## Express.js Exam – Shoe Shelf

**Exam Rules:**

* You have 4 hours **from 14:00 to 18:00**
* When you are ready, **delete** the ***node\_modules***folder, make sure all dependencies are listed in the ***package.json*** file and submit your archived project at: [SoftUni Judge](https://judge.softuni.bg/Contests/2535/JS-Back-End-Retake-Exam-19-August-2020)
* You are provided with **HTML & CSS** resources, which you **CAN** **modify**
* Use **Express.js** as a back-end framework
* Use **MongoDB** as a database with **mongoose**
* You can use whatever **view engine** you buy (Handlebars, EJS, Pug etc.…)
* Keep in mind that the points listed below, are the maximum point you can receive for a given functionality!

## Application Overview

Get familiar with the provided **HTML & CSS** and create an application for **shoes.**

## Functional Requirements

The Functionality Requirements describe the functionality that the **Application** must support.

**Guest (Not Logged In)**

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

**Users (Logged In)**

The **application** should provide **Users** (logged in) with the functionality to **view all the pair of shoes which are listed**, **pair of shoes details page** and **they should be able to buy a pair of shoes**. Shoes can be **public** or **not.** Users can access **only** **public pair of shoes.**

**Database Models (10 Pts)**

The **Database** of the Shoes application needs to support **2 entities**

### User

* Email - string (required), unique
* Full Name - sting
* Password - string (required)
* **Offers Bought - a collection of** Offers

### Shoes offer

* **Name - string (required), unique**
* **Price - number (required), min value 0,**
* **Image Url - string (required),**
* **brand - string**
* **Created at – Date or String, required**
* **buyers - a collection of Users**

Implement the entities with the **correct datatypes**.

**Client-Side Web Application**

**Design** and **implement** a client-side front-end app (SPA) for managing **offers**. 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.
* **The Logged-in** user navbar should contain the following elements:[**Create new offer**] a **link** to the Create page, the **Shouse Shelf** **logo** a link to the listed shoes, logothe user caption ("Welcome, {email}") and [Logout] link.

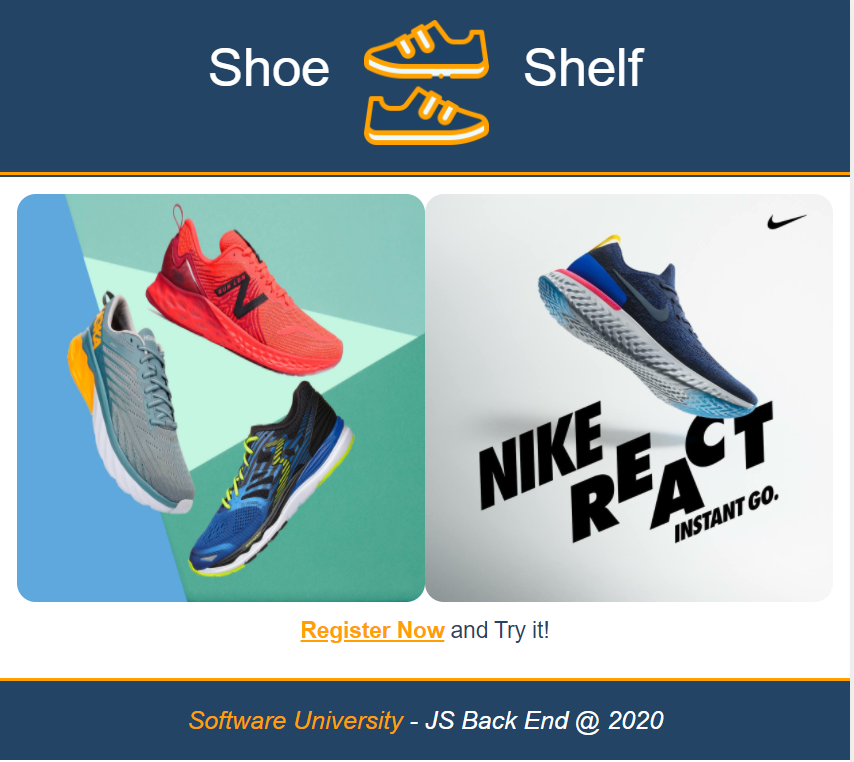


* The guest users navbar should contain the following elements: : **Shoe Shelf** with the **logo** inbetween.



**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 **at least 3 characters** long
  + The **password** should be **at least 3 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 **username** and **password,** the app should login an existing user.

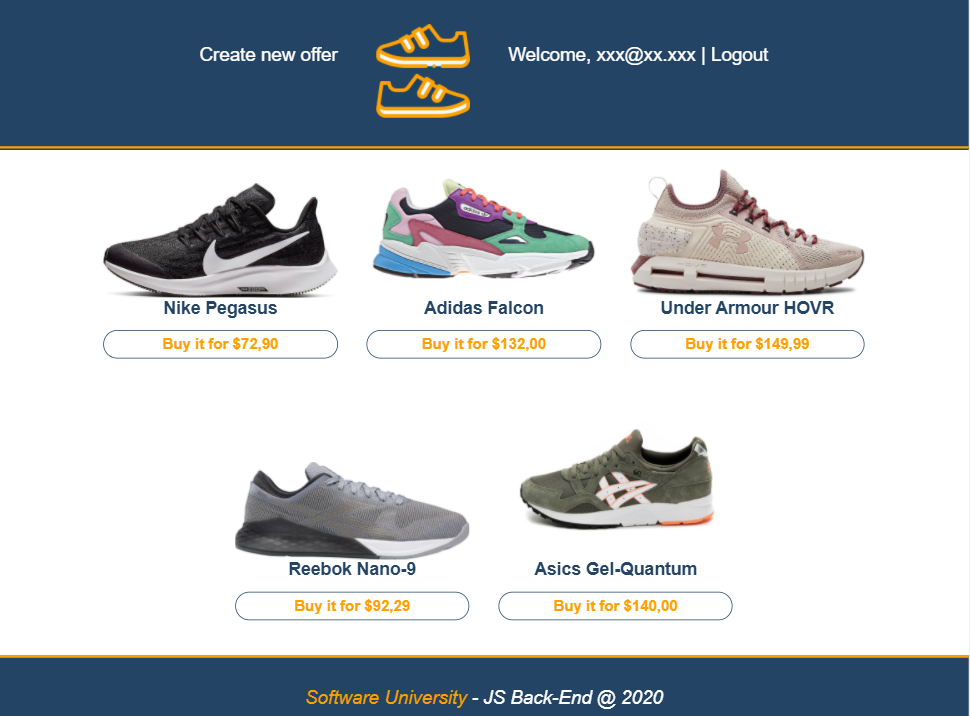
* After a **successful login** redirect to **Home page.**
* In case of **error**, nothing happens, the user should be able to fill in the login form again.



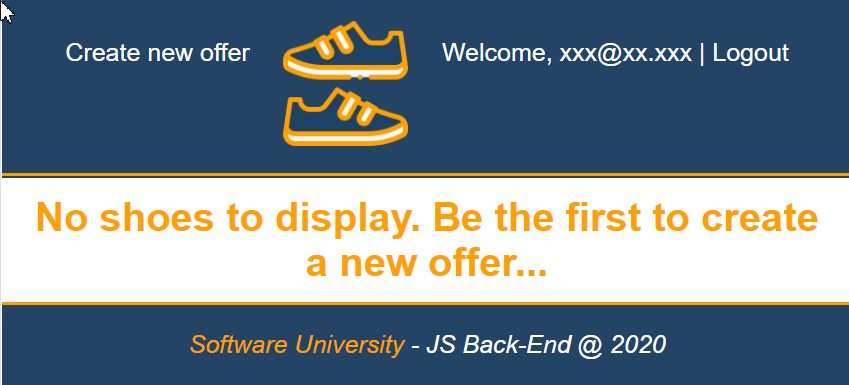
**Logout (5 Pts)**

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

* The **"logout" service** at the back-end **must** be called at logout
* All local information in the browser 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.

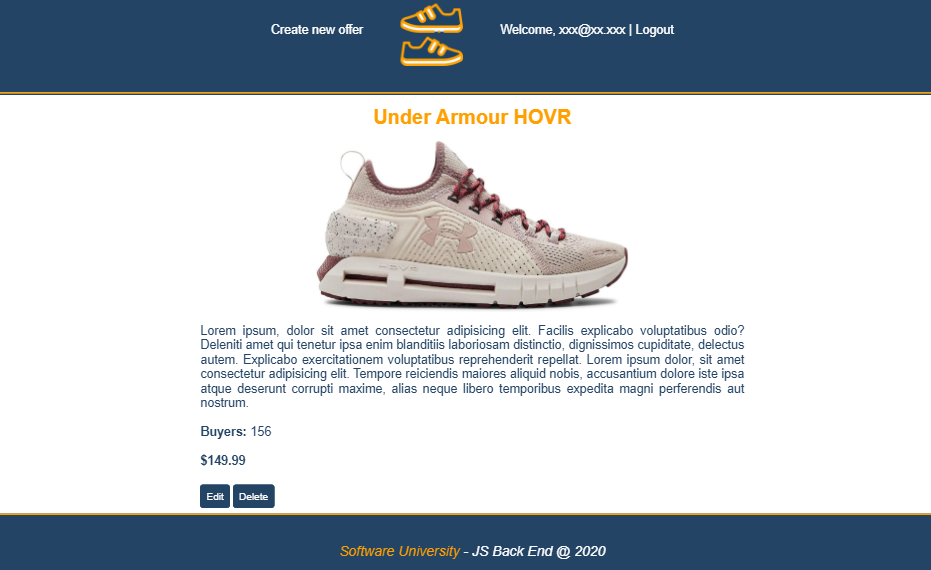
* The form should contain the following validations:
  + All **input** fieldsshouldn’t be **empty**.
  + **By** **default**, every newly created offer must have additional information:
    - **Creator:** string representing the current user;
    - **People bought it:** keeping data of users bought the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

**Details (15 Pts)**

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

Clicking the on a **particular offer [Buy it for ##.##] button** 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].

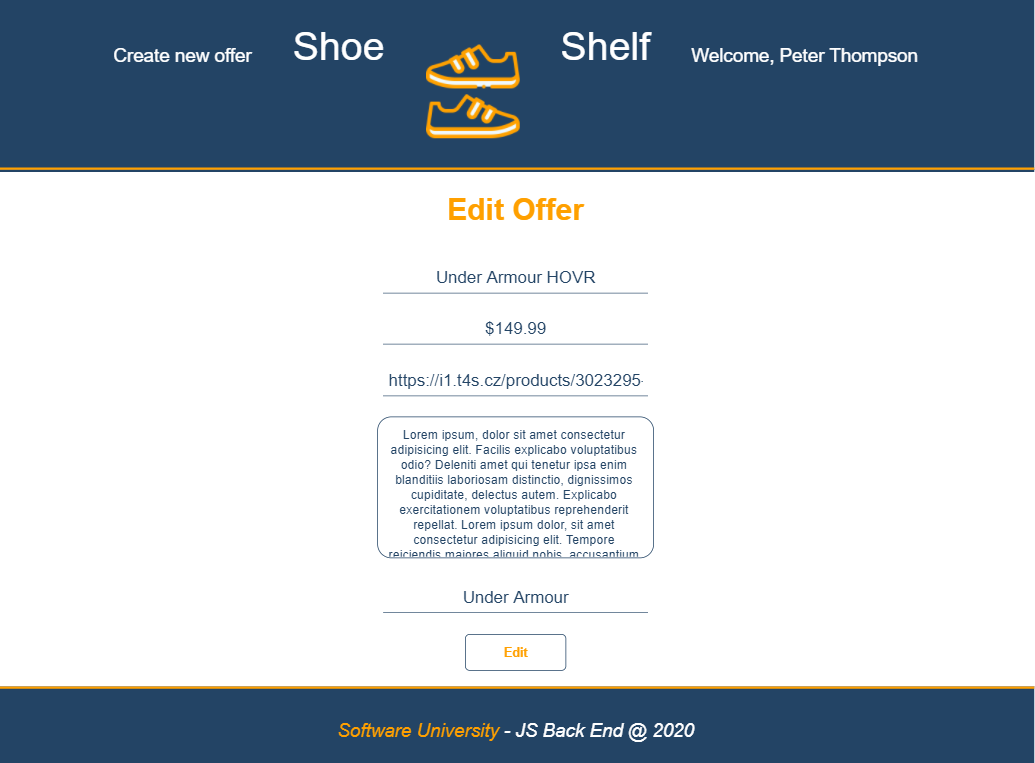




**Edit Offer (10 Pts)**

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.

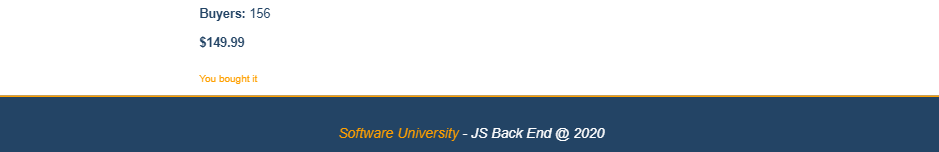
**Buy Offer (10 Pts)**

Logged-in users should be able to **buy offer**, 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 buyers.** After **successfully buying**:

* Display the updated Details page
* [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**.

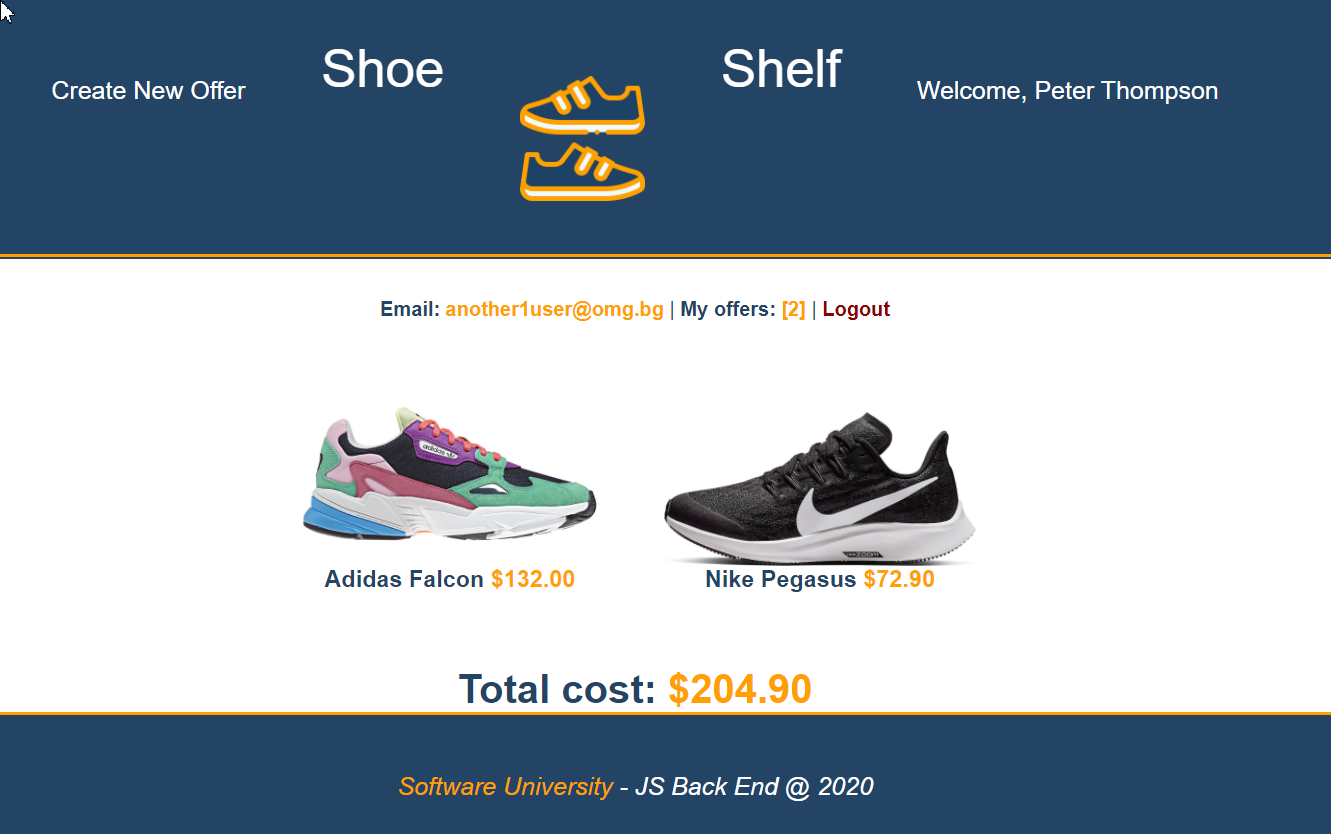
* After **successful** **delete** - Home pageshould be **shown**

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

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

**(BONUS) Profile Page: (10 Pts)**

Implement the logic for the **user profile page**. You should display **all bought offers** for this user and to calculate the **total cost** he has to pay.



**Security Requirements (10 Pts)**

The Security Requirements are mainly **access** requirements. Configurations about which users can access specific functionalities and pages.

* Guest (not logged in) users can access Home page and functionality.
* Guest (not logged in) users can access Login page and functionality.
* Guest (not logged in) users can access Register page and functionality.
* Users (logged in) can access Home page (Listed all Shoes)page and functionality.
* Users (logged in) can access Details page and functionality.
  + **Users** (not creator) can **Buy** a pair of shoes **once**
  + **Users** (creator) can **Edit** and **Delete** the current offer
* Users (logged in) can access **Create Offer** page and functionality.
* Users (logged in) can access Logout functionality.

**Validation and Error Handling (10 Pts)**

The application should notify the users about result of their actions.

### Login / Register

You should make the following validations:

* The **email** should be **at least 3 characters** long and should consist only **english** **letters** and **digits**
* The **password** should be **at least 3 characters** long and should consist only **english** **letters** and **digits**
* The **repeat password** should be **equal to the password**

### Offers

You should make the following validations while **creating** or **editing a** **pair of shoes**:

* The **name** should **not be empty**
* The **price** should **not be empty**
* The **imageUrl** should **not be empty**

## 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!!!**