



Shipping List Application

Regular Exam [Spring Fundamentals]

The **Shipping List** Application is here to help us keep in mind our shopping needs. The functionality is simple. When a user thinks of something important, they log in and add it to existing ones. So, when a person goes to the store, they have a clear idea of exactly what to buy. So, our little app saves a lot of family scandals.

There are several requirements you must follow in the implementation:

1. Database Requirements

The **Database** of the **Shipping List** 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 20** characters long (both numbers are **INCLUSIVE**)
 - The values should be **unique** in the database
- **Password**
 - The **length** of the **values** should be **between 3 and 20** characters long (**INCLUSIVE**)
- **Email**
 - The values should contain a '@' symbol)
 - The values should be **unique** in the database

Product

- **Id** – Accepts **UUID-String** or **Long** values
- **Name**
 - The **length** of the **values** should be **between 3 and 20** characters long (both numbers are **INCLUSIVE**)
 - The values should be **unique** in the database
- **Description**
 - The **length** of the **values** should be **at least 5** characters long
- **Price**
 - The values must be a **positive** numbers
- **Needed Before**
 - **Date and Time** values, that **cannot** be in the **future**
- **Category**
 - One product **has one** category and one category **can have many** products

Category

- **Id** – Accepts **UUID-String** or **Long** values
- **Style name**
 - The values should be **unique** in the database
 - an option between (**FOOD, DRINK, HOUSEHOLD** and **OTHER**)
- **Description**
 - Fell free to add some description to every classification

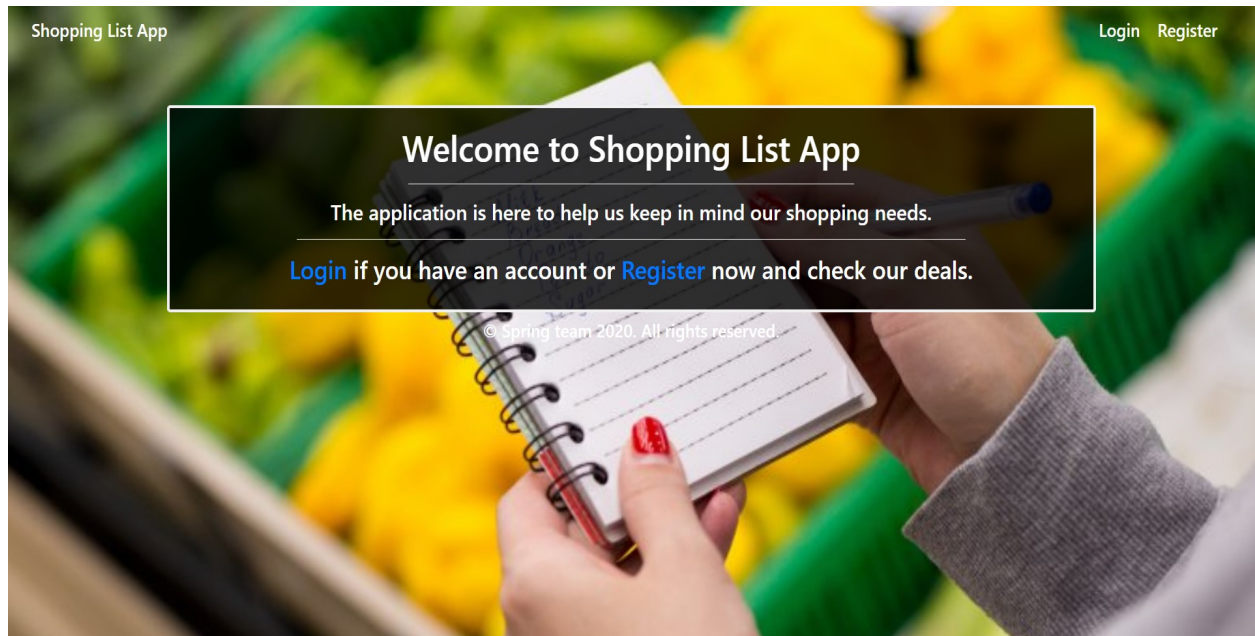
Nullable/Empty/Blank values are not allowed unless explicitly mentioned. Implement the entities with the **correct data types** and implement the **repositories** for them.

2. Initialize categories

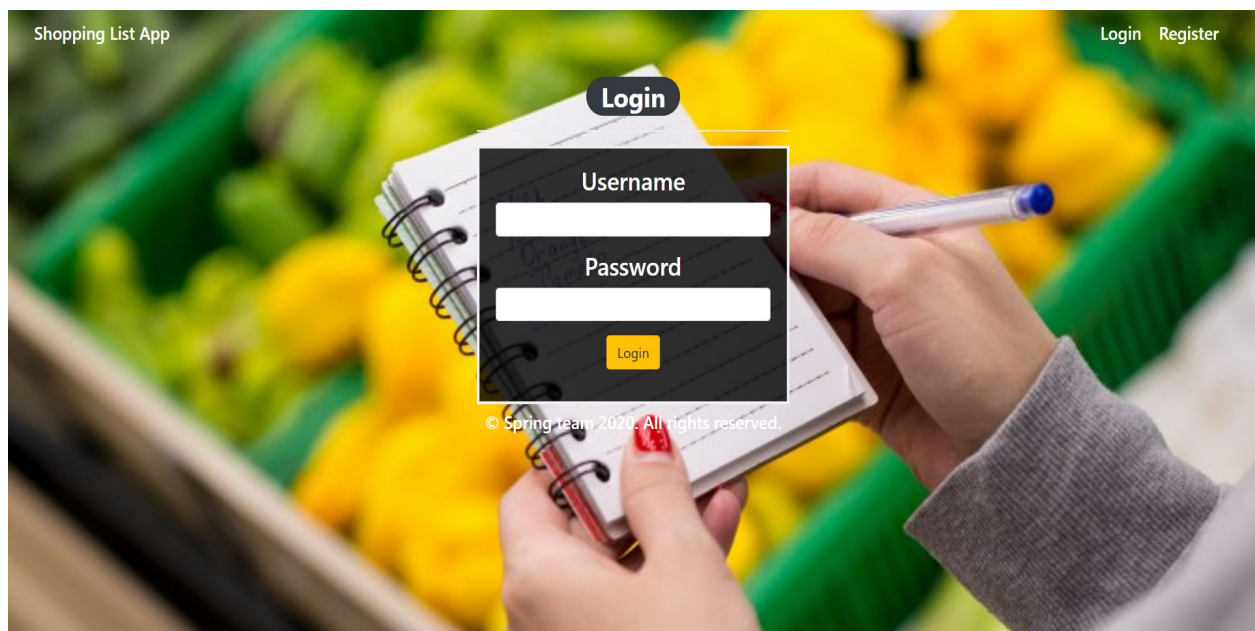
- Implement a method that checks (when app started) if the database does not have any category and initialize them
 - You are free to do this in some different ways
 - You can skip the description if you want

3. Page Requirements

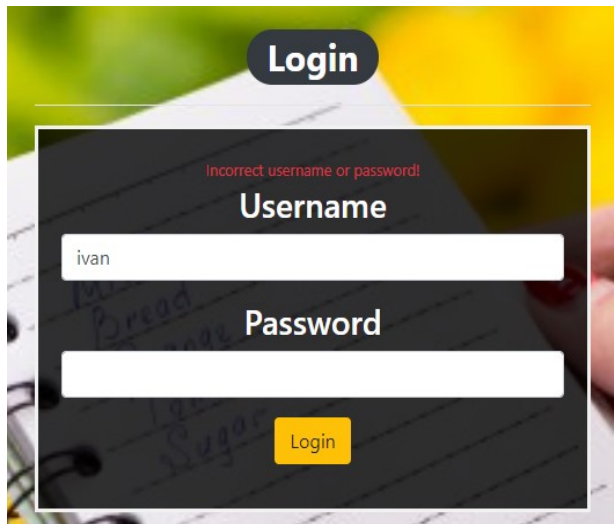
Index Page (logged out user)



Login Page (logged out user)



Login Page validations



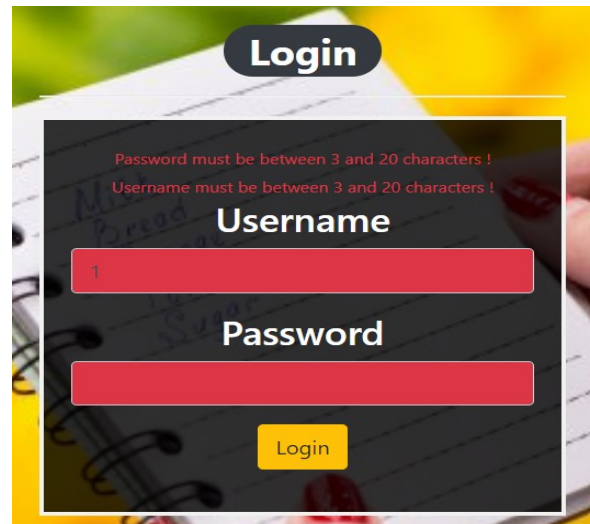
Login

Incorrect username or password!

Username

Password

Login



Login

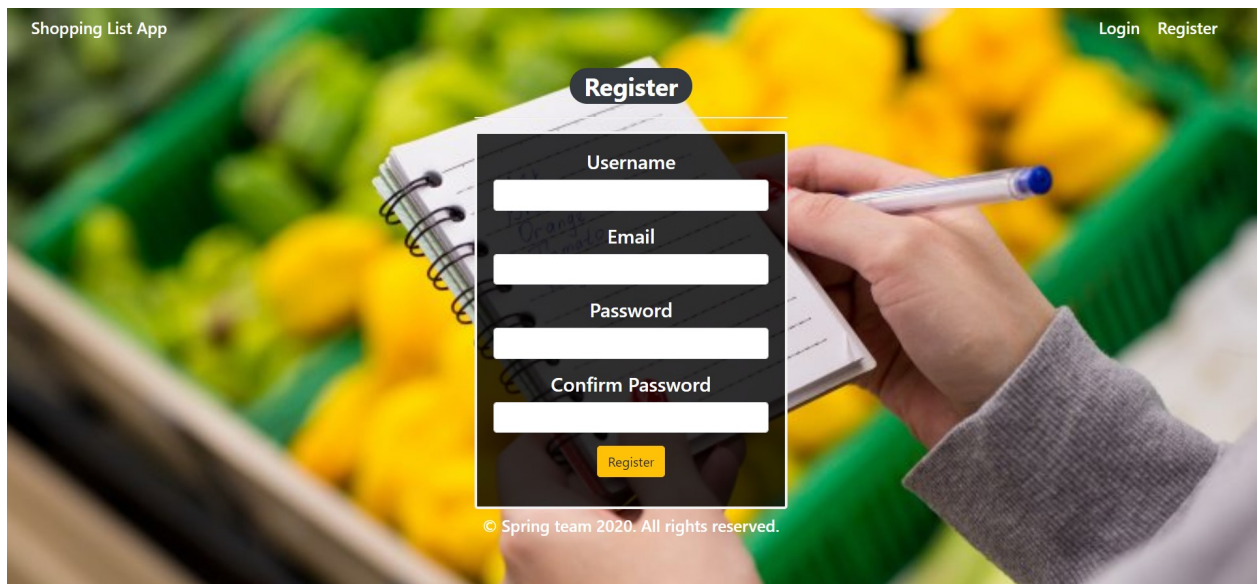
Password must be between 3 and 20 characters !
Username must be between 3 and 20 characters !

Username

Password

Login

Registration Page (logged out user)



Shopping List App Login Register

Register

Username

Email

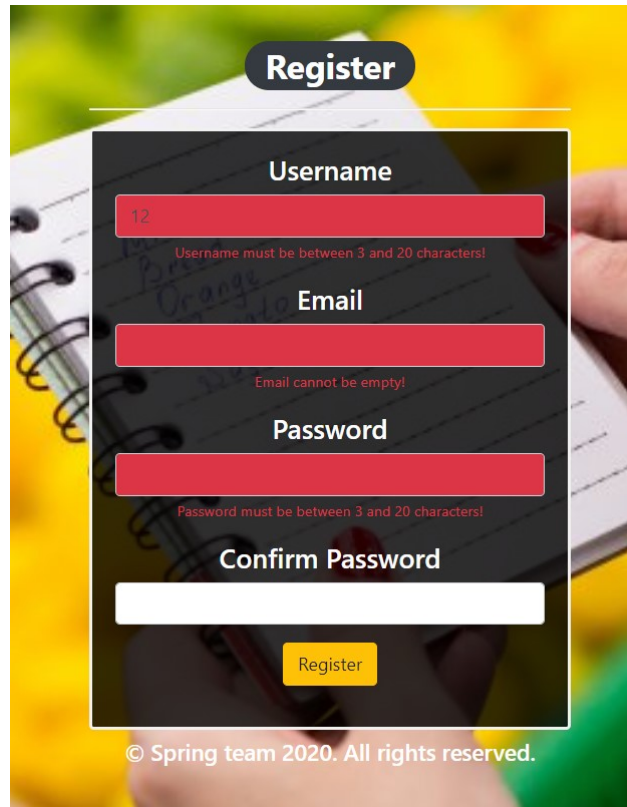
Password

Confirm Password

Register

© Spring team 2020. All rights reserved.

Registration Page validations



The registration form is titled "Register" and is set against a background of a notebook with handwritten notes. It contains four input fields: "Username" (containing "12" with a red error message "Username must be between 3 and 20 characters!"), "Email" (empty with a red error message "Email cannot be empty!"), "Password" (empty with a red error message "Password must be between 3 and 20 characters!"), and "Confirm Password" (empty). A yellow "Register" button is at the bottom. The footer text reads "© Spring team 2020. All rights reserved."

Register

Username

12

Username must be between 3 and 20 characters!

Email

Email cannot be empty!

Password

Password must be between 3 and 20 characters!

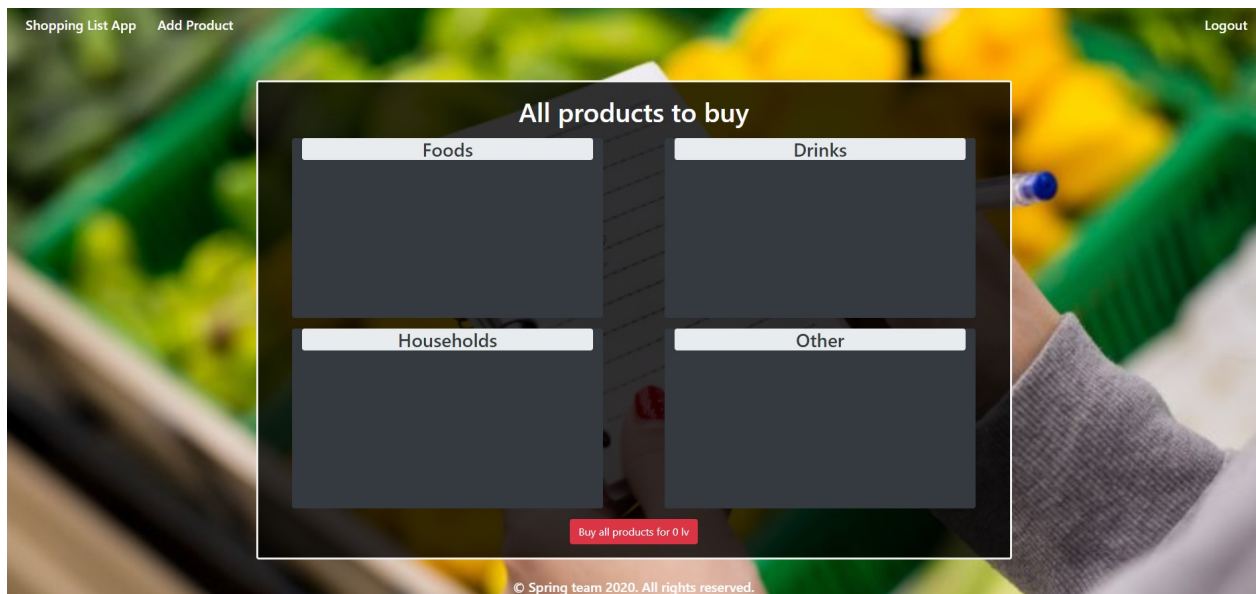
Confirm Password

Register

© Spring team 2020. All rights reserved.

Home Page (without having any products)

- The home page should visualize **all of the products** from the database.



The home page layout features a top navigation bar with "Shopping List App" and "Add Product" on the left, and "Logout" on the right. The main content area is titled "All products to buy" and contains four empty rectangular boxes labeled "Foods", "Drinks", "Households", and "Other". A red button at the bottom of the main area says "Buy all products for 0 lv". The footer text reads "© Spring team 2020. All rights reserved."

Shopping List App Add Product Logout

All products to buy

Foods

Drinks

Households

Other

Buy all products for 0 lv

© Spring team 2020. All rights reserved.

Add products

Shopping List App Add Product Logout

Add product

Name

Description

Before

Price

Category

Add product

© Spring team 2020. All rights reserved.

Add Offer Validation

Add product

Name

Name length must be between 3 and 20 characters

Description

Description length must be more than 5 characters

Before

The date cannot be in the past

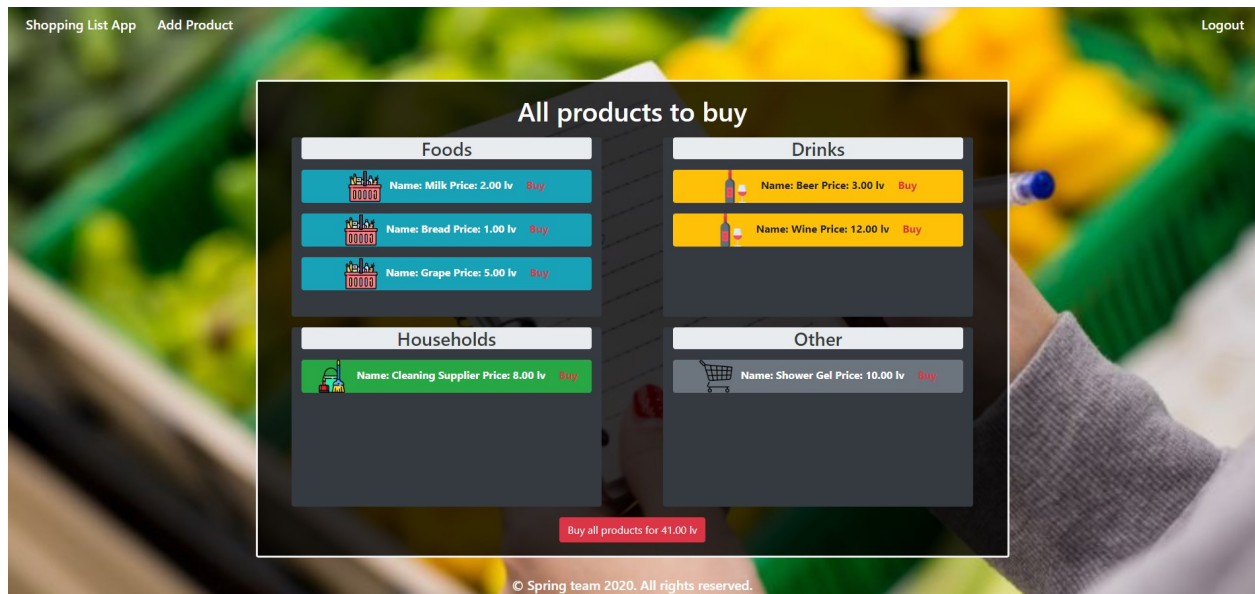
Price

Price must be positive number

Category

Add product

Home Page (with products)



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

NOTES:

- The templates should look **EXACTLY** as shown above.
- The templates do **NOT** require additional **CSS** for you to write. Only **bootstrap** and the **given 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 log in, register and view the Index page.

The application should provide Users (logged in) with the functionality to **log out**, **add a new product** (Add product page), **view all products** (Home page) and **Buy a single product** or **Buy all products**.

Shopping List App in navbar should redirect to appropriate **URL depending** on that if the user is logged in.

The **application** should provide **functionality** for **adding products** with **category** (FOOD, DRINK, HOUSEHOLD or OTHER) and **buying** one or more of them.

The **Buy** all products button shows the **sum** of **all added products** prices. In addition to that you can display a total sum for all products grouped by a category (not required - only optional).

The **product** should be separated into different divs according to their categories. The **image** also depends on the item's category.

When the user clicks on the **Buy button** of some item, he buys it. You need to **delete** this **item** and **redirect** it to the **home** page. When he clicks on **Buy all** products, just **delete all** products in DB and again **redirect** to the **home** page.

The application should store its data in 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 Product** page;
 - **Logout** functionality.

6. Scoring

Database – 10 points

Pages – 25 points

Functionality – 35 points

Security – 5 points

Validations – 15 points

Code Quality – 10 points