# Spring Fundamentals Retake Exam

# Coffee Shop Application

Exam for the ["Spring Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3222/spring-fundamentals-january-2021).

The **Coffee Shop Application** is here to help with the normal work of the cafe. The application records all orders that are not yet ready and reminds the employees of them by visualizing a picture of the category, the product's name and its price. It also calculates the approximate time that will be required for the execution of the orders. As an additional functionality, statistics are kept, on which employees can process how many orders at that moment. At this stage, the functionality of the application is small but later it can be expanded upon.

## Database Requirements

The **Database** of the **Coffee Shop** application needs to support **3 entities**:

### User

* Has an **Id – UUID-string or Long**
* Has a Username (unique)
  + The length of the username must be between 5 and 20 characters (both numbers are INCLUSIVE).
* Has a First Name
  + Can be null.
* Has a Last Name
  + The length of the last name must be between 5 and 20 characters (both numbers are INCLUSIVE).
* Has a Password
  + The length of the password must be more than 3 (INCLUSIVE).
* Has an Email
  + Must contain a '@' symbol.
  + The email must be unique.

### Order

* Has an **Id – UUID-string or Long**
* Has a Name
  + The length of the name must be between 3 and 20 characters (both numbers are INCLUSIVE).
* Has a Price
  + The price must be a positive number
* Has an Order time
  + The date and time that cannot be in the future
* Has a Category
  + Has ONLY ONE category
  + This is the relation with categories.
* Has a Description
  + The length of the description must be at least 5 (INCLUSIVE) characters
  + The description is a long text field.
* Has a Employee (user)
  + A user that adds this order to the Coffee Shop application

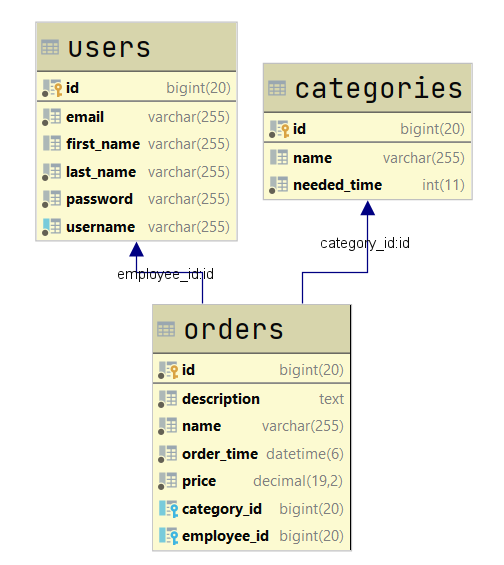
### Category

* Has an **Id** – **UUID-string or Long**
* Has a **Name**
  + An option between (Coffee, Cake, Drink, Other)
* Has a Needed Time (just an integer)
  + The approximate time in minutes needed for the preparation   
    of a product of the specified category.
    - The needed time for a Drink is 1 min.
    - The needed time for Coffee is 2 min.
    - The needed time for an Other is 5 min.
    - The needed time for a Cake is 10 min.

**Nullable/Empty strings are not allowed unless explicitly mentioned.**

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

Here is the ER Diagram:



## Initialize categories

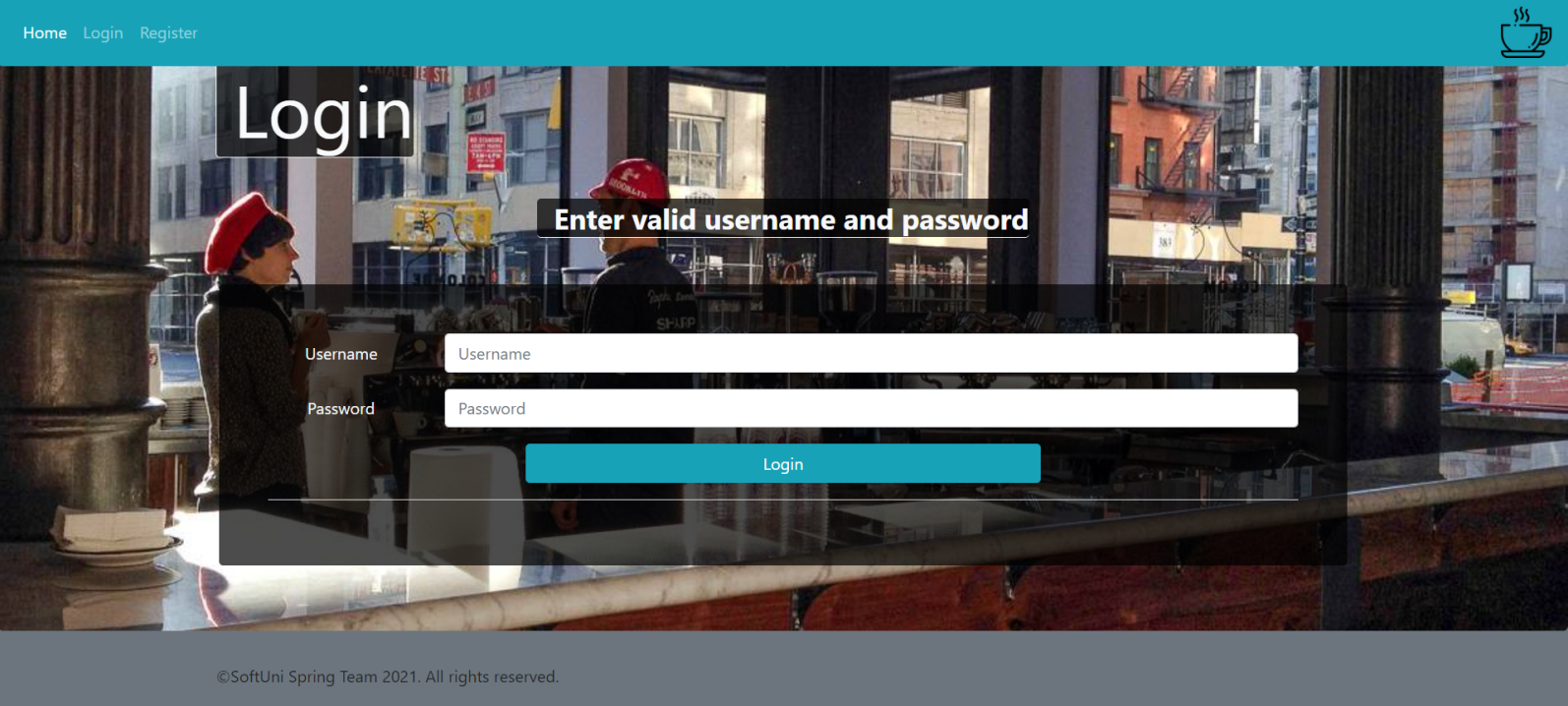
* Implement a method that checks (when app started) if the database does not have any category and   
  initialize them
  + You are free to do this in some different ways.
  + Don't **forget** to add the **needed Time**

## Page Requirements

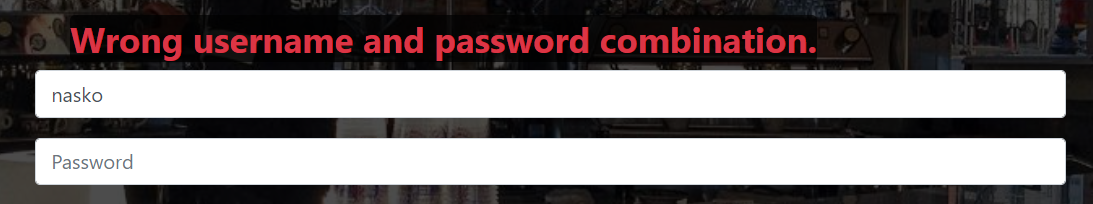
### Index Page (logged out user)



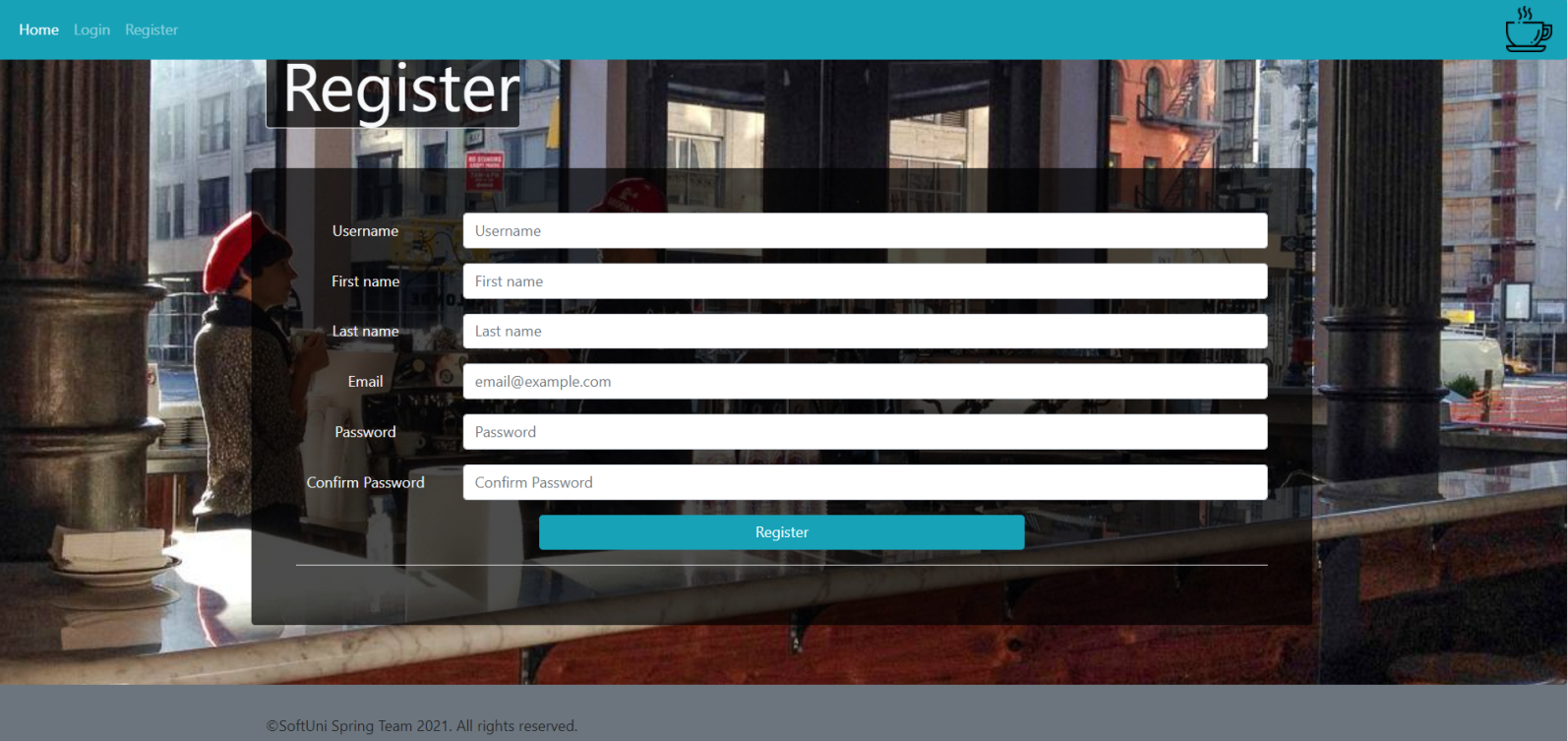
### Login Page (logged out user)



### Login Page validations

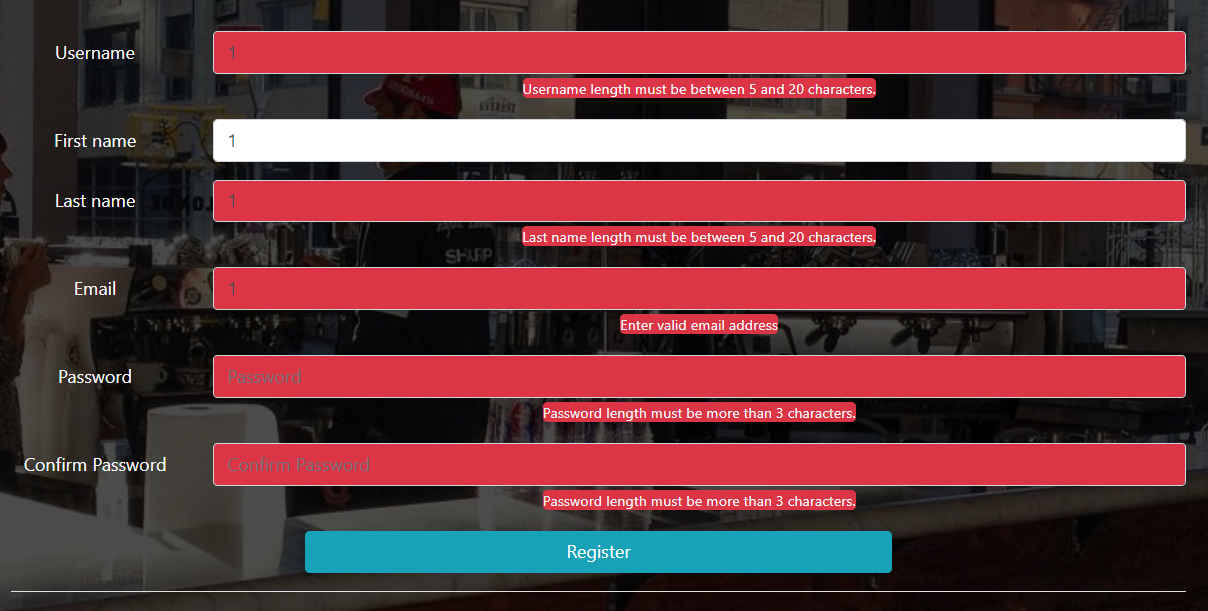


### Register Page (logged out user)



### 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.



### Navigation (Guest user)

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

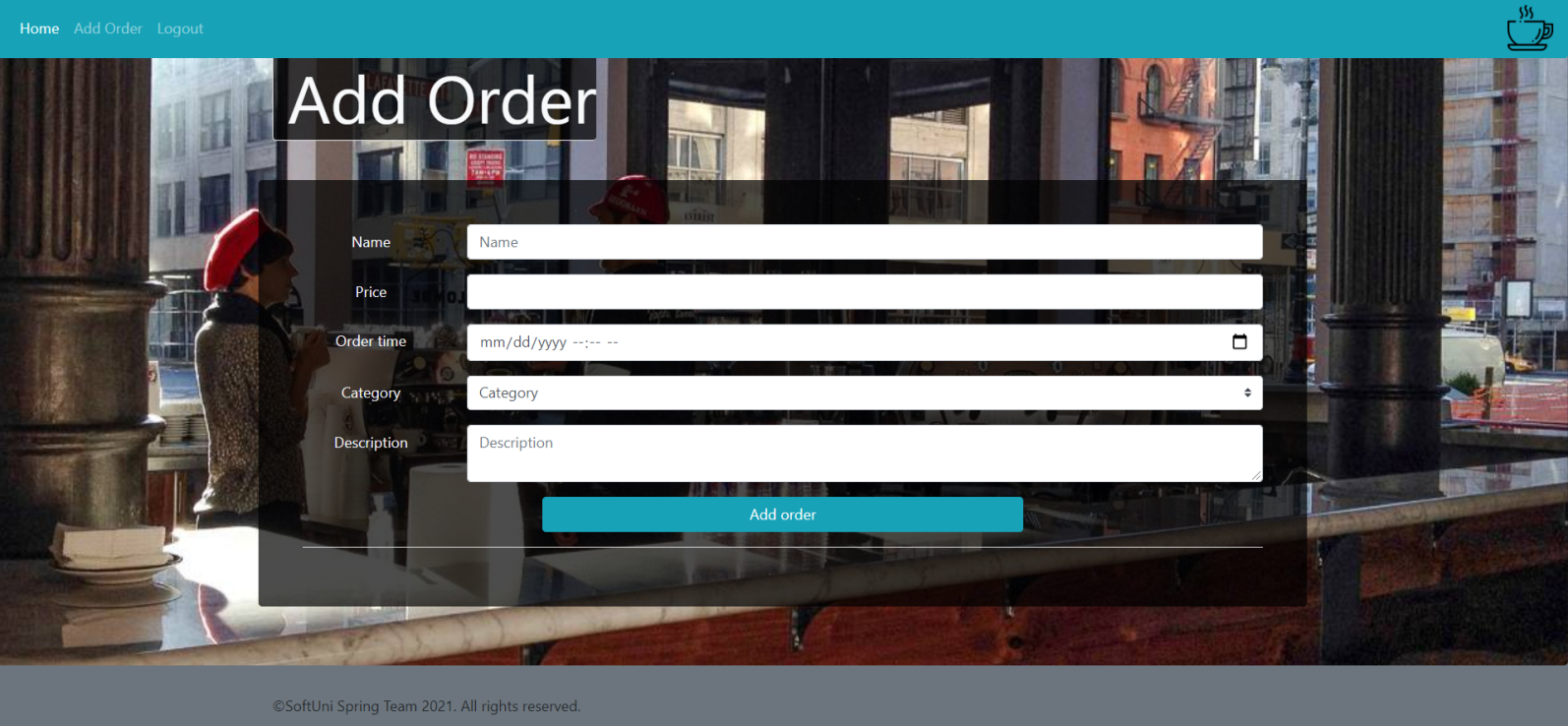


### Navigation (Registered user)

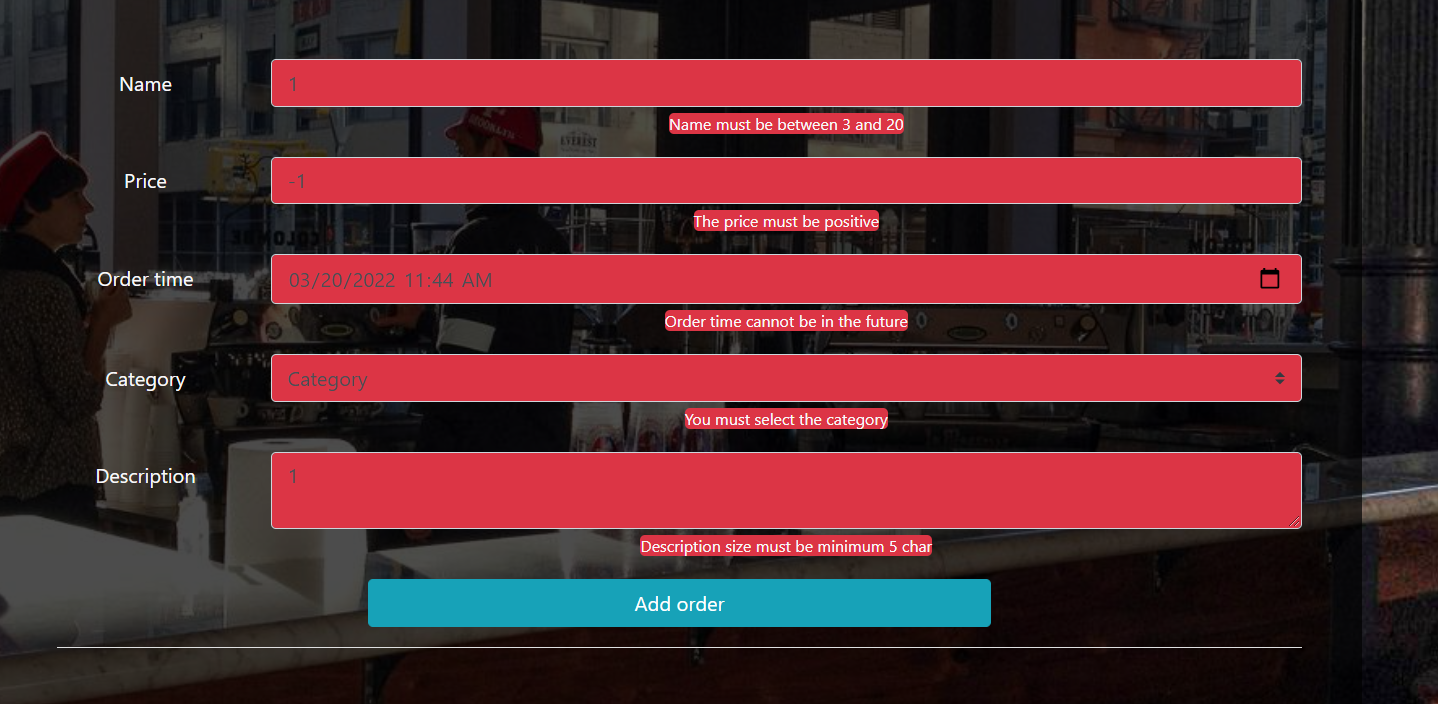
* Note: can access only to **Home**, **Add Order**, **Logout**.



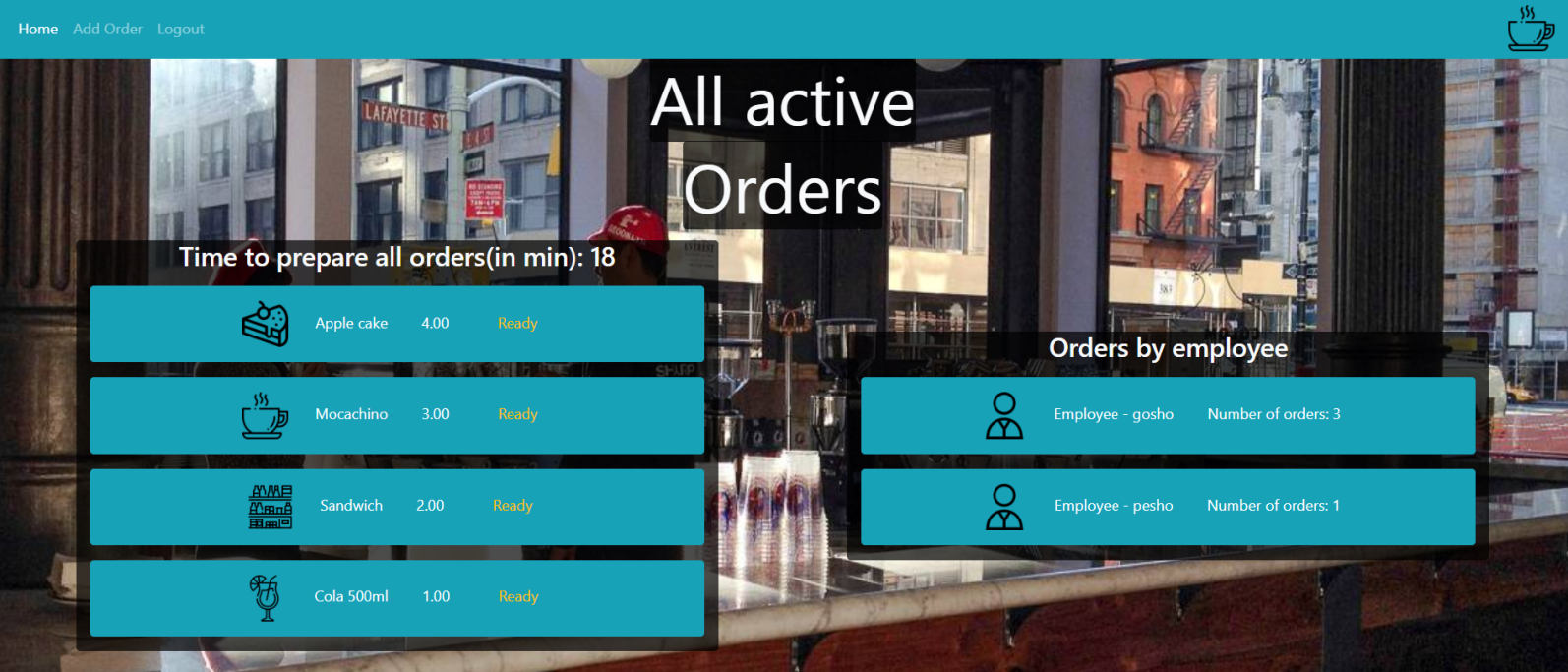
### Add Order



### Add Order validation



### Home Page (with orders)



**NOTE:** The orders are ordered by their price in **descending** order

**NOTE:** The **picture** in front of every order **depends** of order's **category**.

**NOTE:** The employees are ordered by their count of orders in **descending** order.

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 login, register and **view** the Index page.

The **application** should provide Users (logged in) with the functionality to logout, **add an Order**, **view all** Orders in DB (Home page) and **Ready a single one** from orders.

In **Coffee Shop Application**, the navbar should redirect to appropriate **URL depending** on that if the user is logged in.

The **application** should provide **functionality** for **adding orders** with **category and employee.** Also can **Ready** them and remove from the database.

Ready button remove the selected order from the database and again redirect to the home page

The first header on the home page shows the total needed time to prepare all orders in minutes(integer number) in the database at this moment.

Order by Employee tab show every registered employee and his unready orders at this moment.

The **application** should **store** its **data** into 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 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 Order 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.