# Spring Fundamentals Exam

# Reseller Application

Exam for the ["Spring Fundamentals" course @ SoftUni](https://judge.softuni.org/Contests/3883/Spring-Fundamentals-Exam-18-Feb-2023).

**ResellerApp** is the newest online marketplace dedicated to buying and selling new or used items. At any given time, the site has a wide variety of listings ranging from hand tools, clothes and books to accessories, pro audio gear, bikes, smartphones, and more…

There are several requirements you must follow in the implementation:

## Database Requirements

The **Database** of the **Reseller** 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 Offers
  + The offers contains users offers. One user may have many offers and one offer can be created by only one user.
* Has boughtOffers
  + Collection with the offers that the user bought. One user may have many offers in boughtOffers.

### Offer

* Has an **Id – “UUID-String” or Long**
* Has a Description (not null)
  + Description length must be between 2 and 50 characters (inclusive of 2 and 50).
* Has a Price (not null) - floating point number
  + The price must be a positive number
* Has a Condition (not null)
  + One offer has one contition and one condition can have many offers.

### Condition

* Has an **Id** – **“UUID-String” or Long**
* Has a **Condition name** (unique, not null)
  + an option between (EXCELLENT, GOOD, ACCEPTABLE)
* Has a Description (not null)
  + For EXCELLENT - "In perfect condition"
  + For GOOD – "Some signs of wear and tear or minor defects"
  + For ACCEPTABLE – "The item is fairly worn but continues to function properly"

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

## Initialize conditions

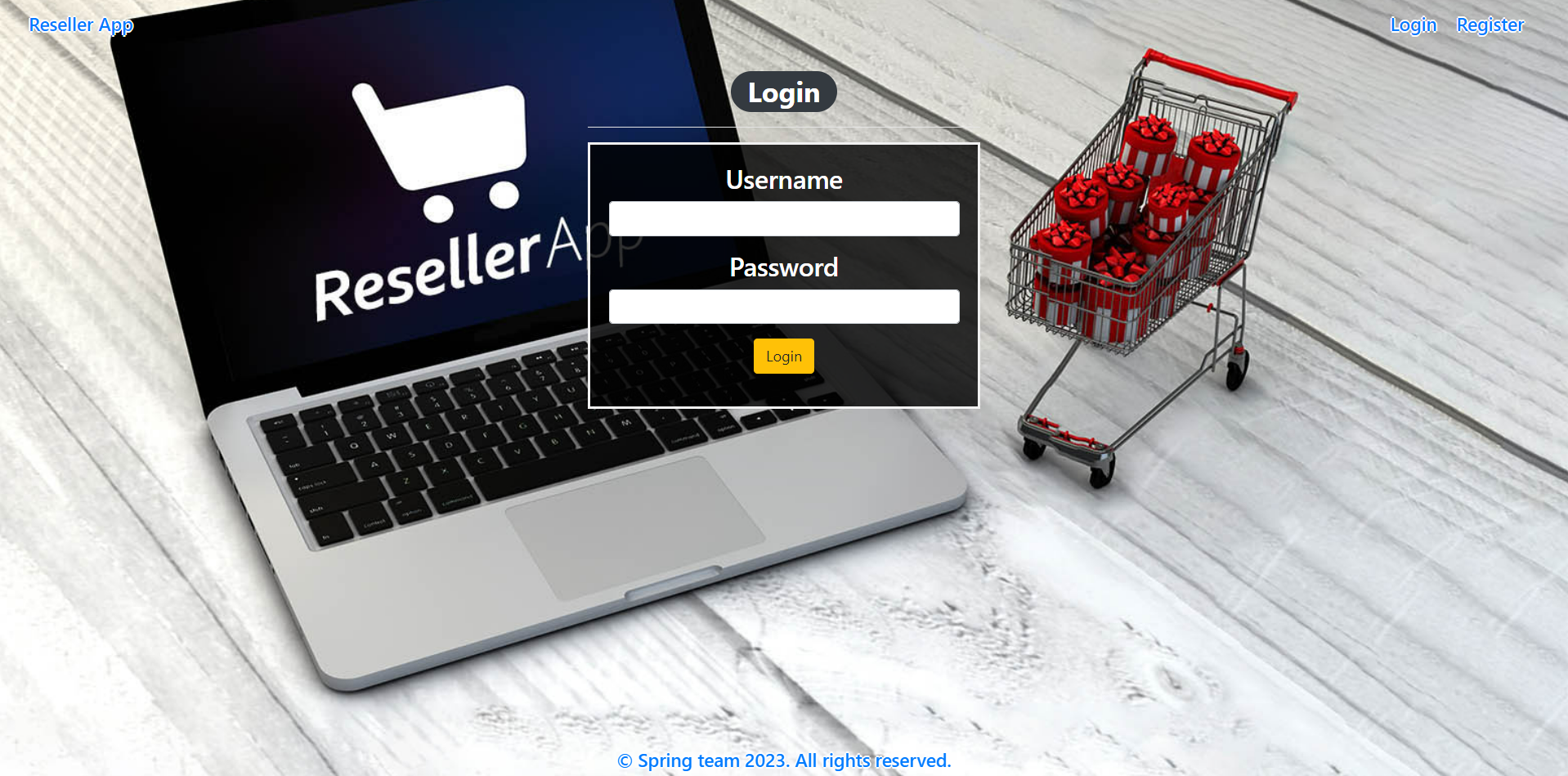
* Implement a method that checks (when the app started) if the database does not have conditions and   
  initialize them
  + **You are free to do it in different ways.**

## Page Requirements

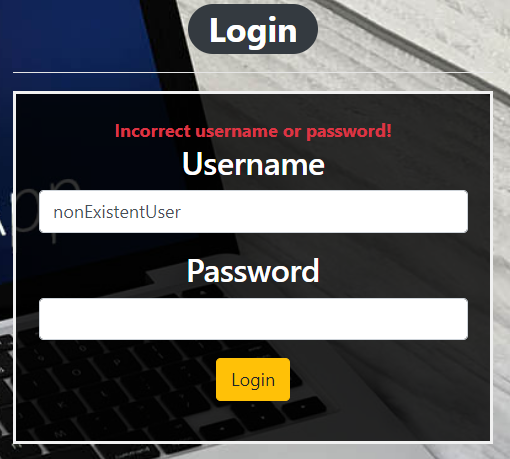
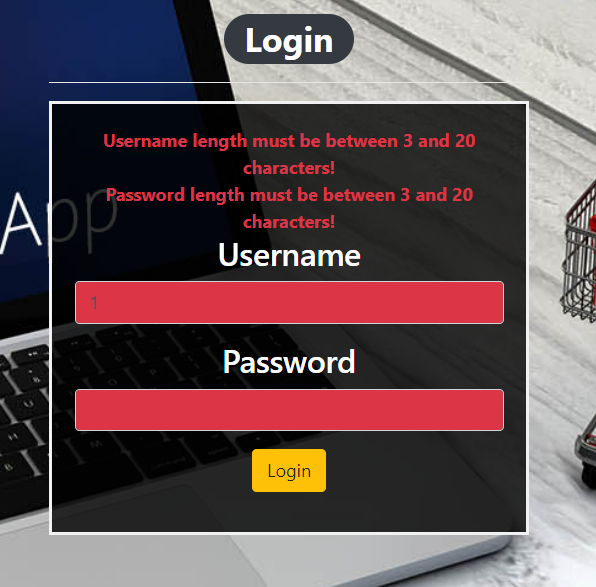
### Index Page (logged out user)



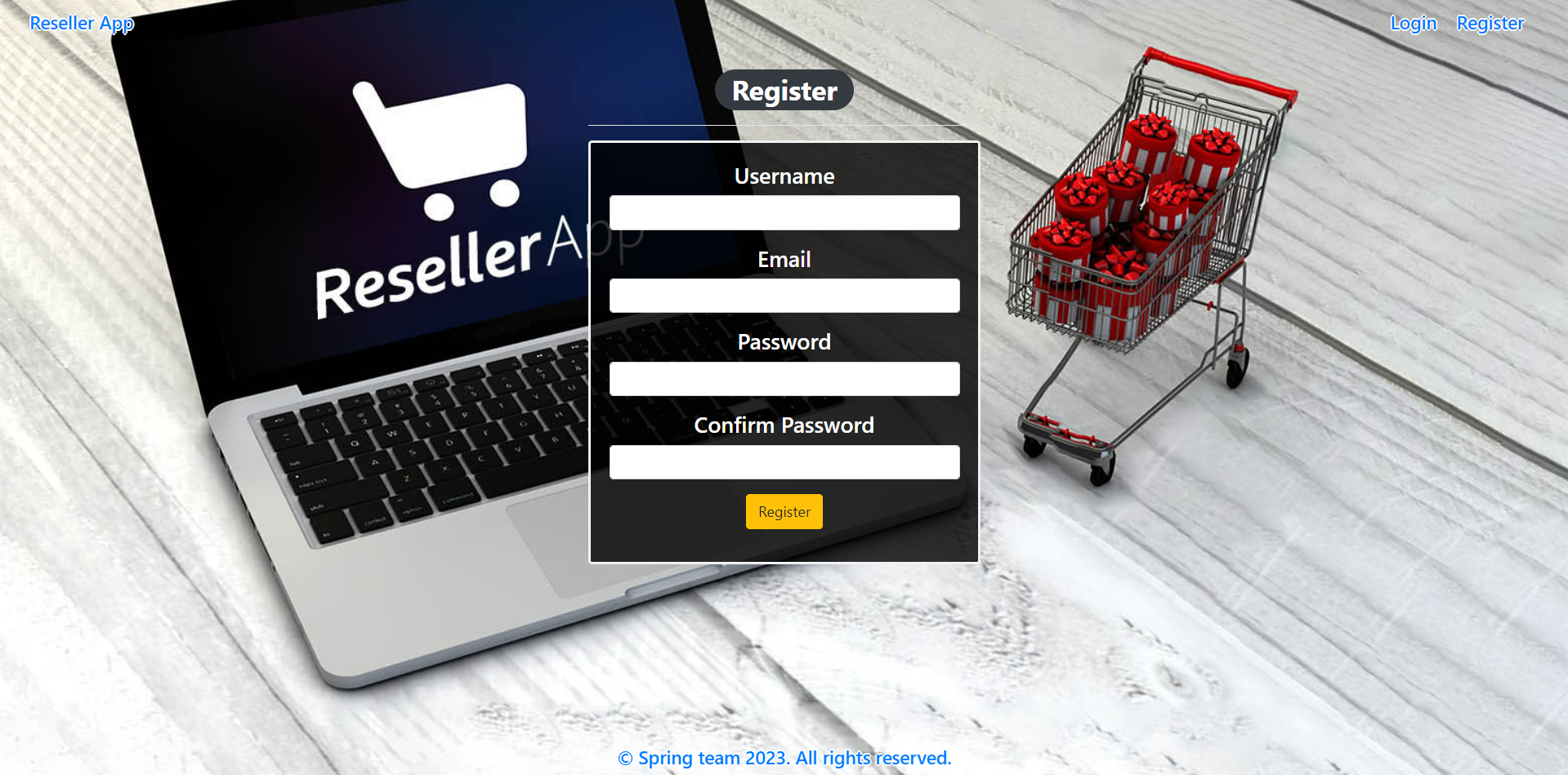
### Login Page (logged out user)



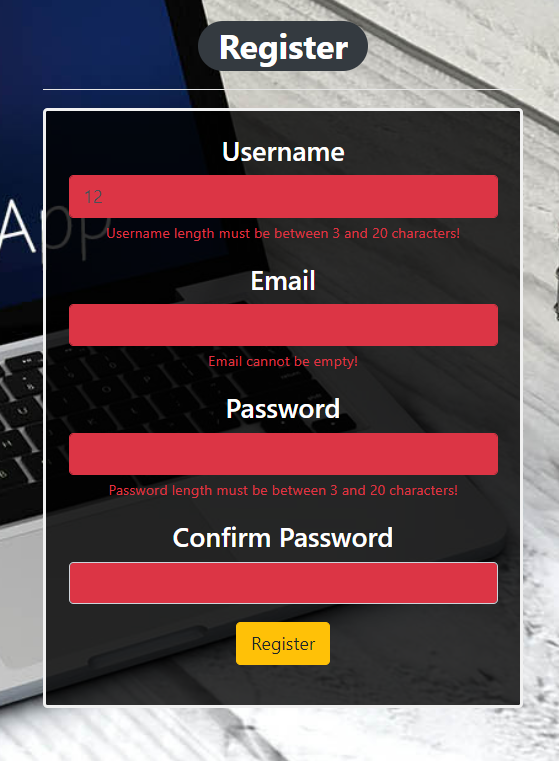
### Login Page validations

### Register Page (logged out user)

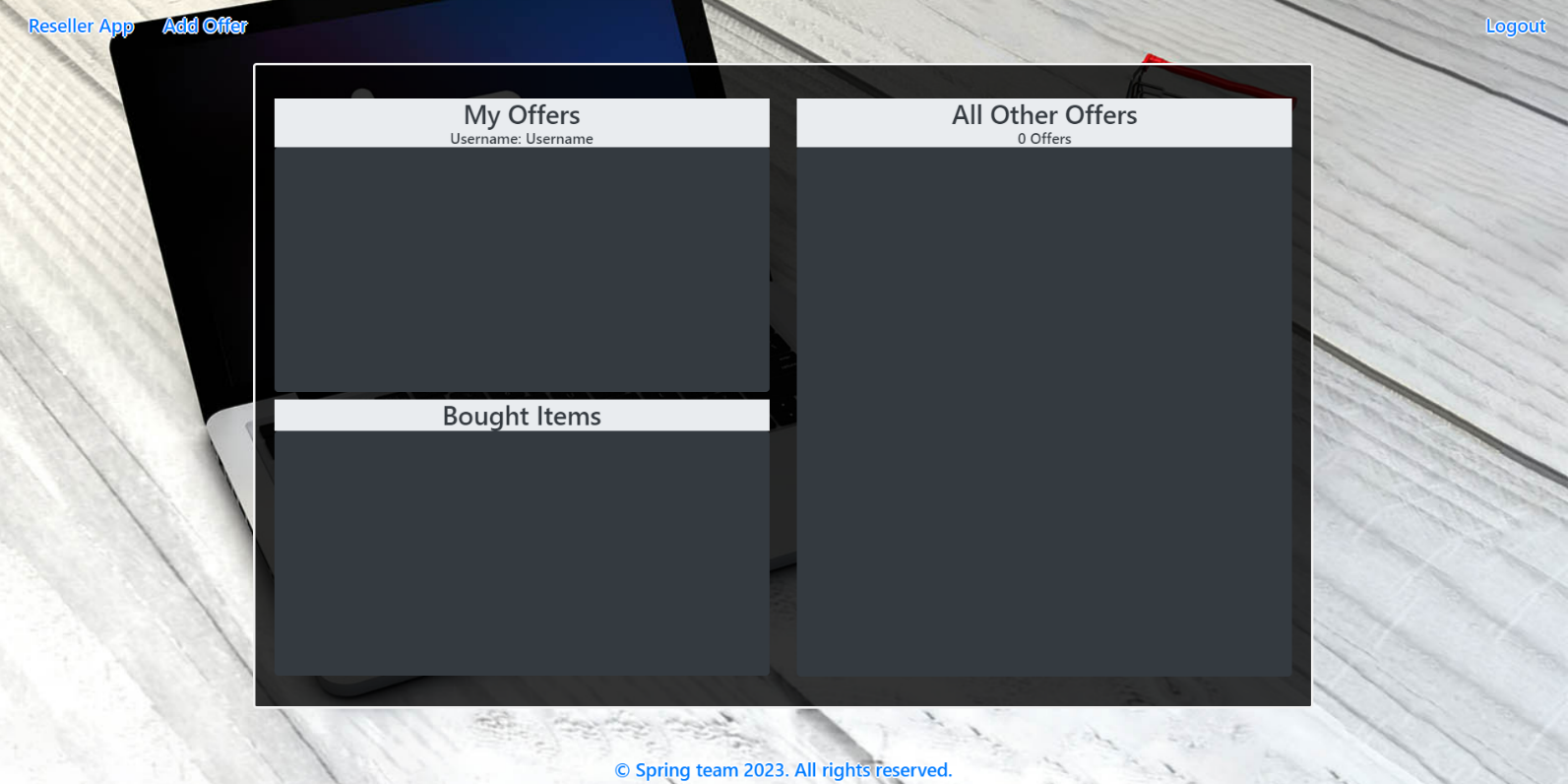


### Register Page validations

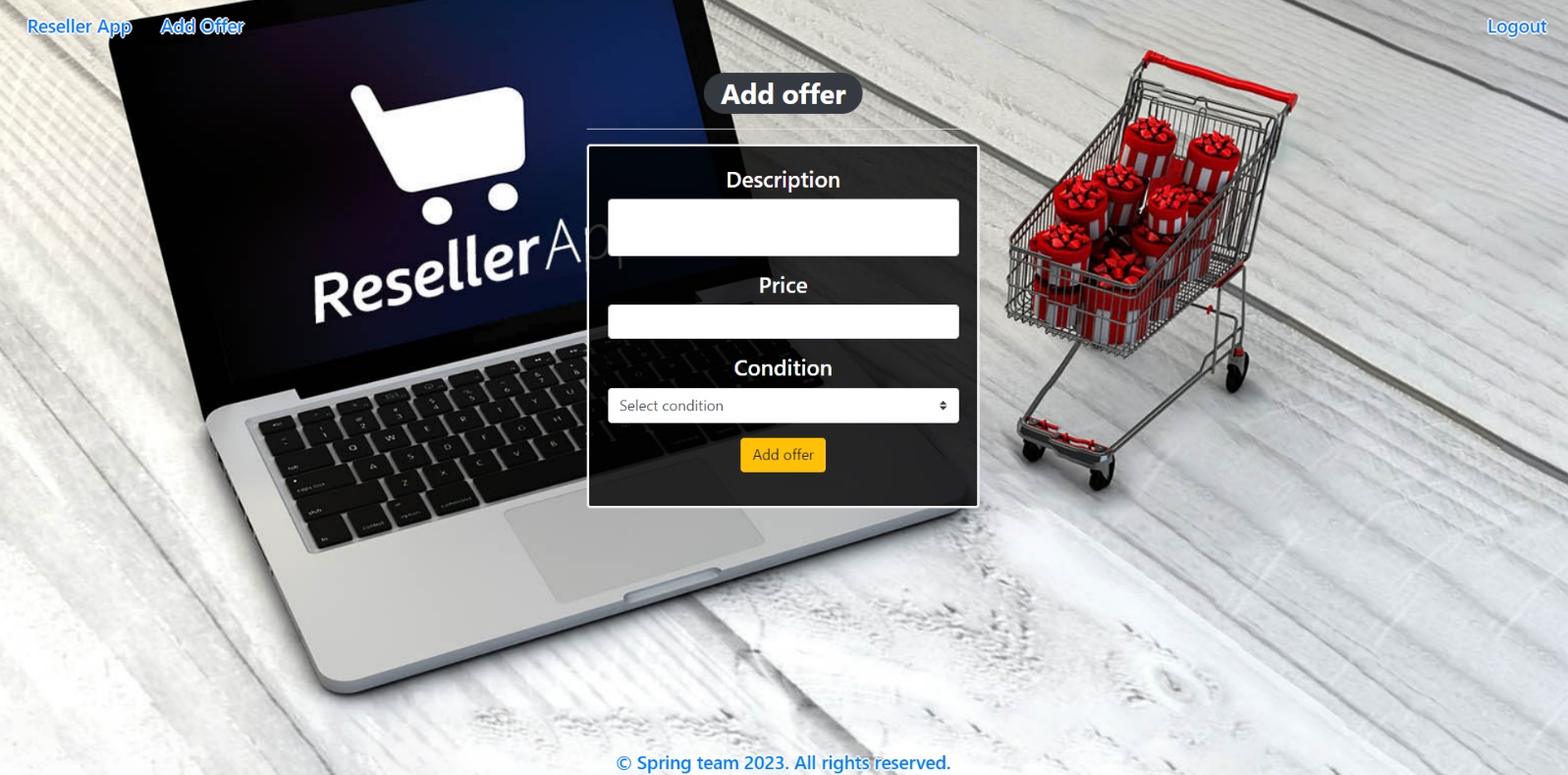


### Home Page (without having any offers)

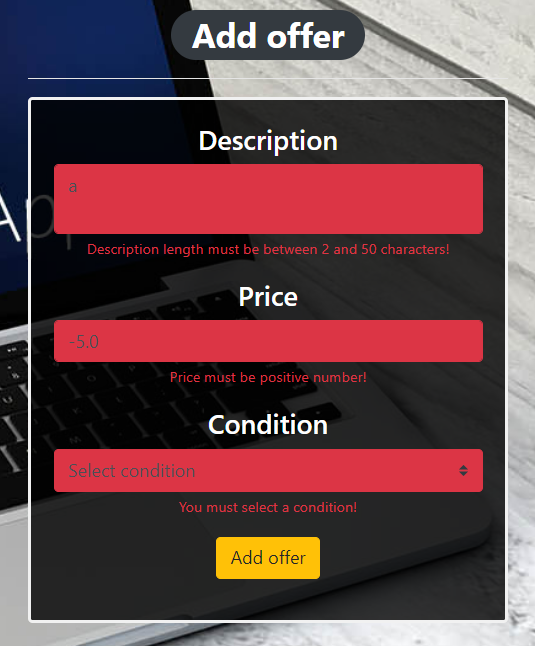
* Note: The left and upper section (My Offers) of the page should visualize **all of the current logged user offers** from the database which he added.
* Note: To the left and down section (Bought Items) of the page should visualize **the bought items(offers) by the current logged user from All Other Offers** section.
* Note: The right section (All Other Offers) of the page should visualize the available(not bought) **offers** from **all other users** (except the current logged one).



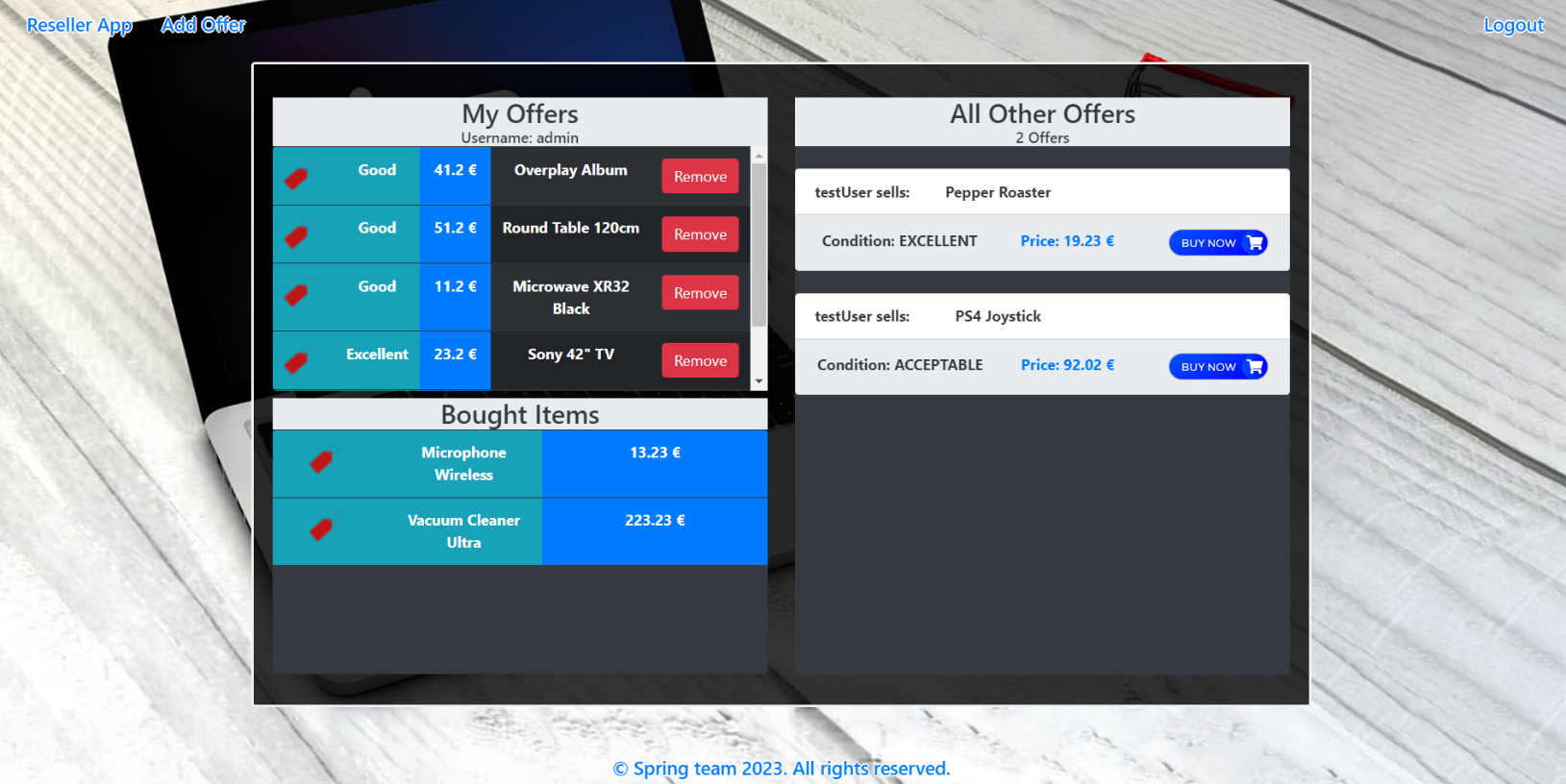
### Add offer



### Add offer validation



### Home Page (with offers)



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 offer (Add offer page), view all offers (Home page) and buy an offer from All Other Offers to Bought Items.**

Reseller Application in navbar should **redirect** to the appropriate URL depending on that if the **user is logged in**.

The application should provide functionality for **adding offers** with conditions of **Excellent, Good or Acceptable**.

The offers should be separated into different sections according to their ownership.

When the user clicks on the **BUY NOW button of some offer**, he buys the offer to **his Bought Items**. You should **not delete** this offer from DB. The offer should be removed from the collection with offers from the seller and should be added to the collection with boughtOffers of the buyer.

The **Remove** button in My Offers section **should remove the offer** from DB.

Bellow the All Other Offers banner is located an info bar that shows the **sum of the** **offers in this field**.

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 Offer 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.