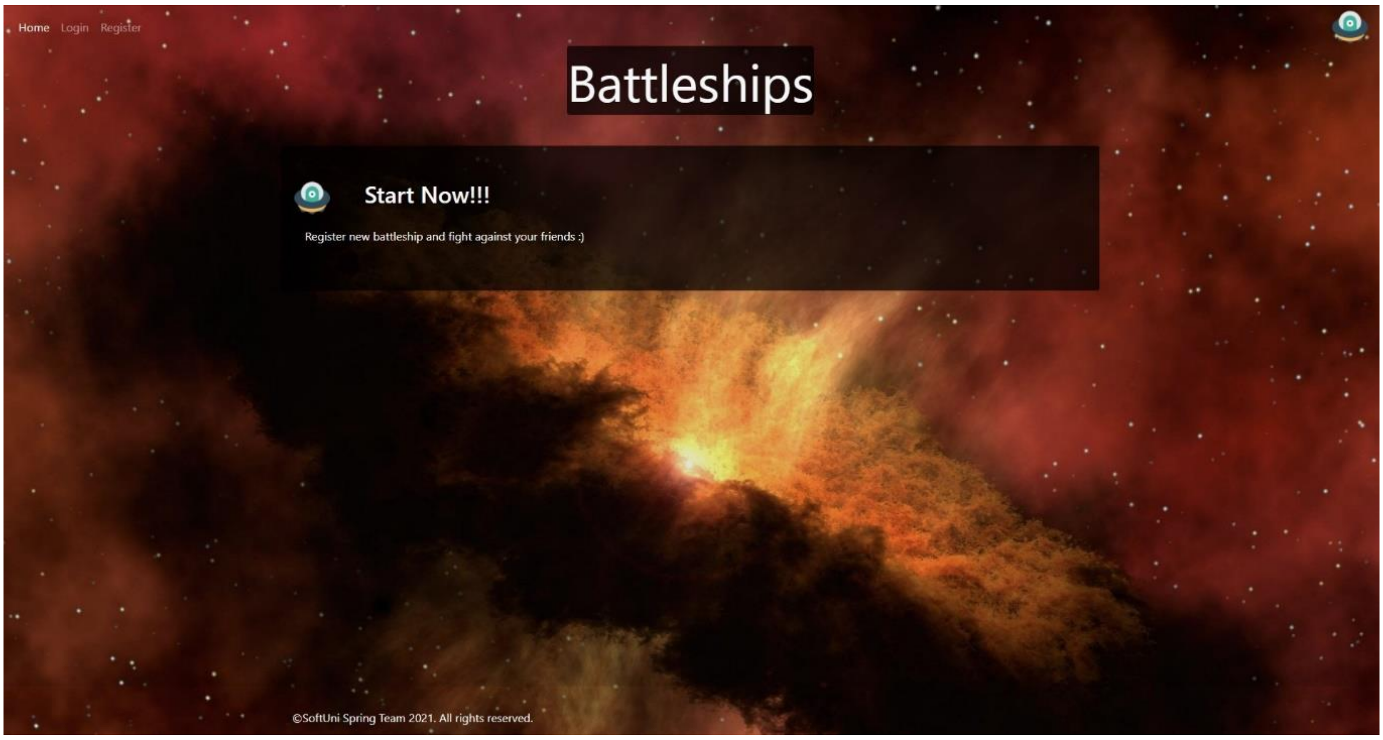


2. Initialize categories

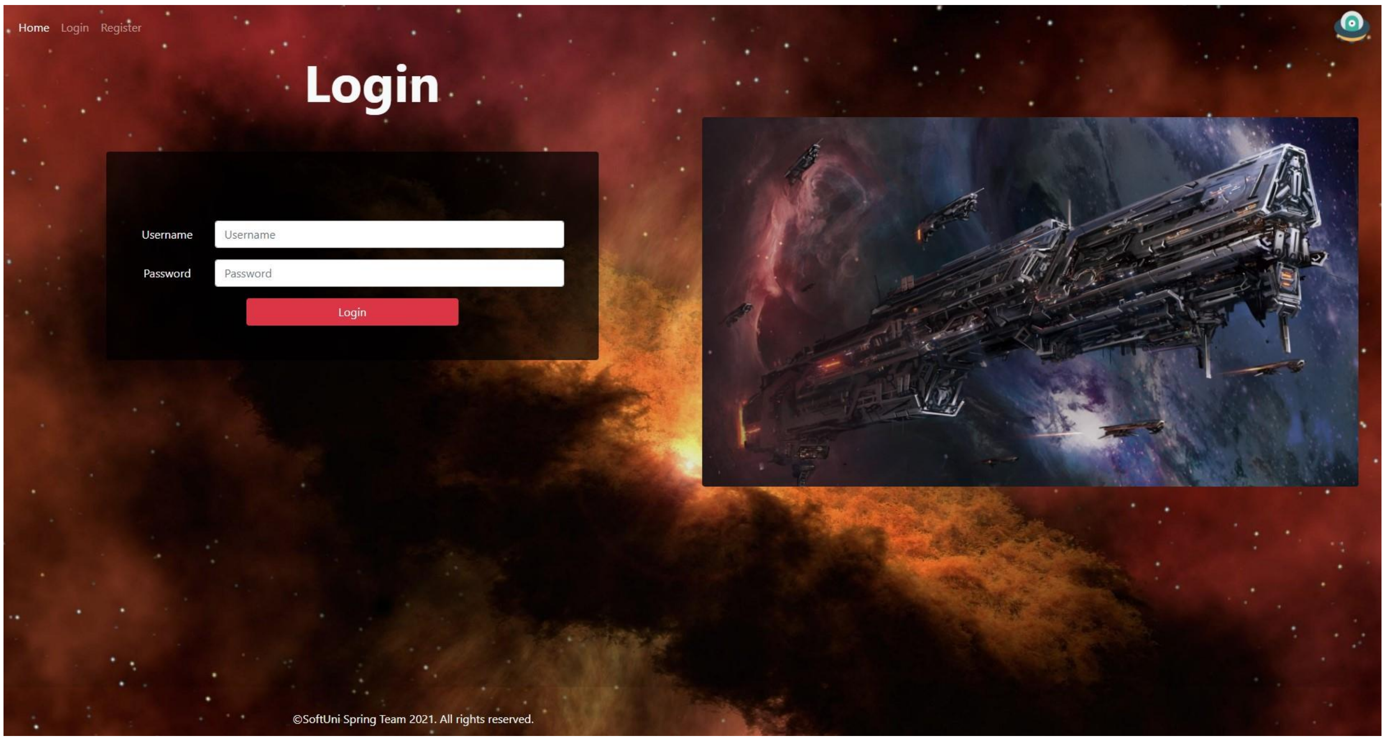
- Implement a method that checks (when app started) if the database does not have any category and initialize them
    - You are free to do this in some different ways.
    - You can skip the description if you want

3. Page Requirements

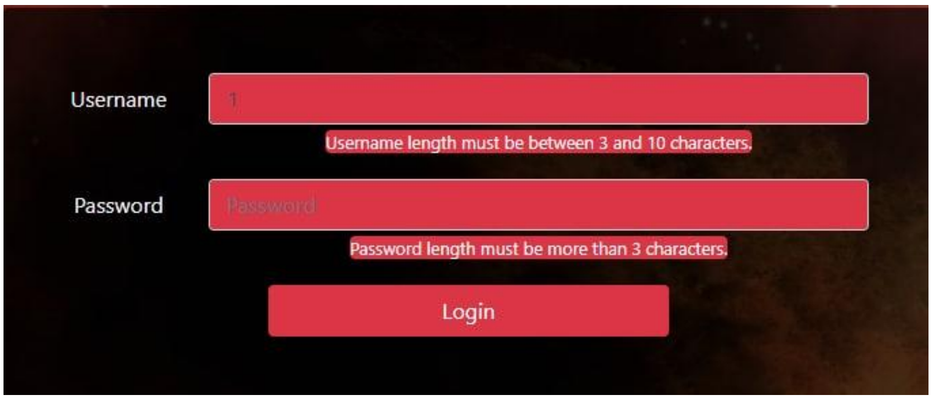
Index Page (logged out user)


Home
Login
Register
Battleships
Start Now!!!
Register new battleship and fight against your friends :)
©SoftUni Spring Team 2021. All rights reserved.

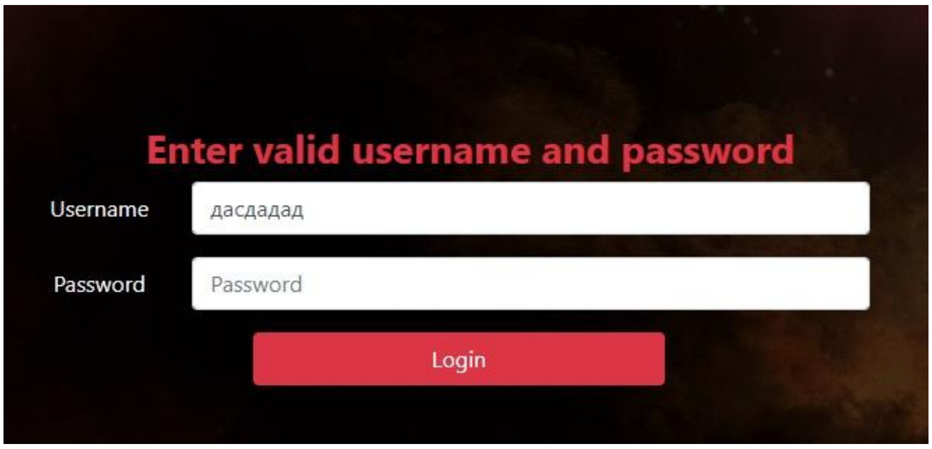
Login Page (logged out user)



Login Page validations



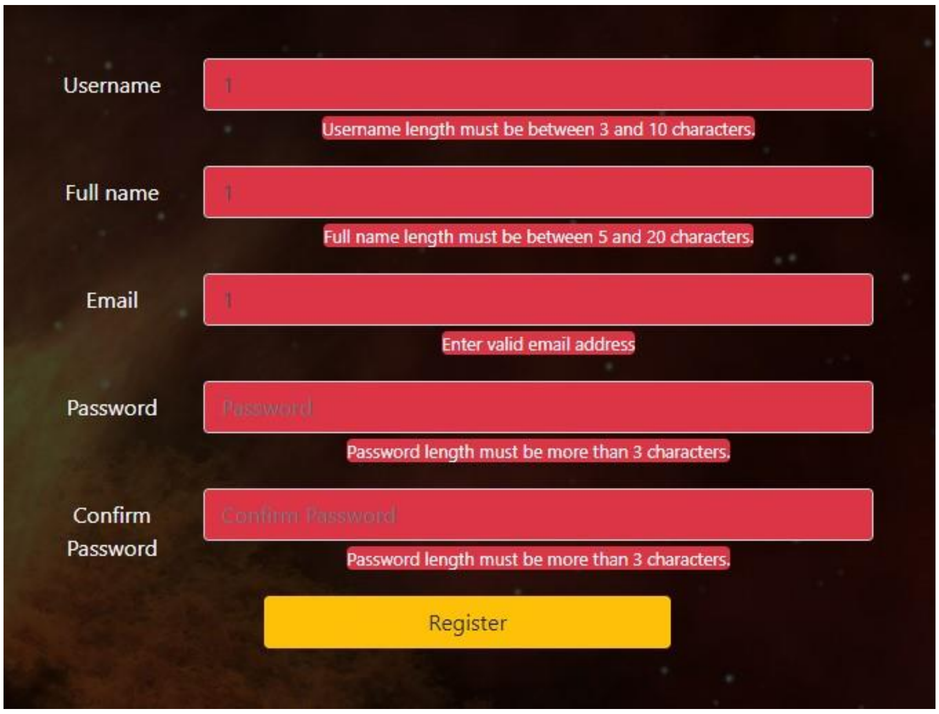
Username length must be between 3 and 10 characters.
Password length must be more than 3 characters.


Enter valid username and password

Register Page (logged out user)

Register Page validations

- Note: it is not necessary to show message for different passwords, just not save the user and redirect again to the register page.



Username
1
Username length must be between 3 and 10 characters.

Full name
1
Full name length must be between 5 and 20 characters.

Email
1
Enter valid email address

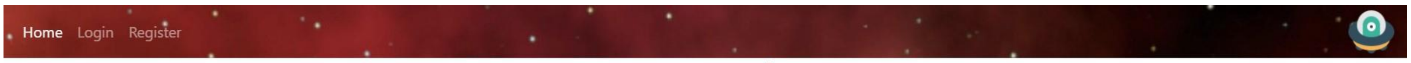
Password
Password
Password length must be more than 3 characters.

Confirm Password
Confirm Password
Password length must be more than 3 characters.

Register

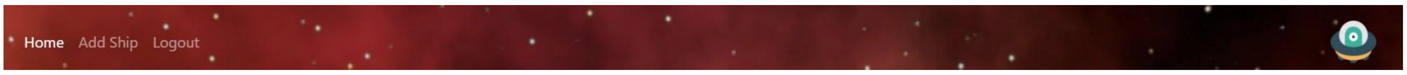
Navigation (Guest user)

- Note: can access only to **Index**, **Login**, **Register** pages.

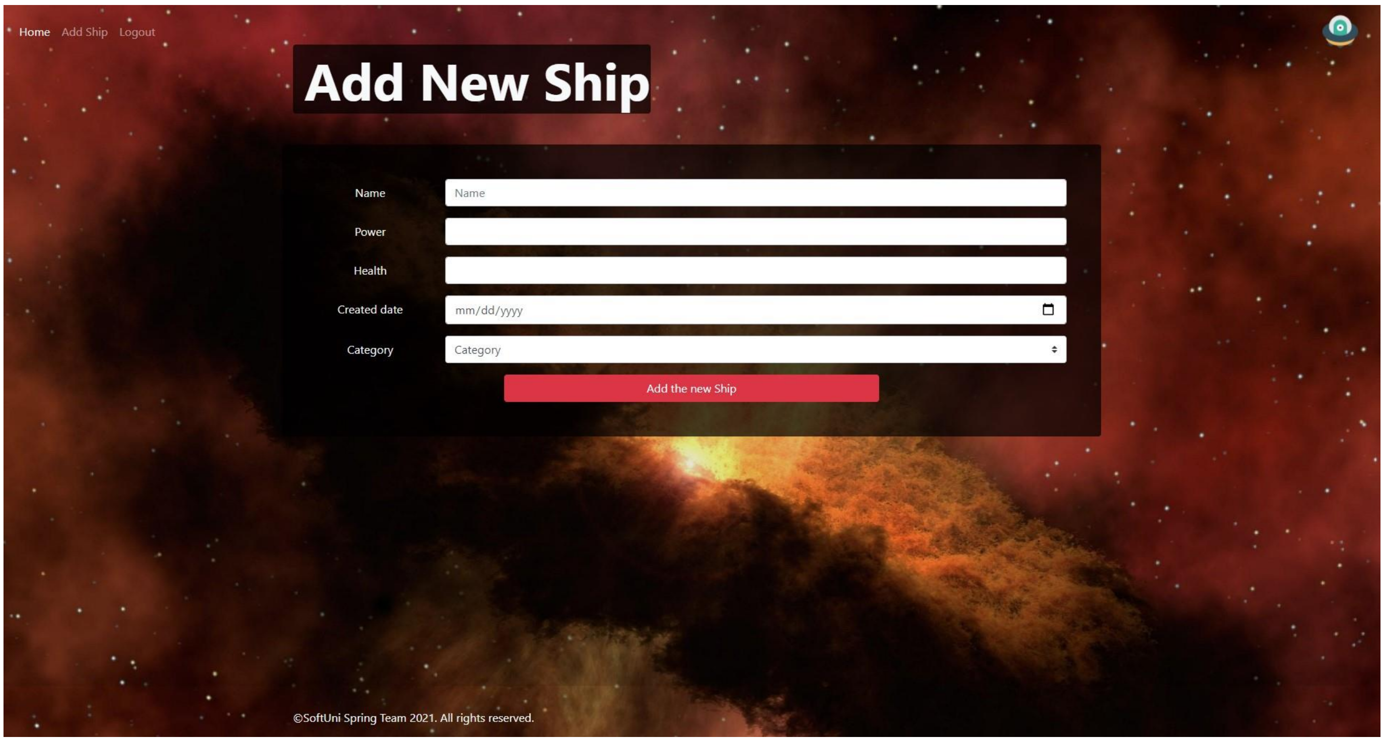


Navigation (Registered user)

- Note: can access only to **Home**, **Add Ship**, **Logout**.



Add Ship



Home Add Ship Logout

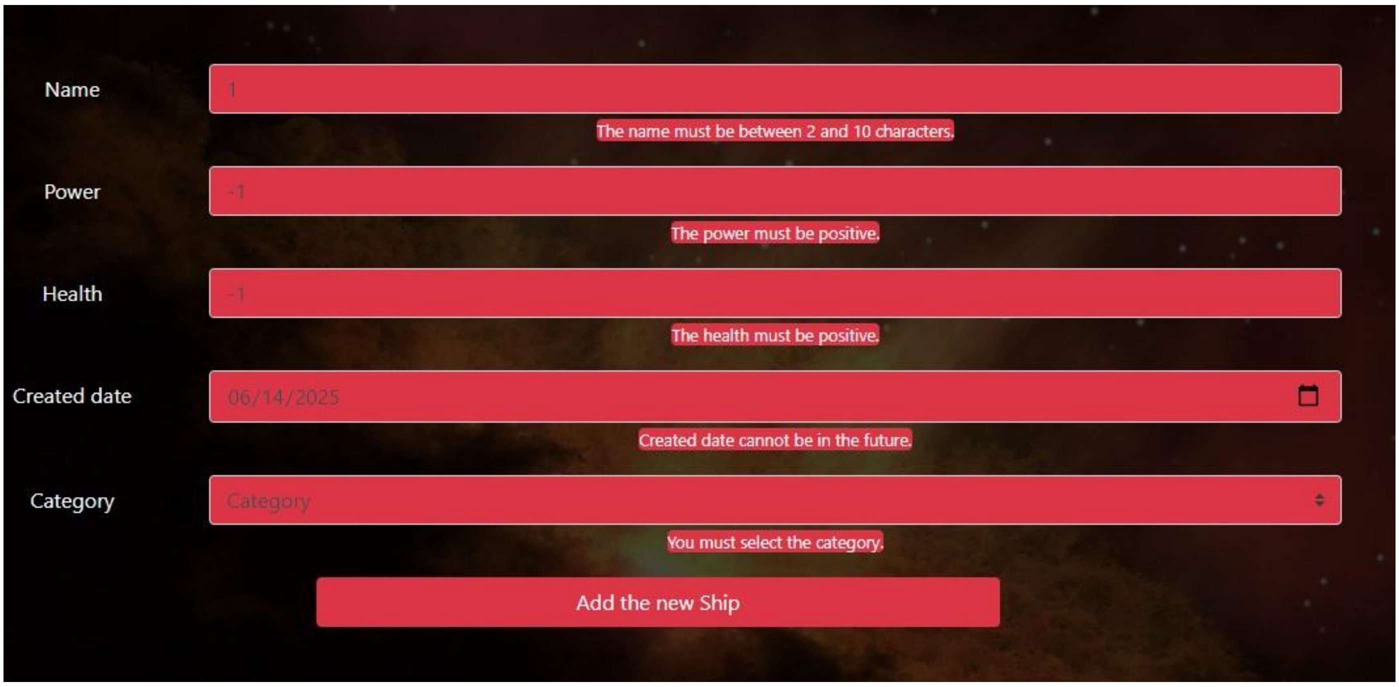
Add New Ship

Name	Name
Power	
Health	
Created date	mm/dd/yyyy
Category	Category

Add the new Ship

©SoftUni Spring Team 2021. All rights reserved.

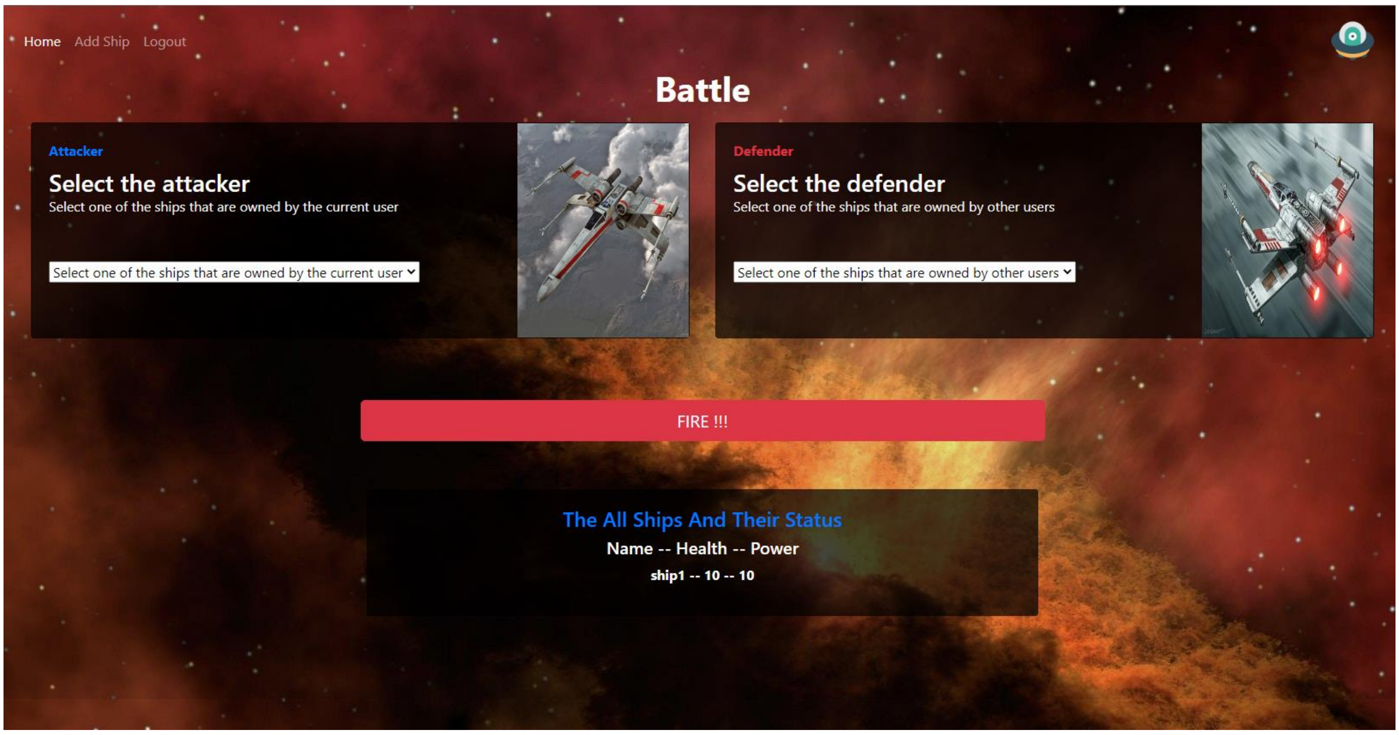
Add Ship validation



Name	1
	The name must be between 2 and 10 characters.
Power	-1
	The power must be positive.
Health	-1
	The health must be positive.
Created date	06/14/2025
	Created date cannot be in the future.
Category	Category
	You must select the category.

Add the new Ship

Home Page


Home
Add Ship
Logout
Battle
Attacker
Select the attacker
Select one of the ships that are owned by the current user
Select one of the ships that are owned by the current user
Defender
Select the defender
Select one of the ships that are owned by other users
Select one of the ships that are owned by other users
FIRE !!!
The All Ships And Their Status
Name -- Health -- Power
ship1 -- 10 -- 10

NOTE: You must select **one** of the ships that are **owned** by the **current user**.

NOTE: You must select **one** of the ships that are **owned** by **other users**.

NOTE: In the last section you should list all the ships **ordered by** their status (name, health, power) in the following format:

`name -- health -- power`.

NOTE: When pressing the **fire button**, the **attacker hits** the **defender** and **reduces** his health by the value of the attacker's **power**. If the defender's health is **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.

NOTE: The templates should look **EXACTLY** as shown above.

NOTE: 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 to **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 **fire** at other ships.

In the **BattleShips Application**, the navbar 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 **Fire** 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.
- **`Guest`** (not logged in) users can access **`Login`** page.
- **`Guest`** (not logged in) users can access **`Register`** page.
- **`Users`** (logged in) can access **`Home`** page.
- **`Users`** (logged in) can access **`Add Ship`** page.
- **`Users`** (logged in) can access **`Logout`** functionality.

6. Scoring

Database – 10 points.

Pages – 25 points.

Functionality – 35 points.

Security – 5 points.

Validations – 15 points.

Code Quality – 10 points.