# Spring Fundamentals

# Andrey's

Exam Preparation for the ["Spring Fundamentals" course @ SoftUni](https://softuni.bg/trainings/2844/spring-fundamentals-may-2020).

**Andrey's** is a shop application. Users can register and login from home page. When users register in the app, they see a different navigation bar with more actions and also see all the items from the database. Clicking on add item, they can see the item’s detail page. From the details page, they can delete the item if they want. There are several requirements you must follow in the implementation.

## Database Requirements

The **Database** of the **Andrey's** application needs to support **3 entities**:

### User

* Has an **Id – UUID-string**
* Has a Username (unique)
  + Username must be unique
  + Username length must be more than two characters
* Has a Password
  + Password length must be more than two characters
* Has an Email
  + Email must contain '@'
* Has a Budget
  + Must be a positive number

### Item

* Has an **Id – UUID-string**
* Has a Name (unique)
  + Username must be unique
  + Username length must be more than two characters
* Has a Description
  + Description length must be more than three characters
* Has a Price
  + Price must be positive number
* Has a Category
  + Has a relation with Categories
* Has a Gender – an option between (Male and Female)

### Category

* Has an **Id** – **UUID-string**
* Has a Name – **an** option between (Shirt, Denim, Shorts or Jacket)
* Has a Description

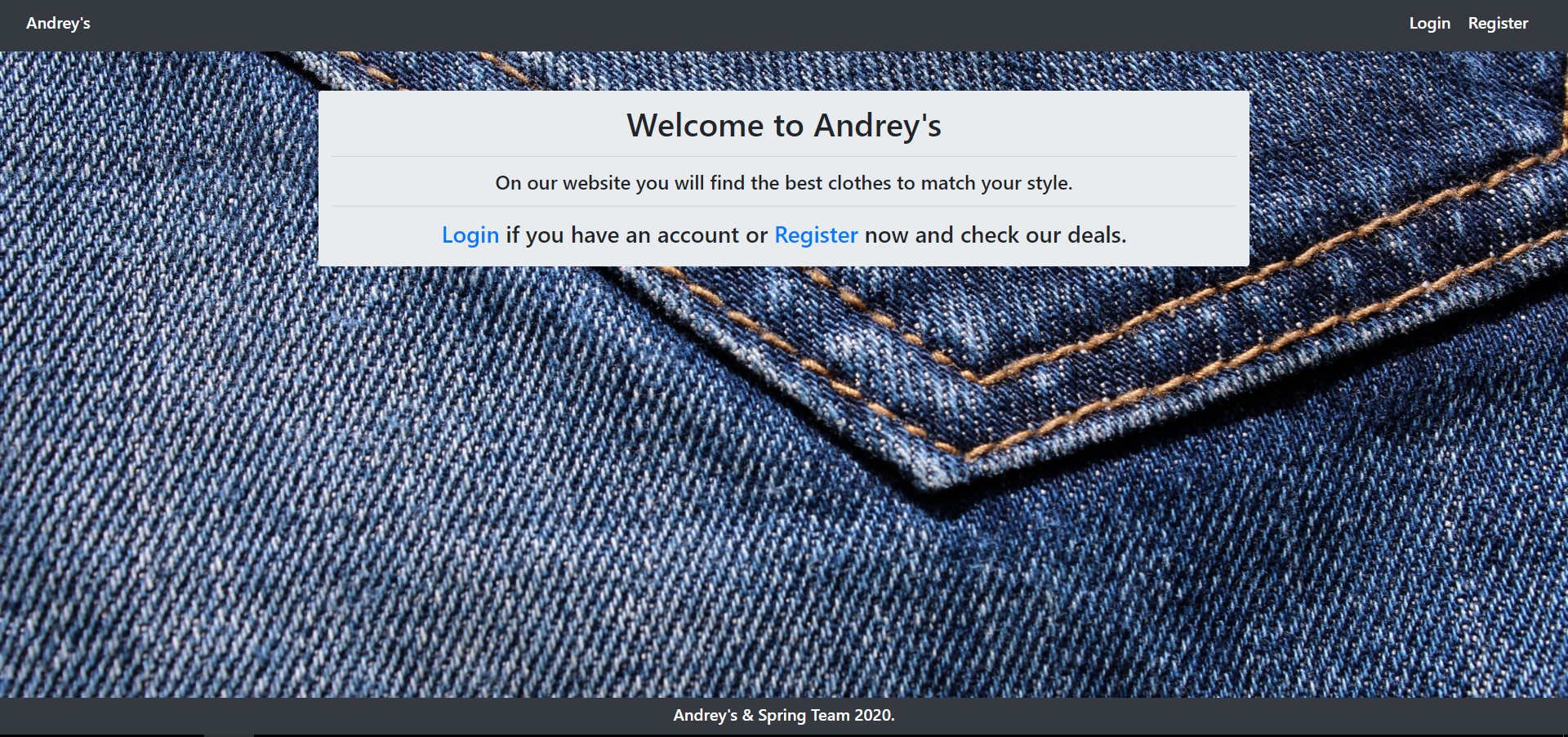
Implement the entities with the **correct datatypes** and implement **repositories** for them.

## Initialize categories

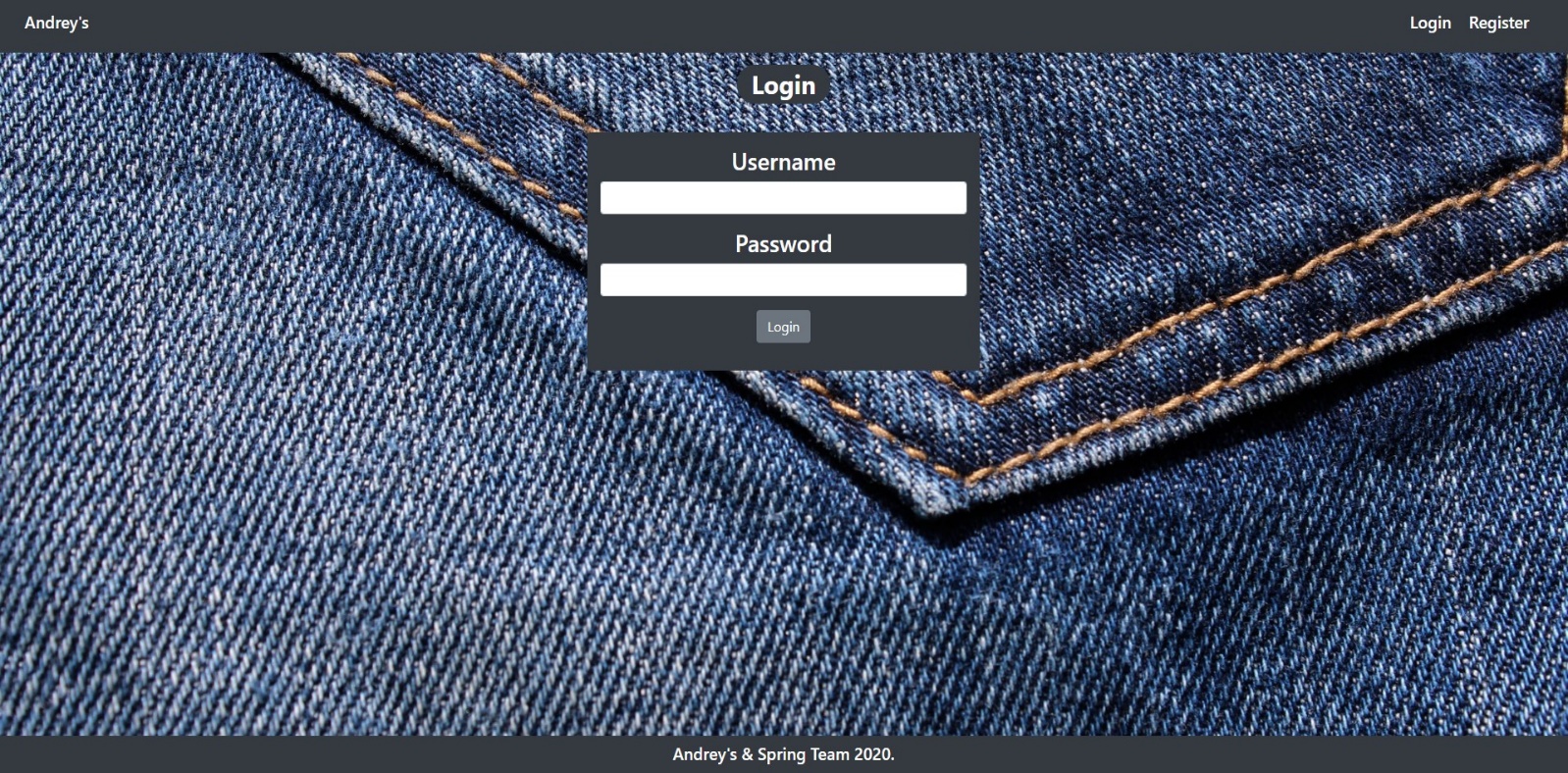
* Implement method that check if the database does not have categories and initialize them
  + You are free to do in different ways.

## Page Requirements

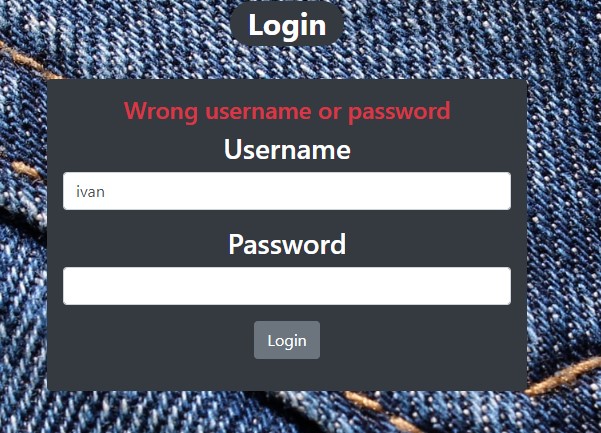
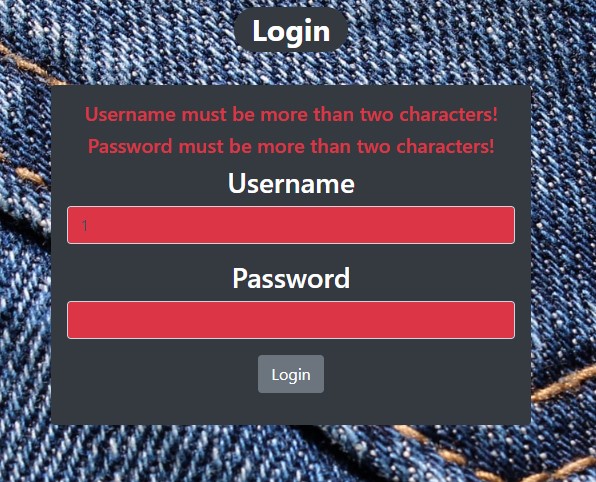
### Index Page (logged out user)



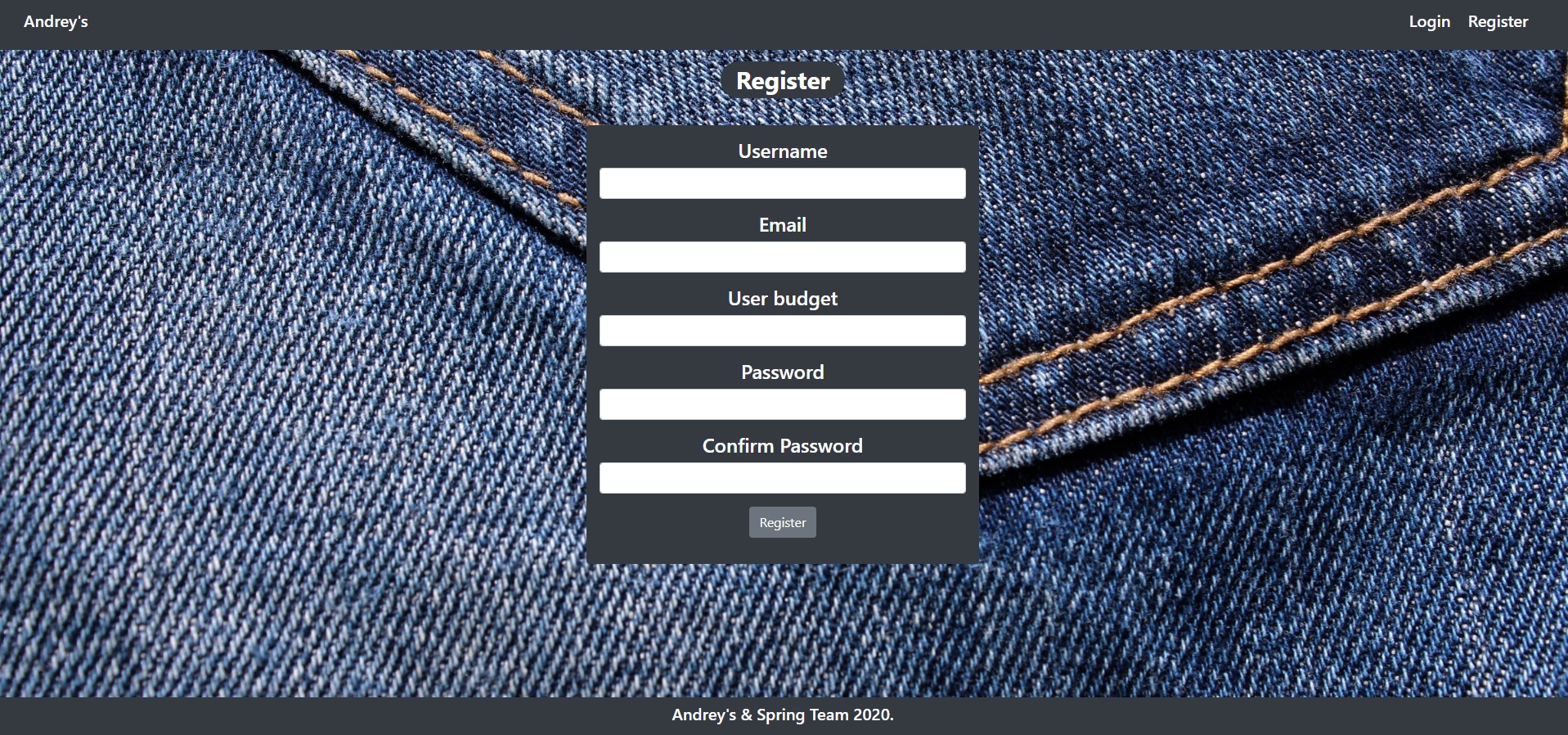
### Login Page (logged out user)



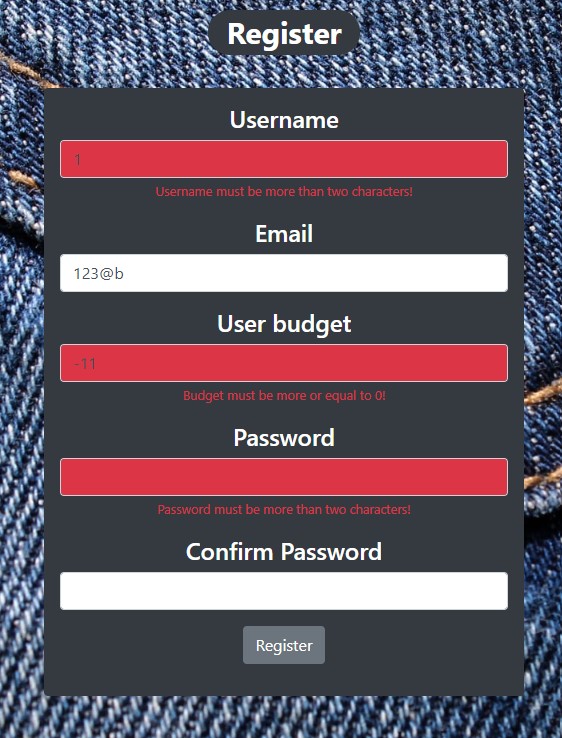
### Login Page (wrong input and missing user)



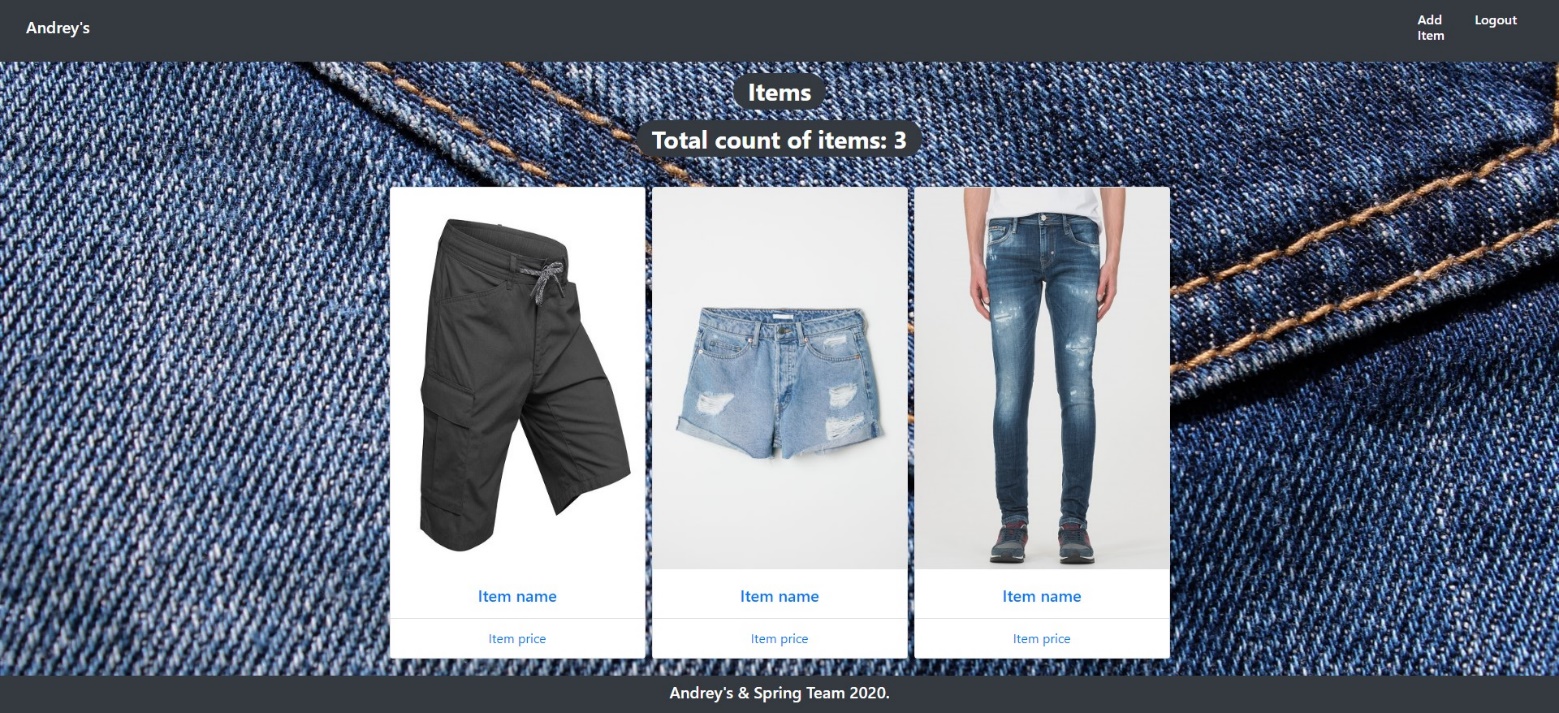
### Register Page (logged out user)



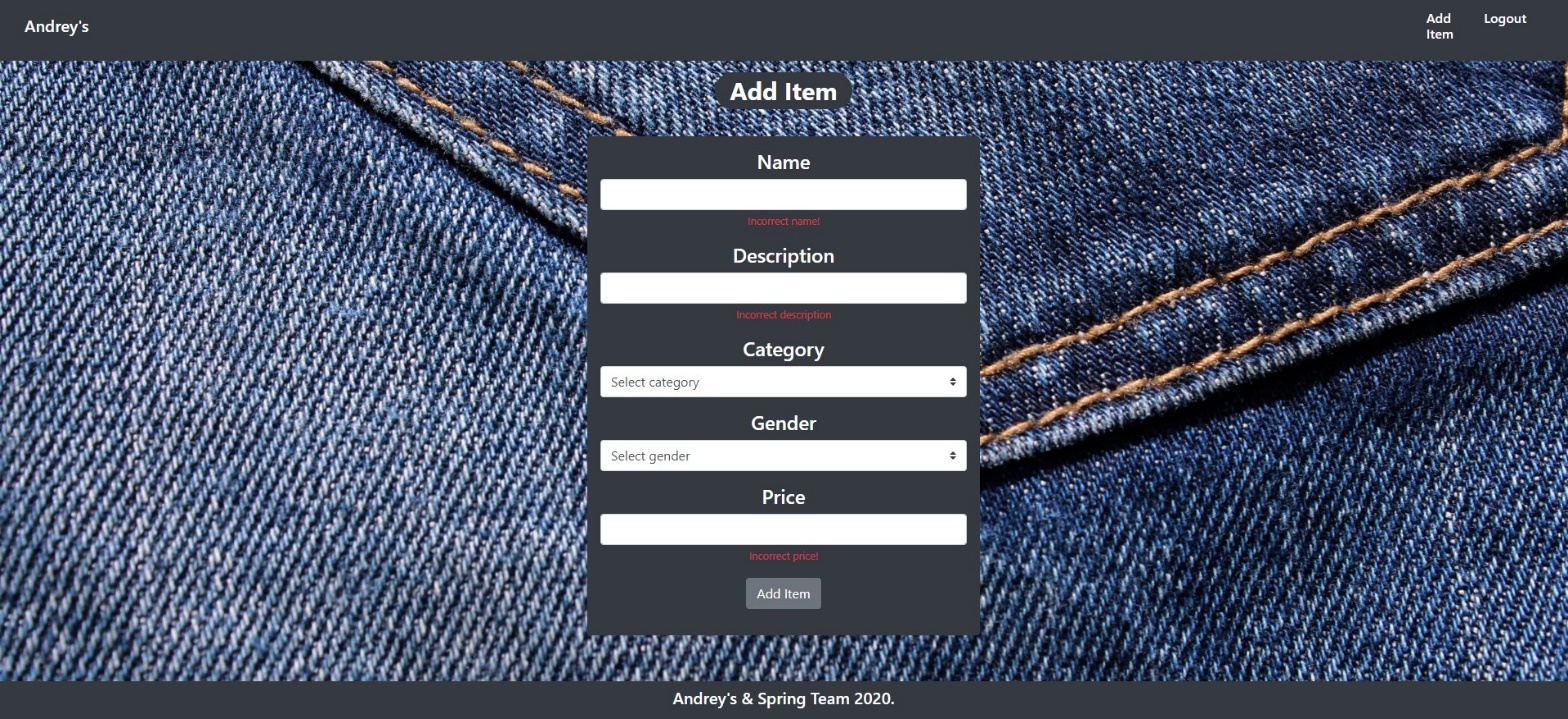
### Register Page (wrong input)



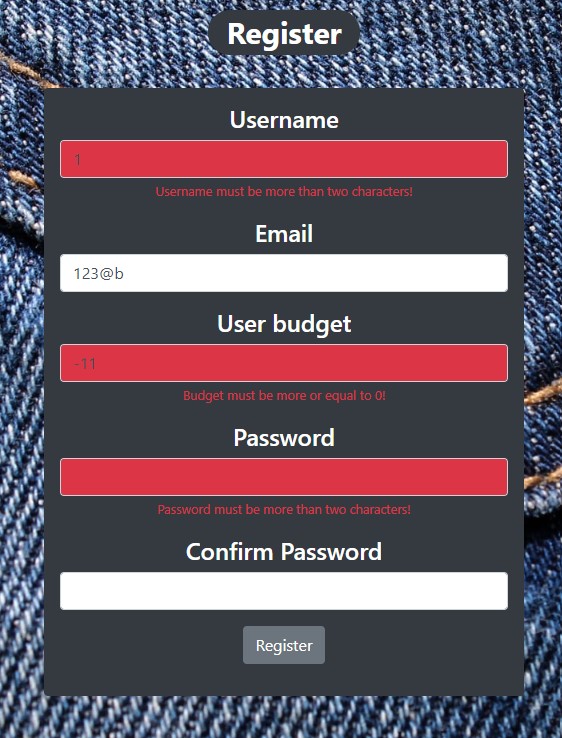
### Home Page (logged in user)



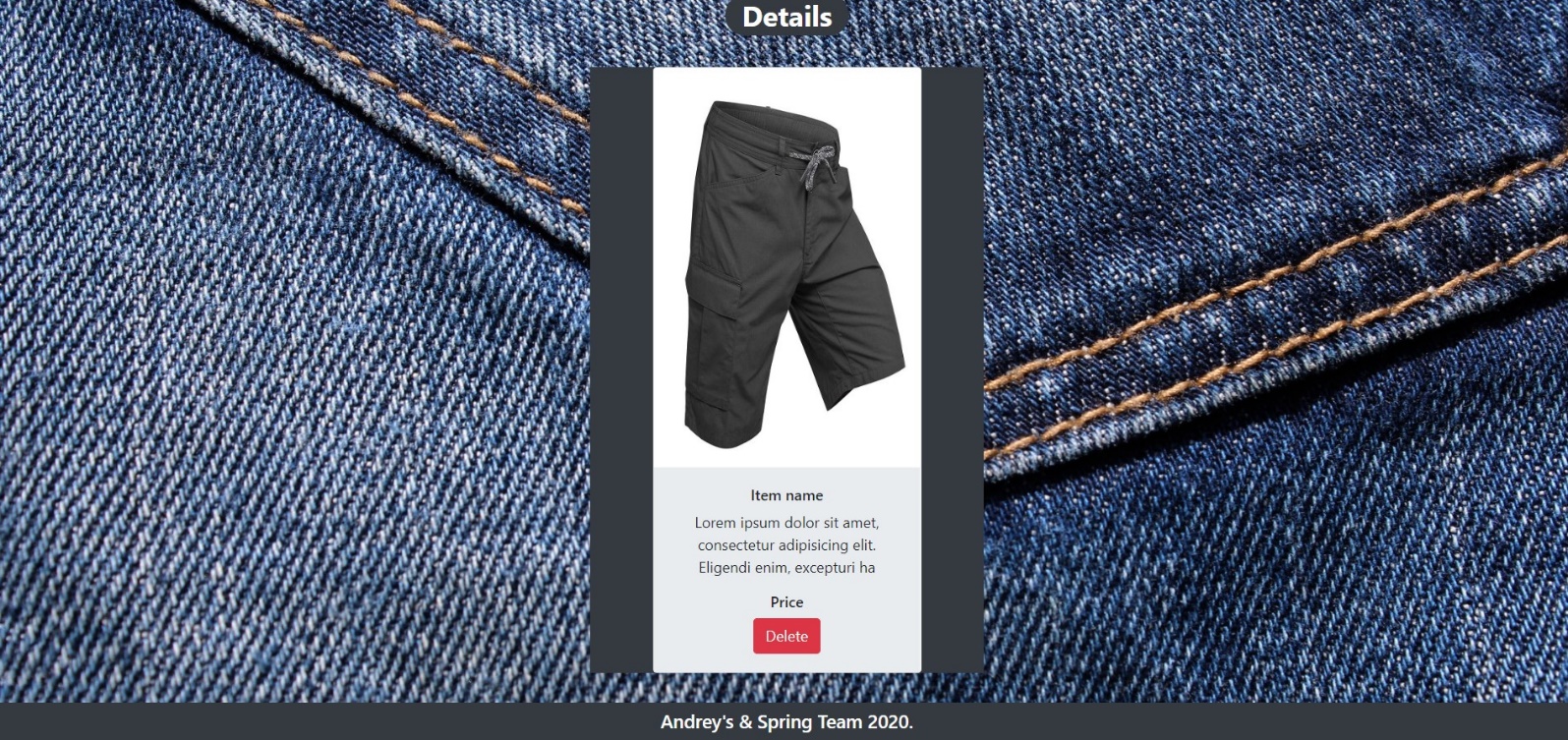
### Add Item (logged in user)



### Add Item (logged in user)



### Item Details Page (logged in user)



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. The given **bootstrap** and **css** are enough.

## Functional Requirements

The **application** should provide the Guest (not logged in) users with the functionality to login, register and **view** the Index page.

The **application** should provide the Users (logged in) with the functionality to logout, **add an Item**, **view** **details** about an Item, **delete** an Item, and **view all** Items (Home page).

**Andrey's button** on the navbar should redirect to appropriate **URL depending** if the user is logged in.

The **application** should provide **functionality** for **adding items** with **categories (Shirt, Denim, Shorts, Jacket) and gender (Male, Female)**

The **image** is combination between **category** and **gender**. When you click on the image on **Home** **Page**, you should be redirected to **Details** **page**. On **Details** **page** you should have **Delete** **button**.

The **application** should **store** its **data** into a MySQL database, using Hibernate native.

## 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 the Item Add page.
* Users (logged in) can access the Item Details page.
* Users (logged in) can access the Logout functionality.

## Scoring

### Database – 10 points.

### Pages – 25 points.

### Functionality – 35 points.

### Security – 5 points.

### Validations – 15 points.

### Code Quality – 10 points.