# JS Apps Exam – SoftBay App

You are assigned to implement a **Web application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Firebase). Initially, all views and forms are shown by the HTML. Your application must display the current views depends on the routing URL. You may render the views/forms/components with **Handlebars**.

You are **allowed** to use libraries like **jQuery**, **Handlebars** and **Sammy**. Frameworks and libraries like React, Angluar, Vue are **not permitted**.

Important  
Don't change the elements' **class names** and **ids**. You are **allowed** to add **data attributes** to any elements. You may modify href **attributes** of links and add action**/**method **attributes** to **forms**, to allow the use of a routing library.

## Create a REST Service

Use any cloud-based databaseand create application to keep your data in the cloud.

Create a collection called **softbay.**

## HTML and CSS

You have been given the web design of the application as **HTML** + **CSS** files.

* Initially all views and forms are shown by the HTML. Your application may **hide**/**show elements** by CSS (display: none) or **delete**/**reattach** from and to the DOM all unneeded elements, or just display the views it needs to display.
* You may render the views/forms/components with **JavaScript** or **Handlebars**.
* You are **allowed** to add **attributes** to any HTML elements.

## SoftBay - Client-Side Web Application

**Design** and **implement** a client-side front-end app (SPA) for managing **offers**. Implement the functionality described below.

### Navigation Bar (Header) (5)

Navigation links should correctly change the current page (view).

Clicking on the links in the **NavBar** should display the view behind the link (views are represented as sections in the HTML code).

Your application must display the views it needs to display.

The **Logged-in** **user** navbar should contain the following elements: **[**SoftBay], **[**Dashboard**]**, **[Create Offer]** and **[**Logout**]**.



The guest users navbar should contain the following elements: **[SoftBay]** and **[Login]**



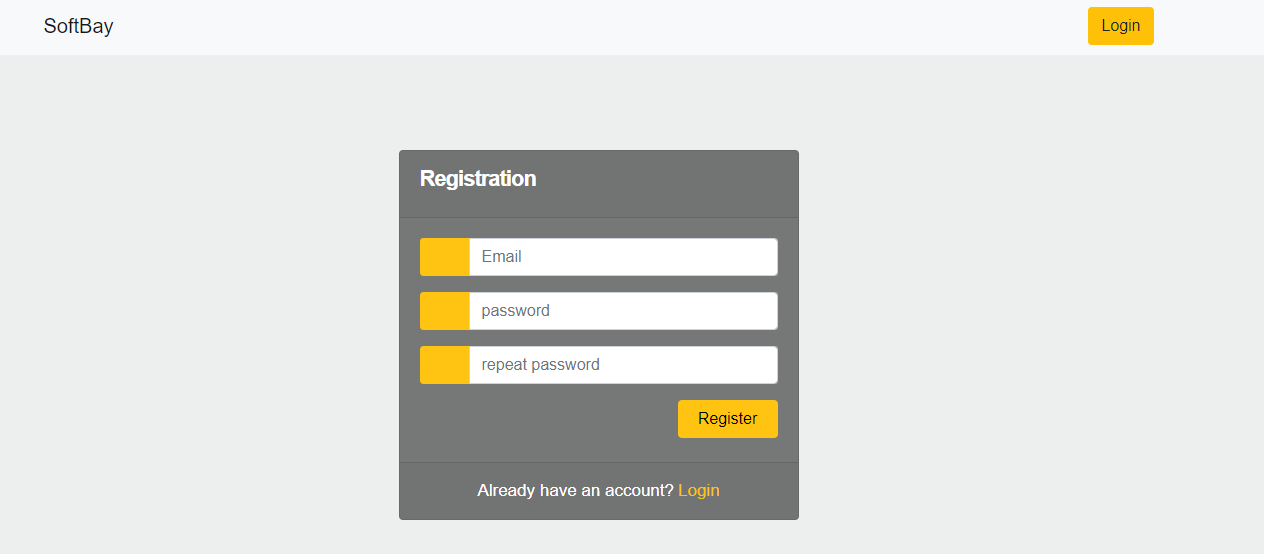
### Footer



### Register Page (5)

By given **email** and **password,** the app should register a new user in the system.

* The following validations should be made:
  + The **email** and **password** must be non-empty string
  + The **re-password** should be **equal to the password**
* After a **successful registration**, **home** **page** should be displayed **with the right navbar**.
* Keep the user session data in the browser's **session or local storage**.

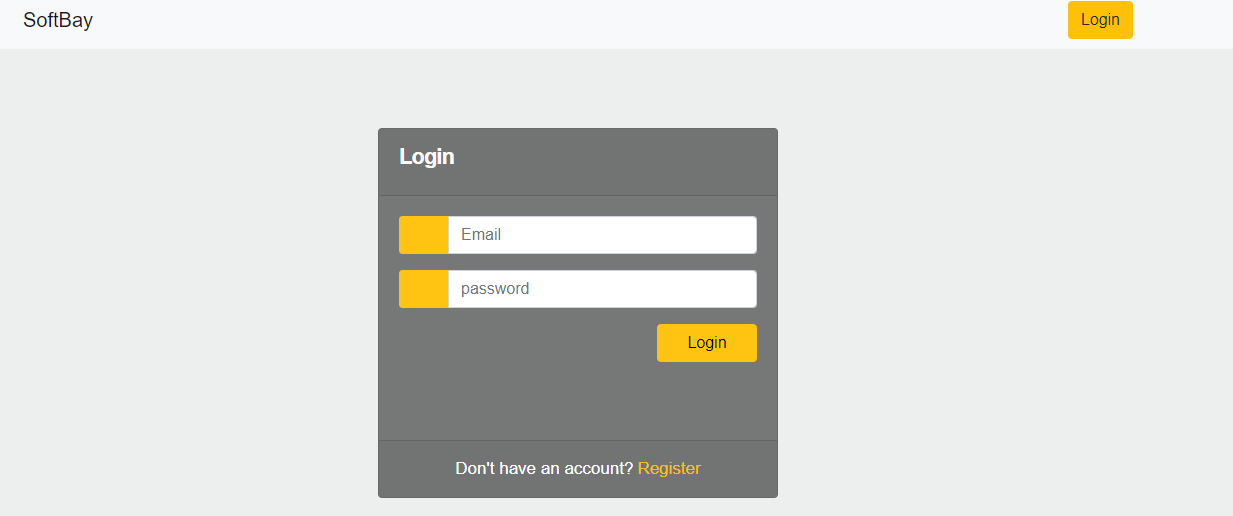


[**Login**] **link** should refer to the **login form (view)**

### Login Page (5)

By given **email** and **password,** the app should login an existing user.

* After a **successful login**, a **home** **page** should be displayed with the **right** **navbar**.
* Keep the user session data in the browser's **session or local storage**.



[**Register**] **link** should refer to the **register form (view)**

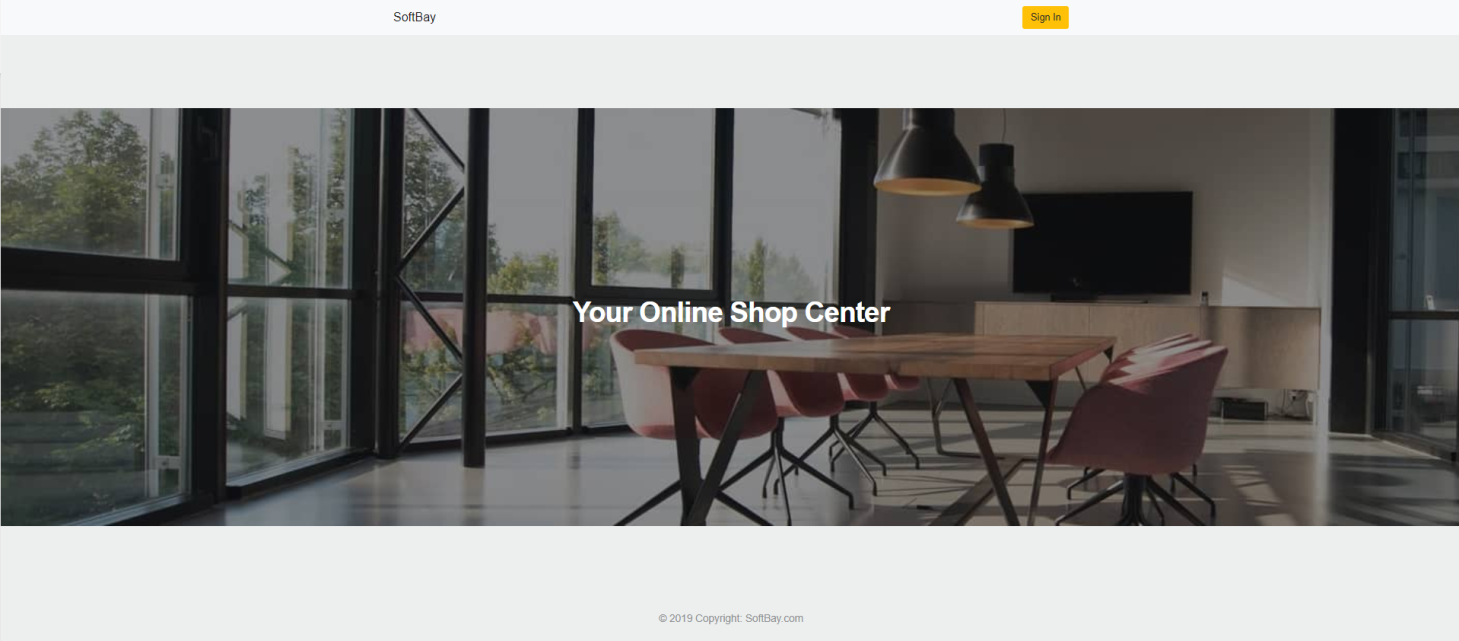
### Logout (5)

Successfully logged in users should be able to **logout** from the app.

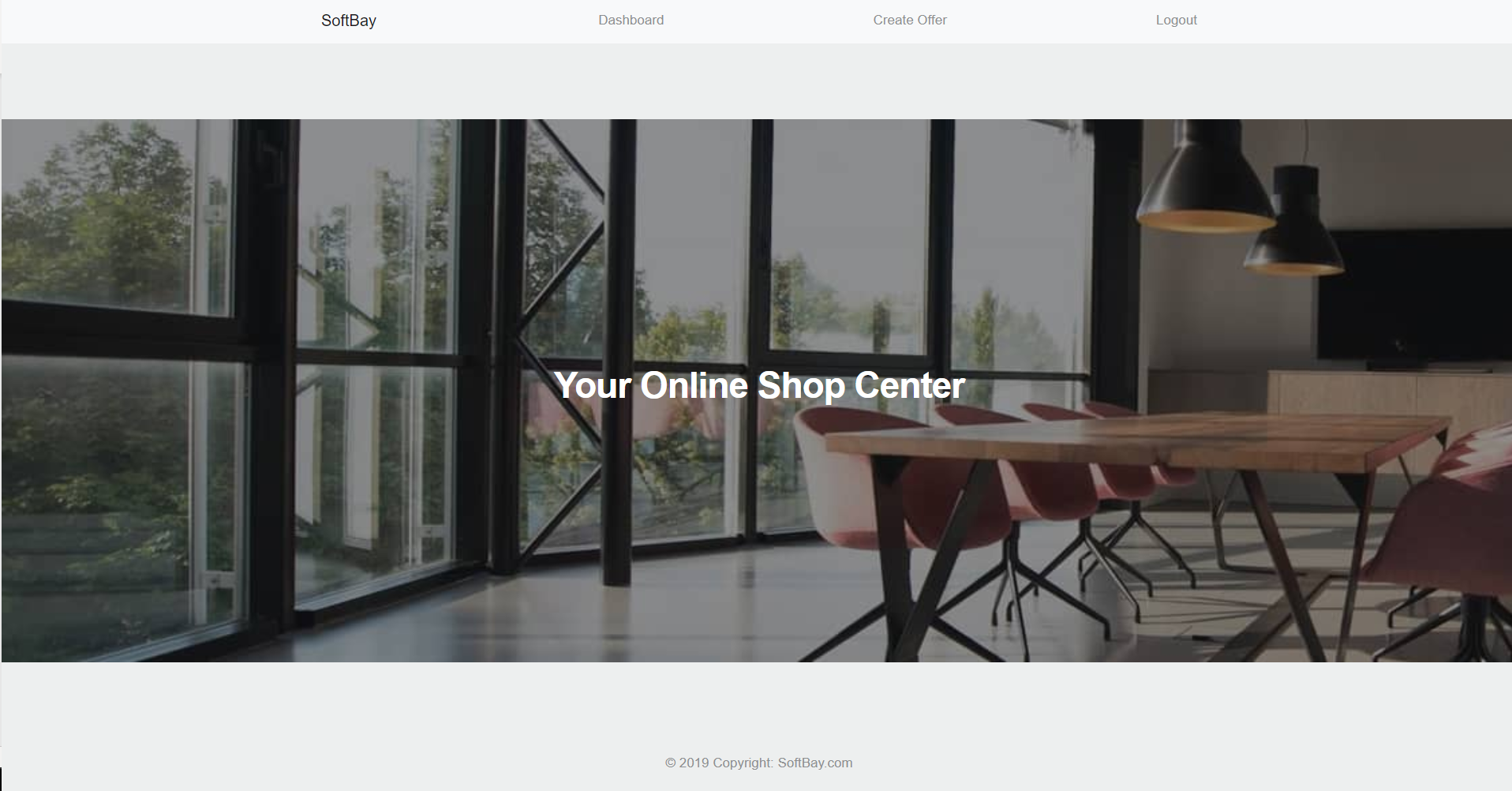
* After a **successful** logout the **home page for guest users** should be shown
* All local information in the browser (**user session data**) about the current user **should be deleted**

### Home Page (5)

The initial page (view) should display the **navigation bar for guest users** + **Home Page** + Footer. [**SoftBay**] **button** should refer to the **home page (view)**



If user is logged-in the page (view) should display the **navigation bar logged-in users** + **Home Page** + Footer.



### Create Offer (15)

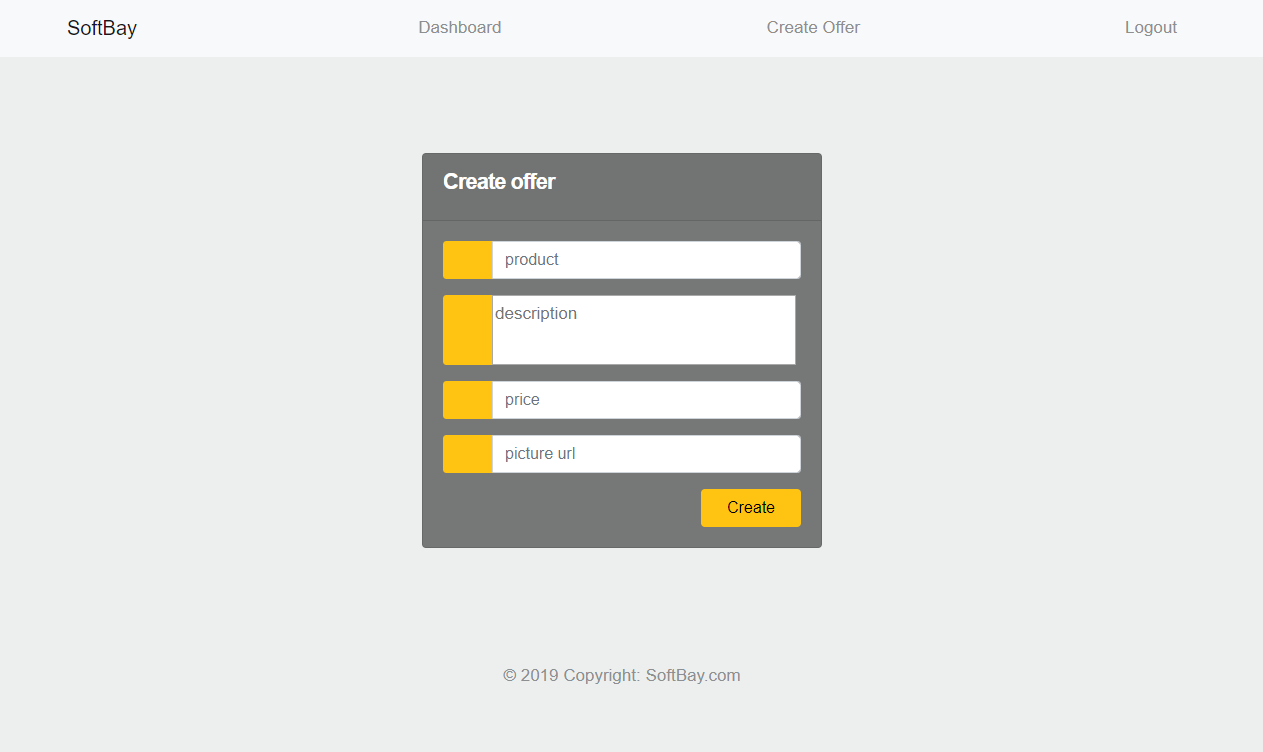
Logged-in users should be able to **Create offers**.

Clicking the [Create Offer] **link** in the **NavBar** should **display** the Create Offer page.

* The form should contain the following validations:
  + The input fields for **product**, **description** and **price** should be **non-empty strings**
  + The input field for **imageUrl**, must be **valid** **url** *(starts with "https://")*

After a **successful** offer creation the Dashboardshould be shown.

The newly created offer should be stored in the database collection "offers".

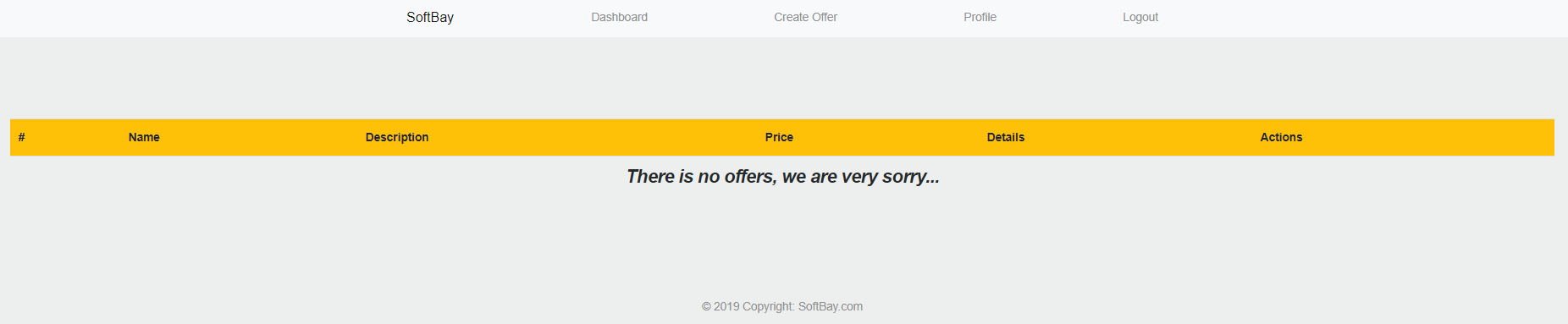


### Dashboard (25)

By clicking over [**Dashboard**] button, the currently logged-in user should be able to see all created offers.

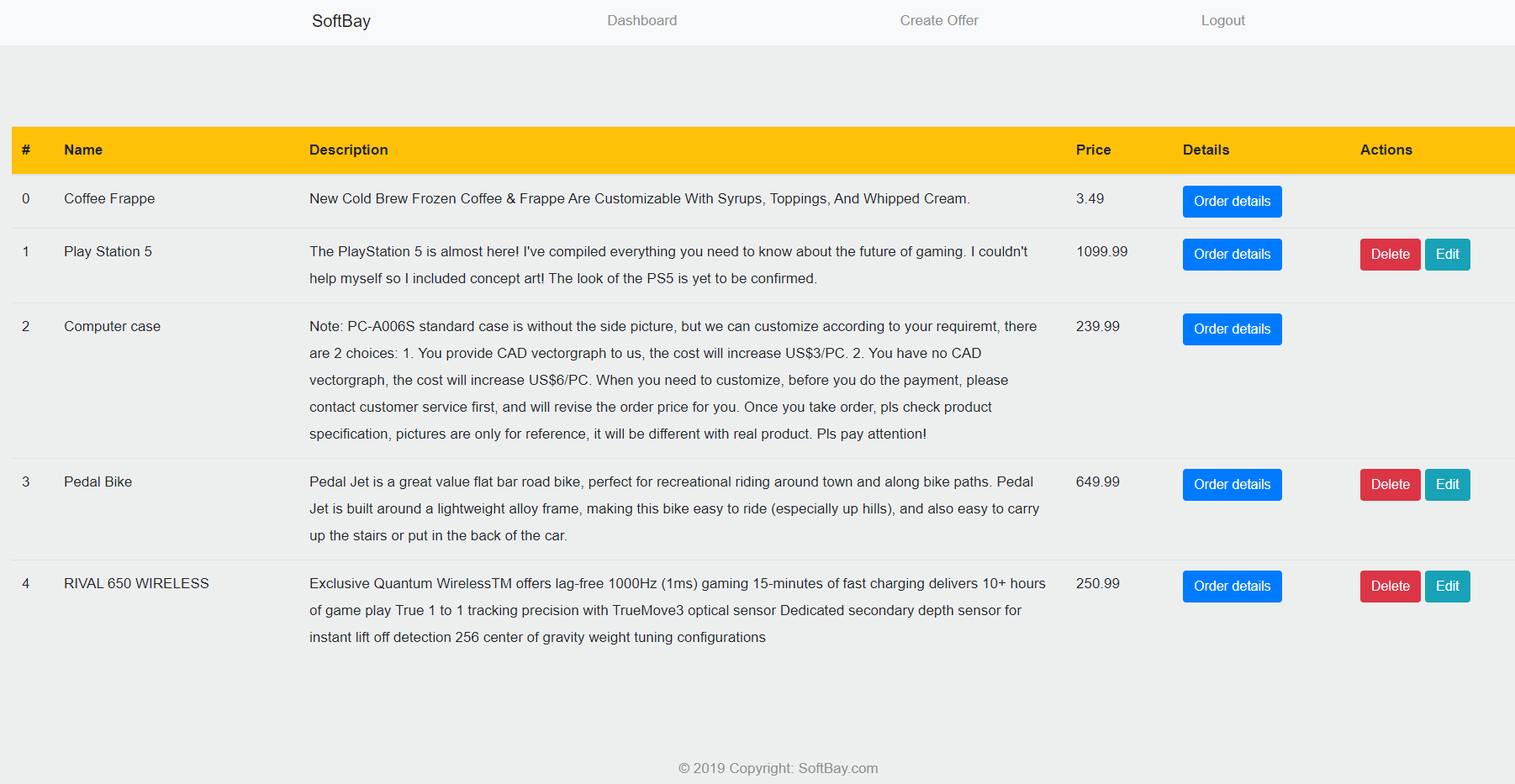
The initial page (view) should display the **navigation bar for logged-in users** + **Dashboard** + Footer.

If there are **NO** such offers, the following view should be displayed.



If there is at least **one registered offer,** the offer must be shown like the example below in format:

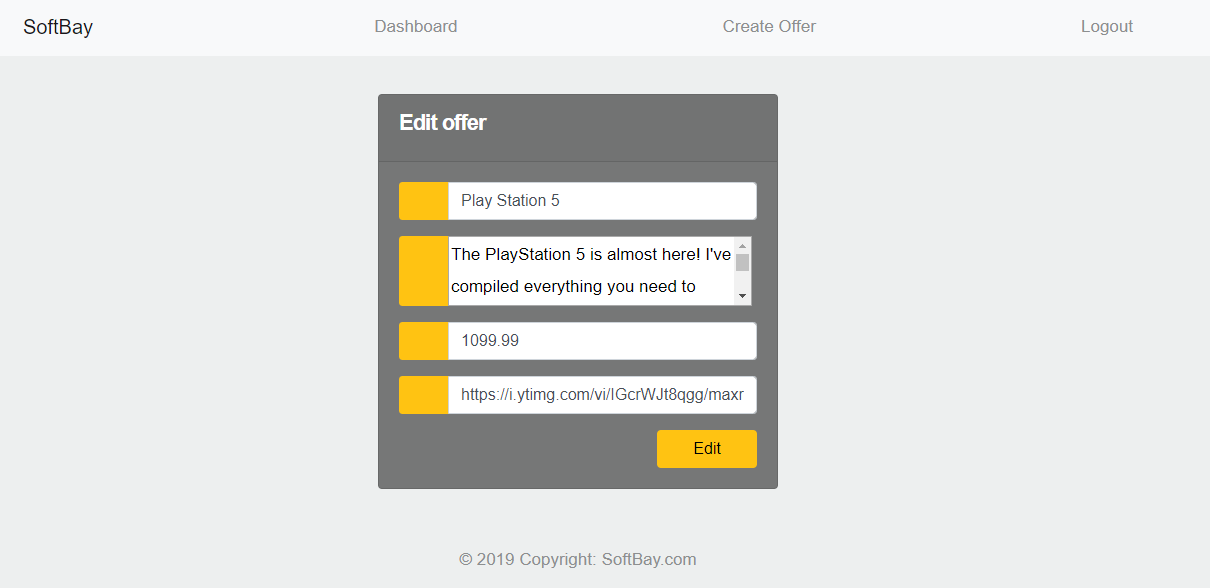
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Name | Description | Price | Details | Actions |
| Current offer **index** | Current product **name** | Current product **description** | Current product **price** | [**Order details**] button which refer to the details for the current offer/product | [**Delete**] [**Edit**]  Buttons, only if the currently logged-in user is the creator of that offer. Where both buttons refer to the delete and edit view (pages) for the current offer/product |



### Edit Offer (15)

Logged-in users should be able to **edit** their **own** offers.

Clicking the [Edit] **link** of a **particular offer**, Edit Offer Page should be displayed with **already filled** input fields with the current offer information. By cliking the [**Edit**] button, a put request should be send to the **offers** **collections** and the current offer **should be changed**.



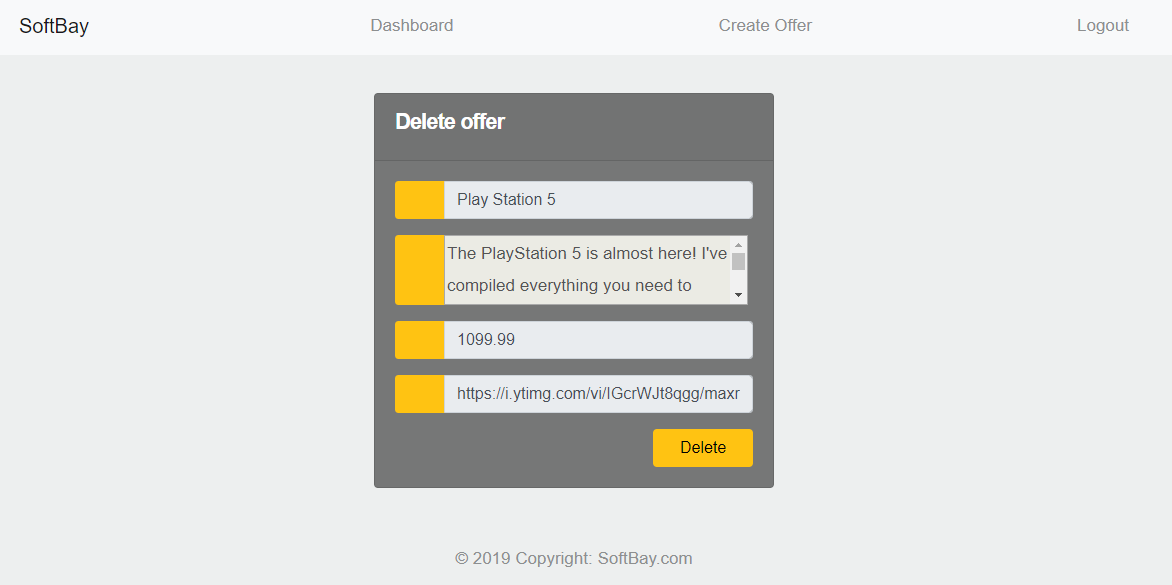
### Delete Offer (5)

Logged-in users should be able to **delete their** offers.

Clicking the [Delete] **link** of an **particular offer** the Offer Delete page should be displayed with already filled information about that offer, but all input fields are disabled.

If the [**Delete**] button in the **Offer Delete Page** is clicked, the current offer must be **deleted** from the DOM and from the **database** collection.

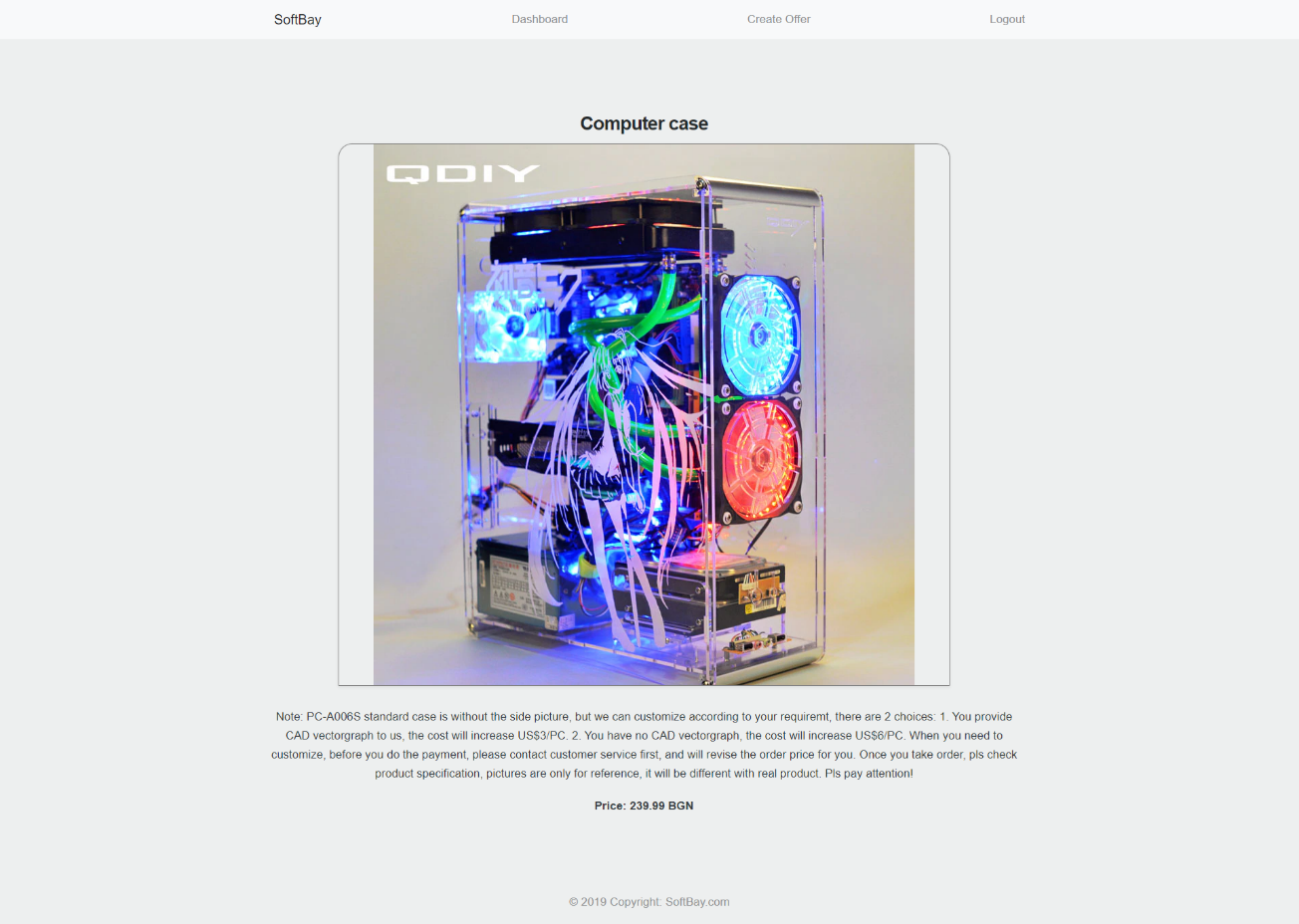
After **successful** **offer delete** a **Dashboard page** must be shown.



### Offer Details (15)

Logged-in users should be able to **view details** aboutoffers.

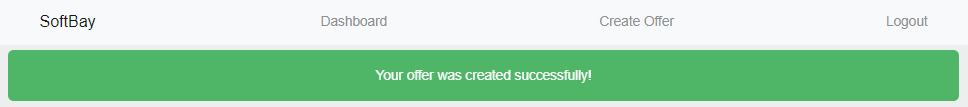
Clicking the [Order details] **link** in of a **particular offer** should **display** the Offer Details page, where the **product** **name**, **picture**, **description** and the **price** are shown in the format below on the picture.



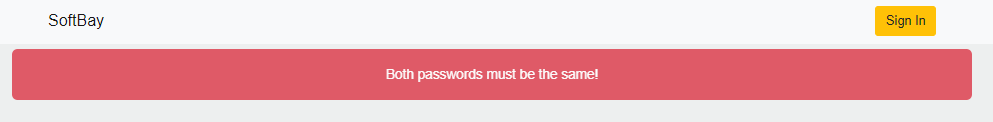
### BONUSES (15)

#### Notifications (5)

* In case of a **successful** action (create offer, login, registration, edit, delete), a **notification message** **(green)** should be shown, which disappears after **3 seconds**.



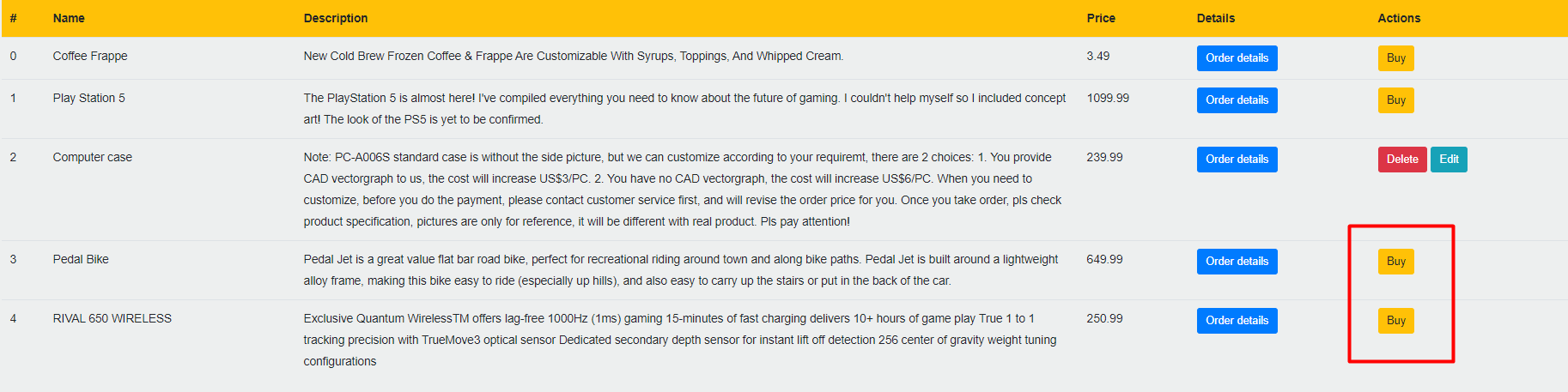
* In case of **error** (wrong passwords, wrong ajax call, authorization, etc...), an **error notification message (red)** should be shown, which disappears after **3 seconds** or **on user click**.

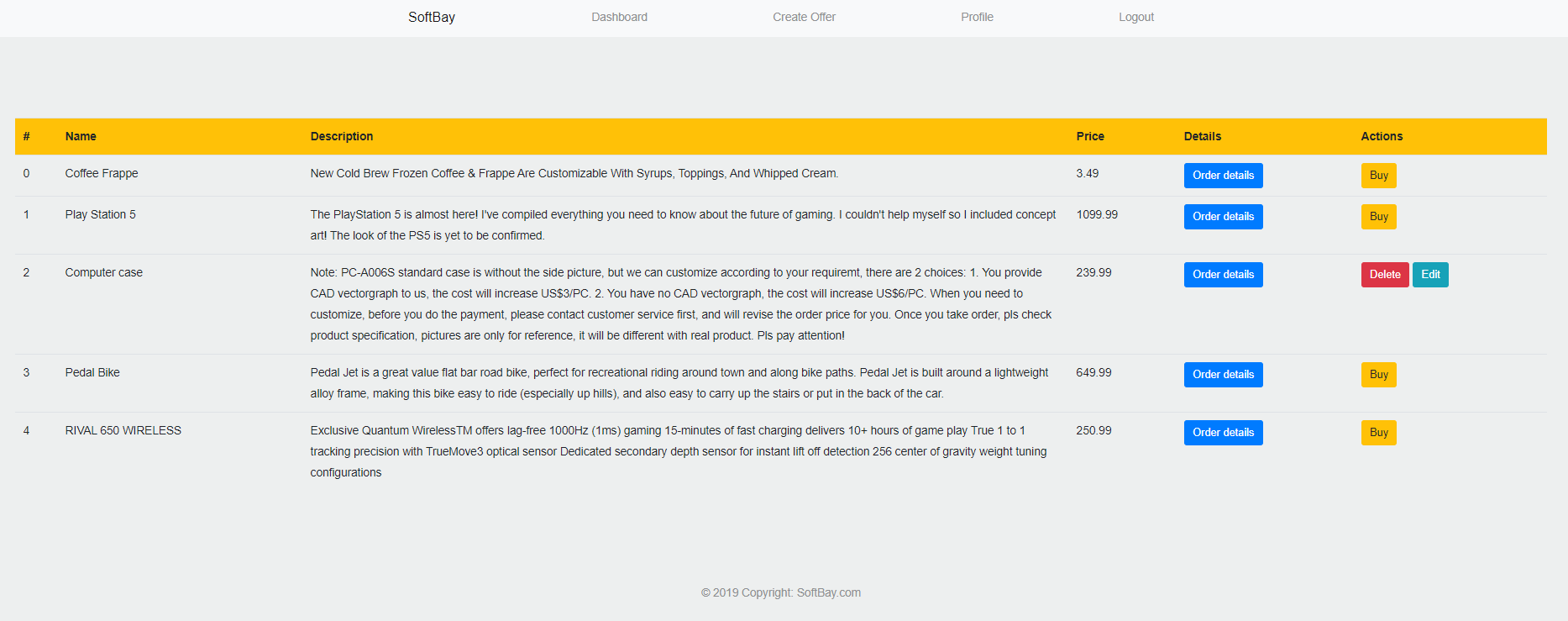


**Note:** The notification should stay visible also when redirected to another page. By default all notifications are styled with "***display: none"***

#### Buy an item + Profile page (10)

In these cases when some of the offers is **not made** from the currently logged-in user, in the section "**Actions**" must be added a [**Buy**] button, which by clicking it the current product must be successfully bought by the user (there are no quantities on the products, so you don’t have to remove them from the list or something else, just store the number of purchases in the database for the current user)



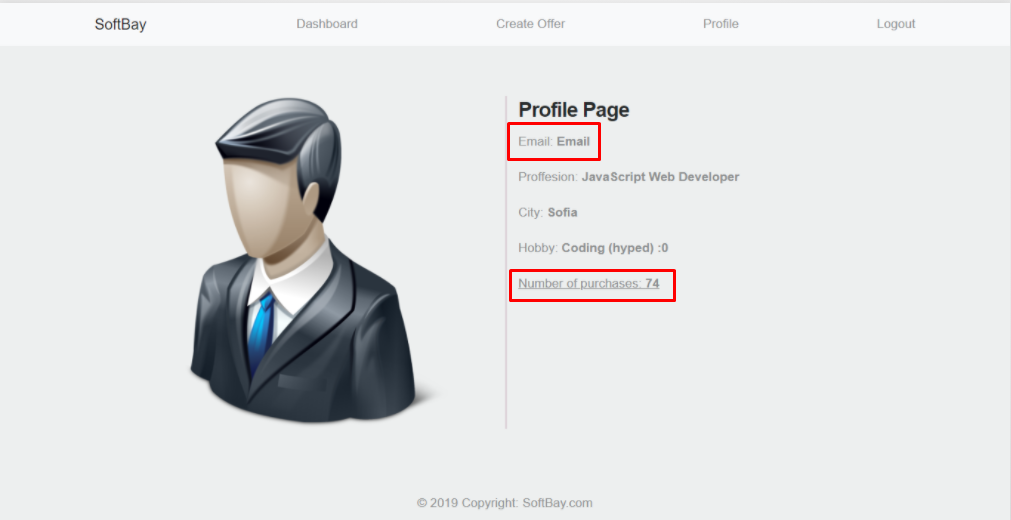


#### Profile Page

In the NavBar (header) must be added a new link [**Profile**] which refer to the currently logged user.



It should be display **2** types of information: **email** for the current user and **number** **of** **purchases** that current user is made. **Any other info is static,** and you don’t have to think about it.



### Note

**Each view should have his own unique and properly named action (url), to be able to get the full points from each section.**

## Submitting Your Solution

Exclude the node\_modules folder and ZIP your project. Upload the archive to Judge system.



**Again: zip the project without the** node\_modules **folder!!!**