# Spring Fundamentals Exam Preparation 18 June 2024

# Bon App-étit

Exam for the ["Spring Fundamentals" course @ SoftUni](https://judge.softuni.org/Contests/4469/Spring-Fundamentals-Retake-Exam-14-December-2023).

*Introducing the* ***Bon App-étit****, your culinary companion for exploring new tastes and connecting with  
fellow food enthusiasts. This app is all about taking your cooking experience to the next level by offering a  
wide range of recipes to suit your palate. Browse through hundreds of innovative recipes created by passionate  
chefs and home cooks. You can add your favorite dishes to your personal recipe collection, share your own  
culinary creations, and discover exciting flavor combinations. Unleash your inner chef and embark on a  
gastronomic journey with Bon App-étit!*

There are several requirements you must follow in the implementation:

## Database Requirements

The **Database** of the **Bon App-étit** application needs to support **3 entities**:

### User

* Has an **Id – "UUID-String" or Long**
* Has a Username (unique, not null)
  + Username length must be between 3 and 20 characters (inclusive of 3 and 20).
* Has a Password (not null)
  + Password length must be between 3 and 20 characters (inclusive of 3 and 20).
* Has an Email (unique, not null)
  + Must contain '@'.
* Has an addedRecipes
  + A collection that contains all recipes that the user added. One user may have many recipes and one recipe can be added by only one user.
* Has a favouriteRecipes
  + A collection that contains recipes that the user marked as favourite. One user may have many recipes and one recipe can be marked as favourite by many users.

### Recipe

* Has an **Id – "UUID-String" or Long**
* Has a Name (not null)
  + Name length must be between 2 and 40 characters (inclusive of 2 and 40).
* Has Ingredients
  + Translation length must be between 2 and 150 characters (inclusive of 2 and 150).
* Has a Category (not null)
  + One recipe has one caegory and one category can have many recipes.
* Has an addedBy
  + The user who has added the recipe in the app. One recipe has one user, but one user may have added many recipe.

### Category

* Has an **Id** – **"UUID-String" or Long**
* Has a **Category name** (unique, not null)
  + an option between (MAIN\_DISH, DESSERT, COCKTAIL)
* Has a Description
  + For MAIN\_DISH - "Heart of the meal, substantial and satisfying; main dish delights taste buds."
  + For DESSERT – "Sweet finale, indulgent and delightful; dessert crowns the dining experience with joy."
  + For COCKTAIL – "Sip of sophistication, cocktails blend flavors, creating a spirited symphony in every glass."
* Has collection of **Recipes**
  + One category may **have many** recipes, but one recipe has **only one** category.

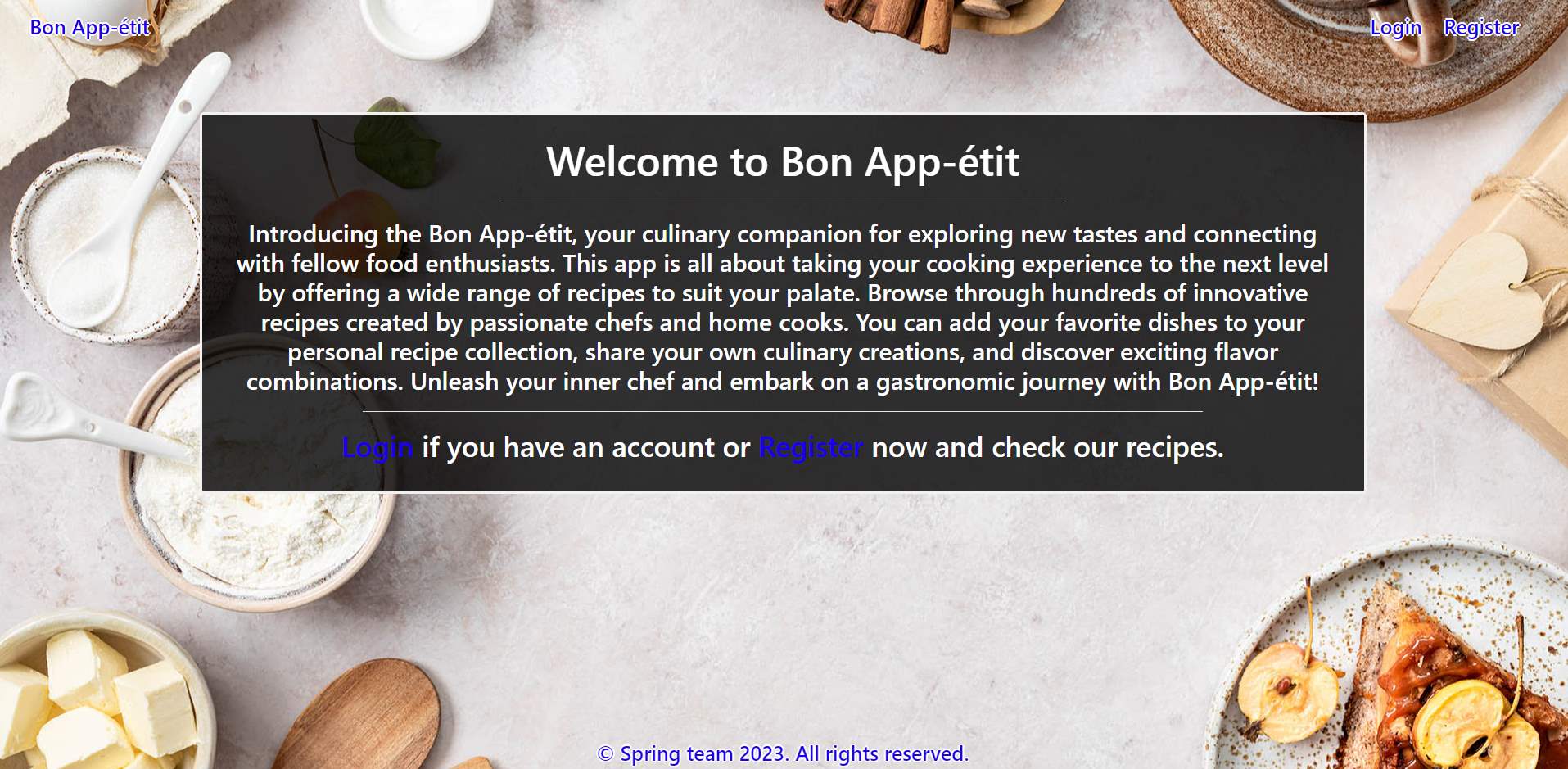
Implement the entities with the **correct data types** and implement **repositories** for them.

## Initialize categories

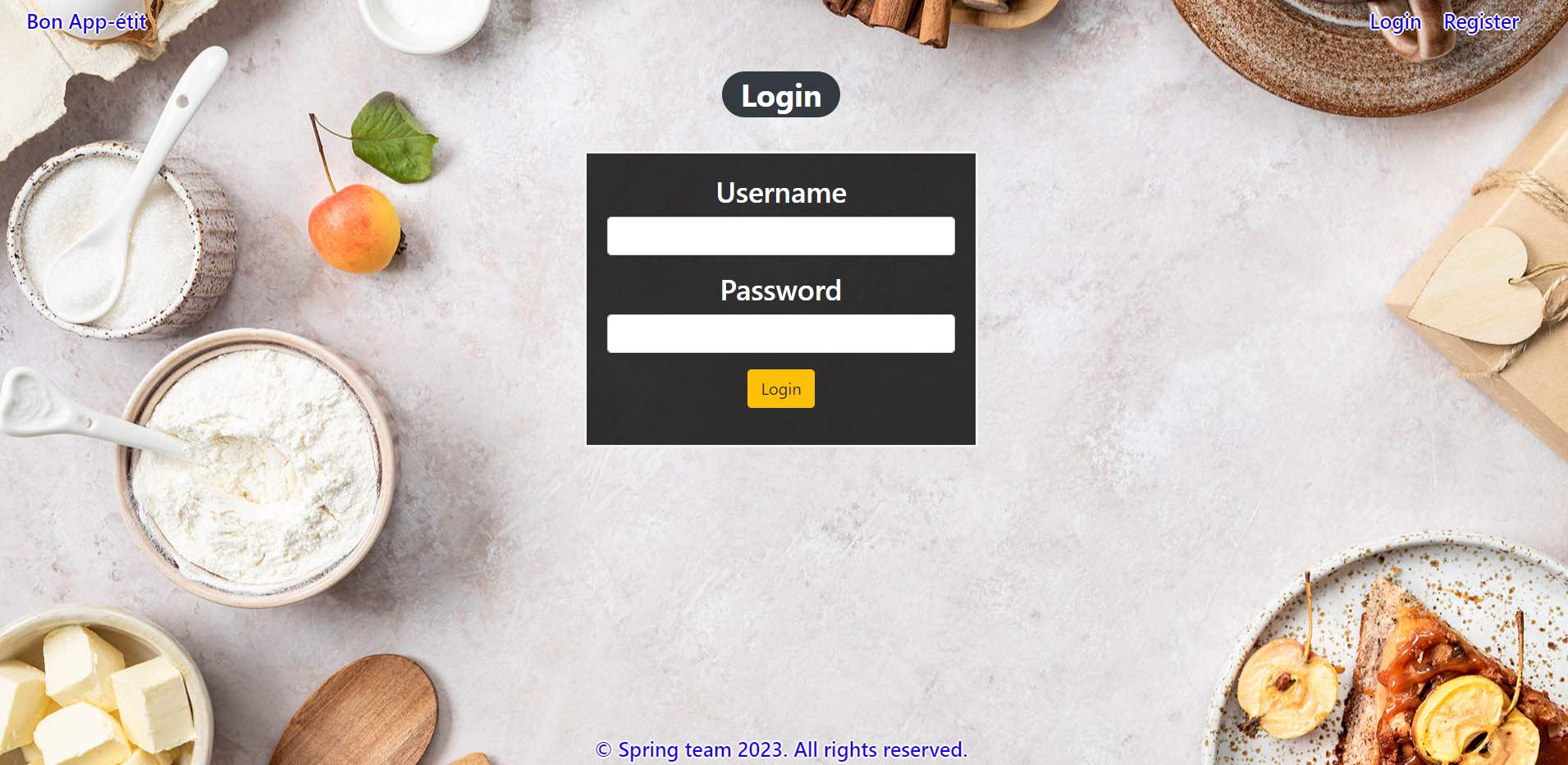
* Implement a method that checks (when the app started) if the database does not have **categories** and   
  initialize them
  + **You are free to do it in different ways.**

## Page Requirements

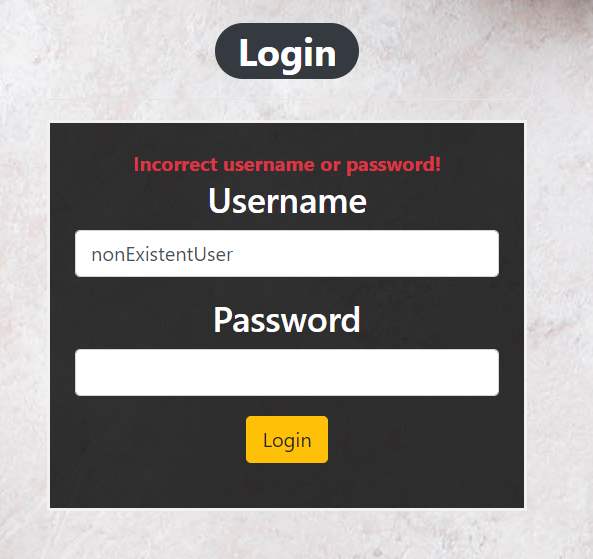
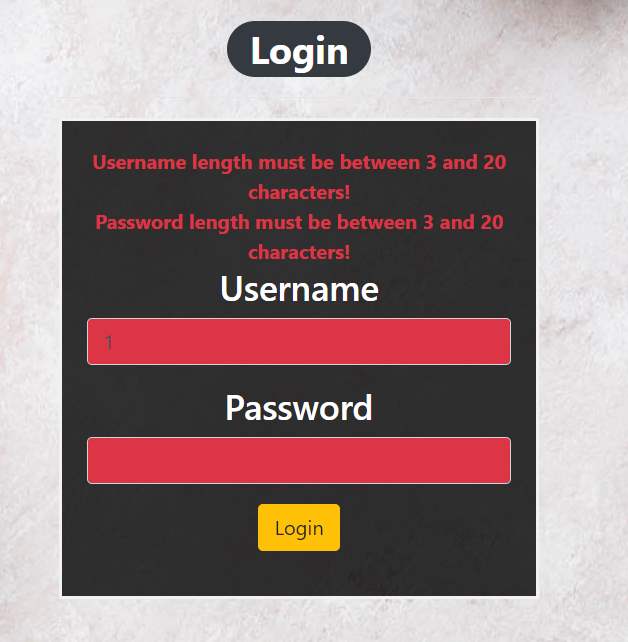
### Index Page (logged out user)



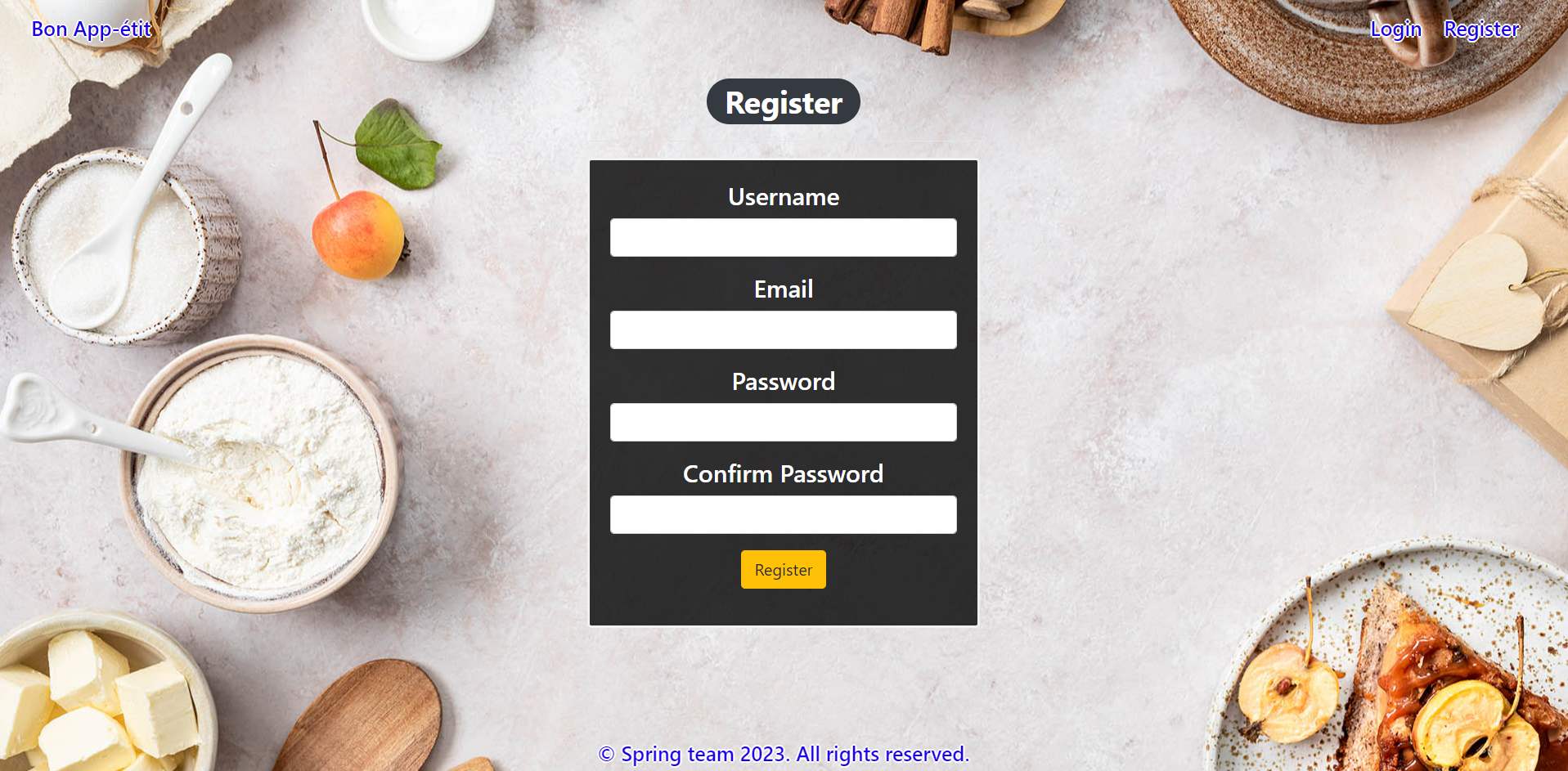
### Login Page (logged out user)



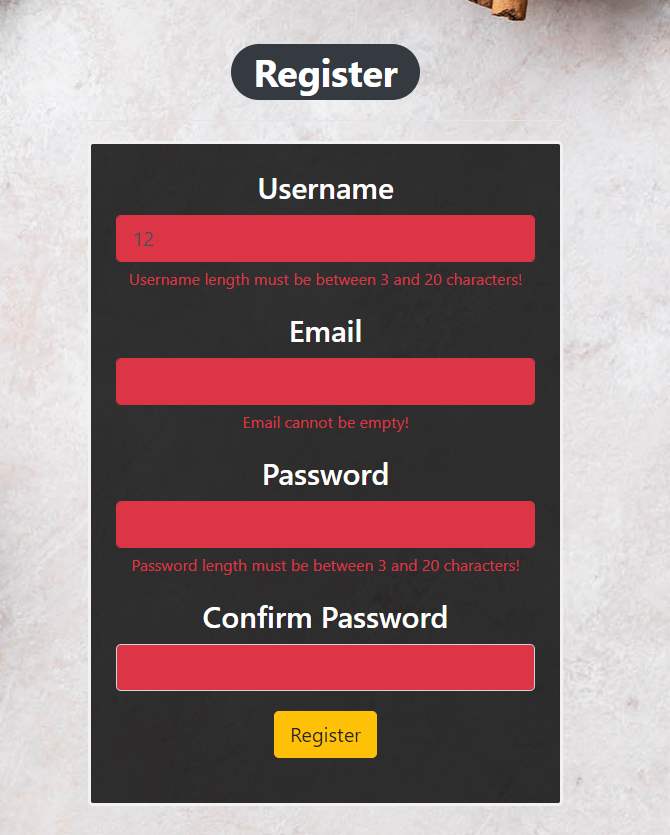
### Login Page validations

### Register Page (logged out user)



### Register Page validations

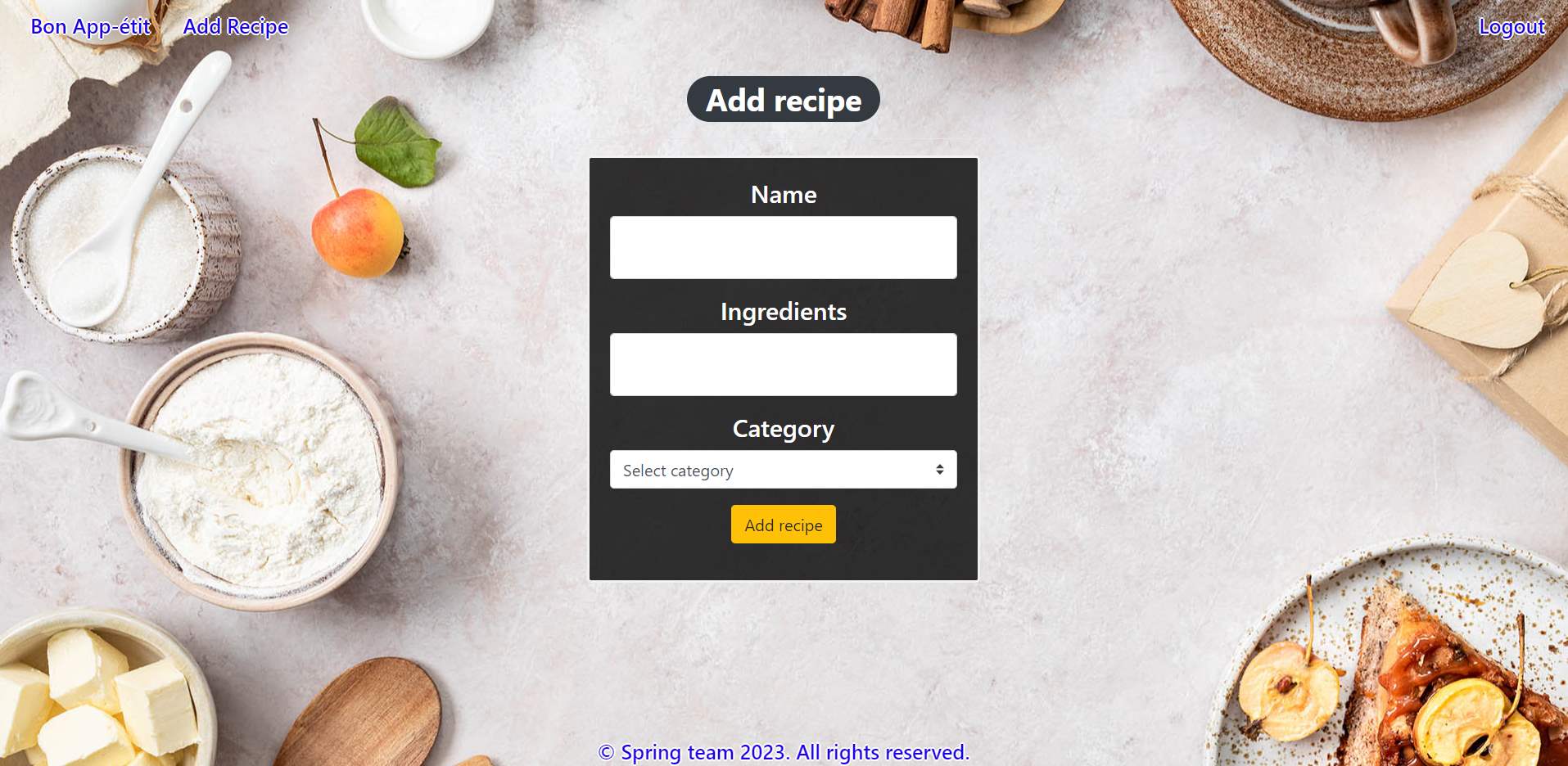


### Home Page (without having any recipes)

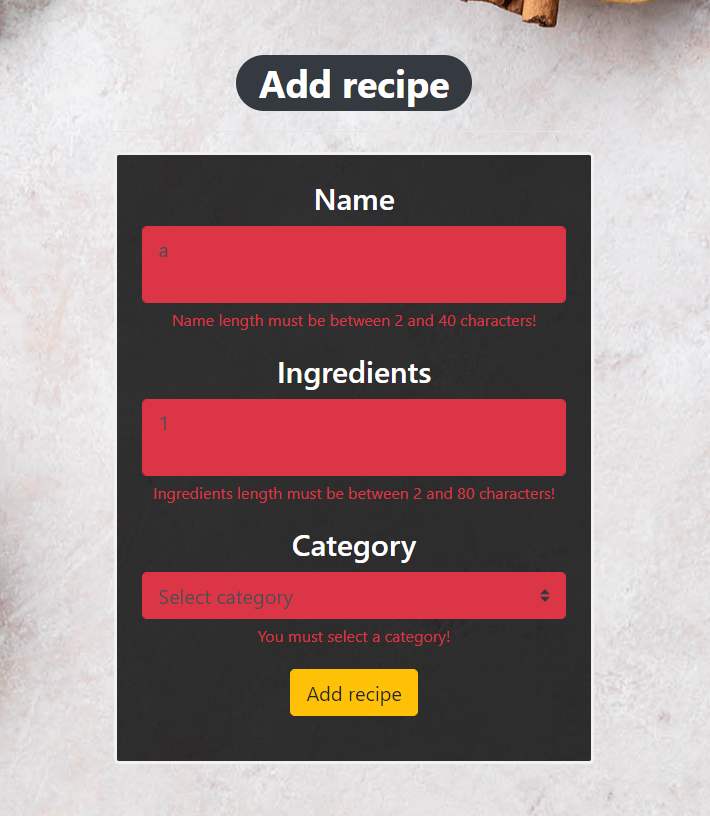
* The upper left section (Desserts) of the page should display **all of the added recipes for Desserts** from the database from all users.
* The upper right section (Main Dishes) of the page should display **all of the added recipes for Main Dishes** from the database from all users.
* The bottom left section (Cocktails) of the page should display **all of the added recipes for Cocktails** from the database from all users.
* The bottom right section (Favourites) of the page should display **all recipes marked as favorites** in the database by users.



### Add Recipe



### Add Recipe validation



### Home Page (with recipes)



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 **bootstrap** and the **given CSS** are enough.

## 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 log in, register and view the Index page.

The application should provide **Users** (logged in) with the functionality to **log out, add a new recipe (Add Recipe page), view all recipes (Home page) and add recipes to Favourites section.**

Bon App-étit in navbar should **redirect** to the appropriate URL depending on that if the **user is logged in**.

The application should provide functionality for **adding recipes** with category of **Main Dish, Dessert or Cocktail**.

On the **Home Page**, recipes should be **organized** into their respective categories: main dish recipes in the **Main Dishes** section, dessert recipes in the **Desserts** section, cocktail recipes in the **Cocktails** section, and favorite recipes in the **Favorites** section.

**Above** these sections, a **greeting text** displaying the current user's username is located: "Welcome, {username}".

Next to the name of each section on the **Home page** is a **counter** indicating the **number of recipes** in the current section.

Each recipe is accompanied by a **star with a plus icon**. When **clicked**, the selected recipe stays in the current section but is also **added** to the current user's **favourite recipes** collection. If the recipe is already present in the favorites collection, no action should be taken.

The application should store its data in a MySQL database.

## 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 the Index page.
* Guest (not logged in) users can access the Login page.
* Guest (not logged in) users can access the Register page.
* Users (logged in) can access the Home page.
* Users (logged in) can access Add Recipe page.
* Users (logged in) can access Logout functionality.

## Scoring

### Database – 10 points.

### Pages – 25 points.

### Functionality – 35 points.

### Security – 5 points.

### Validations – 15 points.

### Code Quality – 10 points.