**JS Applications Exam – Shoe Shelf SPA**

You are assigned to implement a **Single** **Web Application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Firebase, Backendless or other). Using libraries like **Handlebars** and **Sammy** is allowed but is not obligatory. The app keeps **users** and **shoes**. **Guests** should be able to **register** and **login**. Logged-in users should be able to view **all shoes**, **add** **shoes**, **buy shoes**, see **details** about a **shoes** and **logout**. Logged-in users should also be able to **edit** or **delete** the **shoes they have added**.

## Create a REST Service

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

Create a collection called **shoes.** Each **shoes** has a **name**, **price**, **imageUrl**, **description, brand,** **creator** and **people bought** **it.**

## Create Shoes Application

**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.

**Important**: Don’t change the elements’ **class names** and **ids**. Don’t rename form fields/link/ids. You may modify href **attributes** of links and add action**/**method **attributes** to **forms**, to allow the use of a routing library.

**Client-Side Web Application**

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

**Navigation Bar (5 Pts)**

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

* Clicking on the links in the **NavBar** should display the view behind the navigation link.
* 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.
* **The Logged-in** user navbar should contain the following elements:[**Create new offer**] a **link** to the Create page, the **Shoe Shelf** **logo** - link to the listed shoes, the user caption ("Welcome, {email}") and [Logout] link.

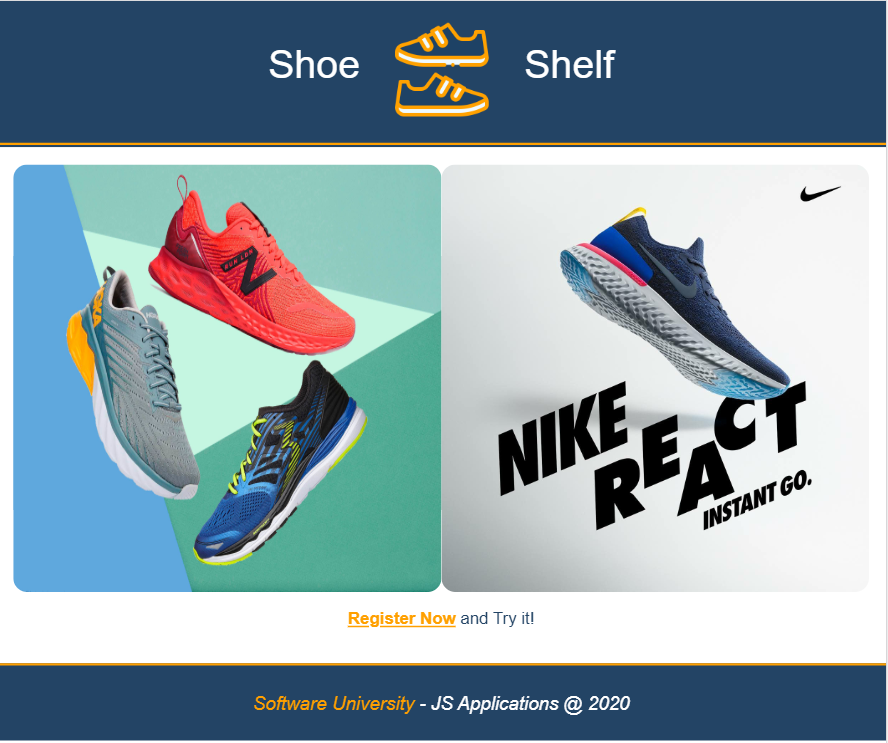


* The guest users navbar should contain the following elements: : **Shoe Shelf** with the **logo** inbetween.



**Home Page (Guest) (5 Pts)**

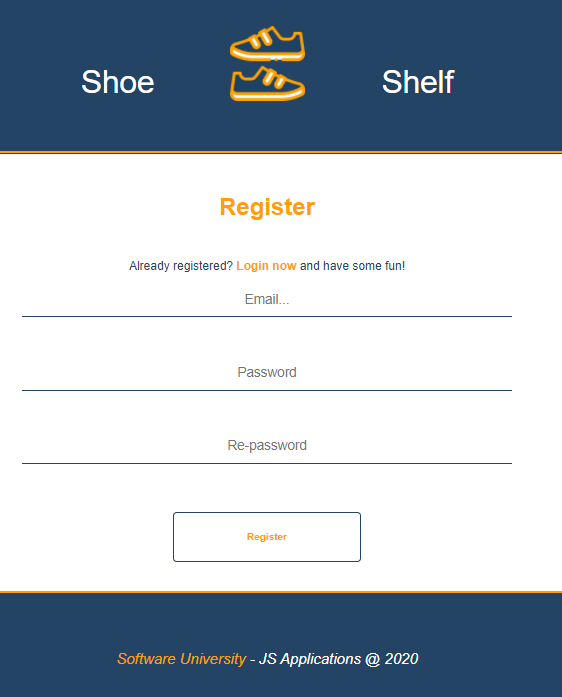
The initial page (view) should display the **guest** **navigation bar** + **Guest Home Page** + Footer.



**Register User (5 Pts)**

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

* The following validations should be made:
  + The **email** input must be **filled**
  + The **password** should be **at least 6 characters** long
  + The **repeat password** should be **equal to the password**
* Keep the user data in the browser's **session or local storage**.
* After a **successful registration** redirect to **Home page.**
* In case of **error**, nothing happens, the user should be able to fill in the form again.



**Login User (5 Pts)**

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

* Keep the user data in the browser's **session or locale storage**.
* After a **successful login** redirect to **Home page.**
* In case of **error**, nothing happens, the user should be able to fill in the login form again.



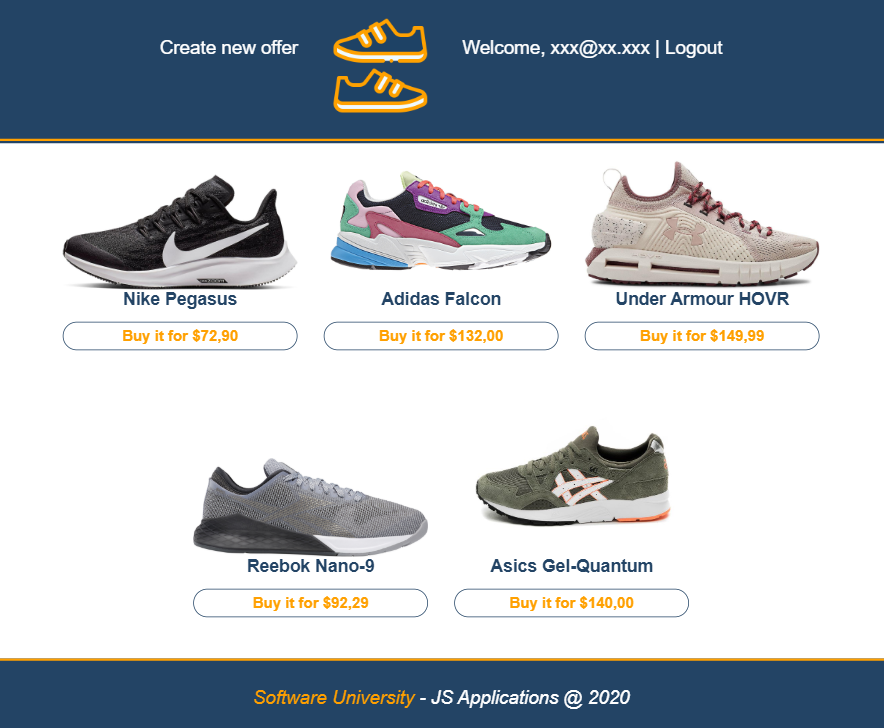
**Logout (5 Pts)**

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

* The **"logout" REST service** at the back-end **must** be called at logout
* All local information in the browser (**user session data**) about the current user should be deleted
* After a **successful logout** redirect to **Login page.**

**Home Page(Logged in User) (30 Pts)**

Successfully logged-in users should be welcomed by the Homepage. They should be able to see all added shoes:



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

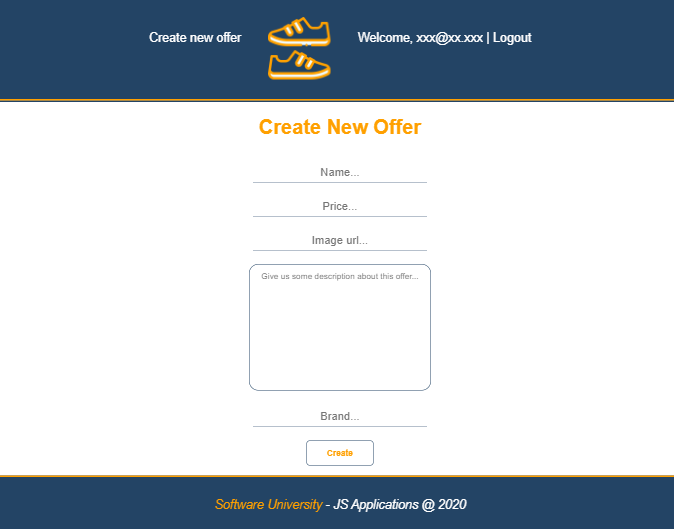


**Create Offer(10 Pts)**

Logged-in users should be able to **add shoes**.

Clicking the [Create new offer] **button** in thenavbar should **display** the Create page.

* The form should contain the following validations:
  + All **input** fieldsshouldn’t be **empty**.
  + **By** **default**, every newly created offer must have additional information:
    - **Creator:** string representing the current user;
    - **People bought it:** keeping data of users bought the shoes;
  + After a **successful** creating Home pageshould be shown.
* The newly added offer should be stored in the database collection "shoes".



**Details (15 Pts)**

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

Clicking on a **particular offer** should **display** the Details page.

* If the currently logged-in user is the creator, the [Delete] and [Edit] **buttons** should be set to **visible**, otherwise there should be only 1 button [Buy].

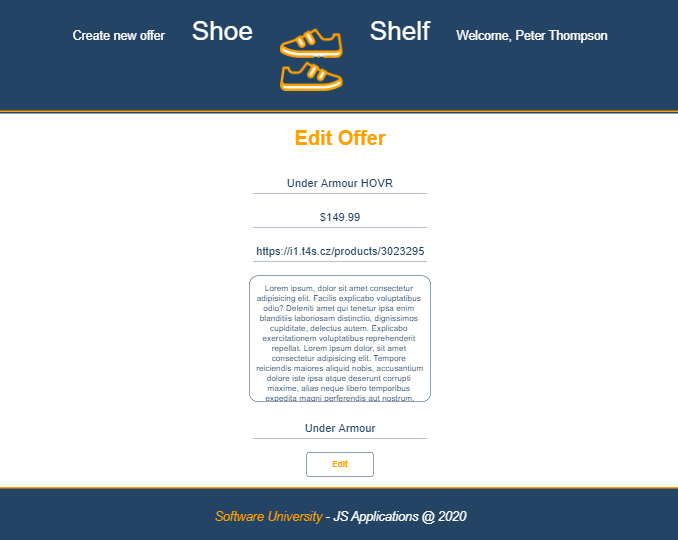




**Edit Offer (10 Pts)**

Logged-in users should be able to **edit** offers, added by them.

Clicking the [Edit] **button** of a **particular offer** on the Details page should **display** the Edit page inserting the additional information of the shoes in the input feelds:



* After a **successful edit** user should be redirected to the current shoes Details page.

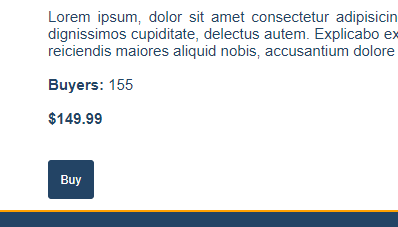
**Buy Shoes (10 Pts)**

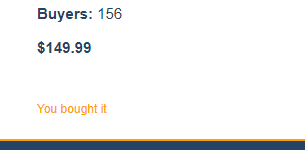
Logged-in users should be able to **buy** shoes, added by **other** **user**.

**NOTE**: A user should **NOT** be able to buy offer, created by **himself**.

Clicking the [Buy] **button** (on the Details page) should **add the current user email to the property People bougth it.** After **successfully buying**:

* Display the updated Details page
* By click [Buy] **button** changes to [You bought it] **span so** users can‘t **buy an item** multiple times**.**





**Delete Offer (5 Pts)**

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

Clicking the [Delete] **button** of an **offer** (on the Details page) should **delete** the **offer**.

* After **successful** **delete** - Home pageshould be **shown**

**(BONUS) Order: (5 Pts)**

**Home page** for logged user shoud display offers in **descending** **orderd by count** of **people bought** the item.

## 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!!!**