Market – SPA Application – JS Apps Exam

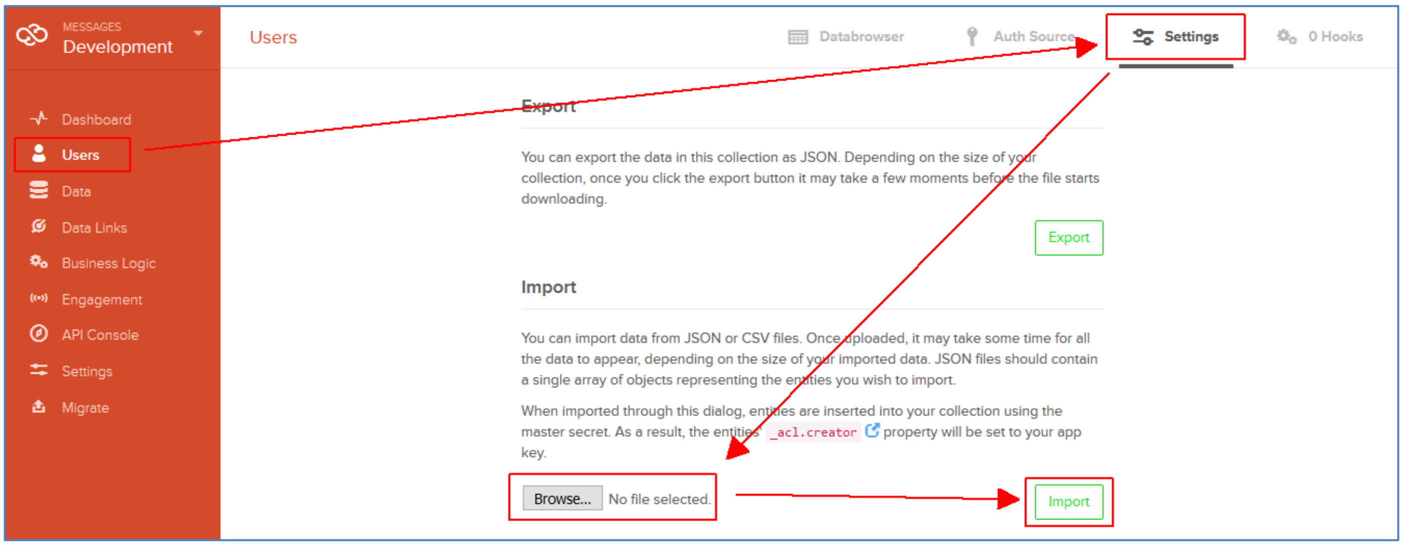
You are assigned to implement a **Shopping Web front-end application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Kinvey). The app keeps **users** and **products**. Users can **register**, **login**, view the **products**, **purchase products** (**adding** them to their **cart**), view their **cart**, **discard products** (from their cart) and **logout**.

Using libraries like jQuery and React is allowed but is not obligatory.

Problem 1. Create a Kinvey REST Service to Hold Users and Products

Register at **Kinvey.com** and create an application to keep your data in the cloud.

In the **Users** collection, import the provided JSON file with sample users to get started with template data. In the **Kinvey Console**, select **Users** from the navigation of the left, click **Settings** in the upper right then scroll down to the **Import** section:



Export

You can export the data in this collection as JSON. Depending on the size of your collection, once you click the export button it may take a few moments before the file starts downloading.

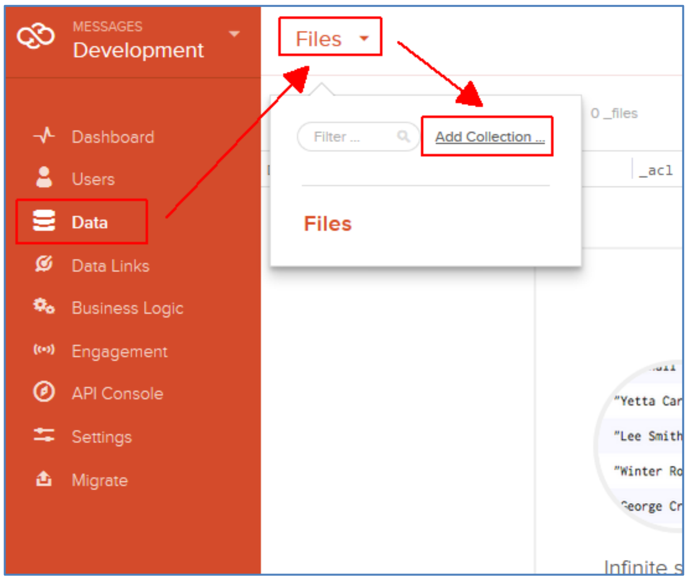
Import

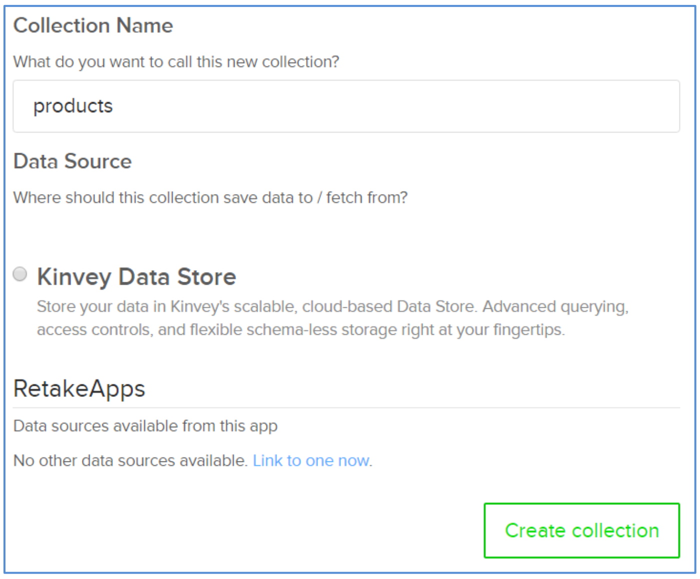
You can import data from JSON or CSV files. Once uploaded, it may take some time for all the data to appear, depending on the size of your imported data. JSON files should contain a single array of objects representing the entities you wish to import.

When imported through this dialog, entities are inserted into your collection using the master secret. As a result, the entities' `_acl.creator` property will be set to your app key.

Create a collection **products(name, description, price)** to hold the products. All the fields will hold text values, except the price - it will hold a numeric value. The "**_kmd.lmt**" field is automatically created by Kinvey and will hold a date and time in the traditional for JavaScript dates ISO8601 format returned by Date.toJSON().

Create a new collection and import the provided JSON file with sample **products** like shown below:





Collection Name

What do you want to call this new collection?

products

Data Source

Where should this collection save data to / fetch from?

Kinvey Data Store

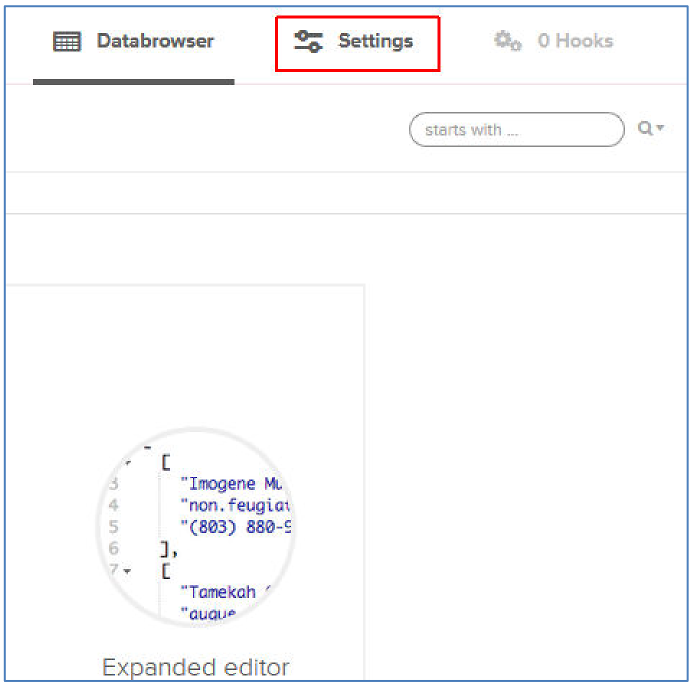
Store your data in Kinvey's scalable, cloud-based Data Store. Advanced querying, access controls, and flexible schema-less storage right at your fingertips.

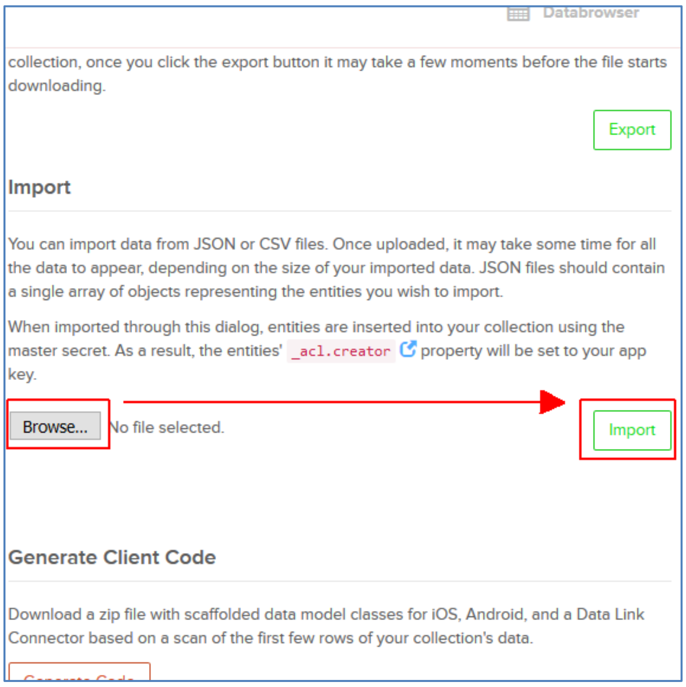
RetakeApps

Data sources available from this app

No other data sources available. Link to one now.

Create collection





Databrowser

collection, once you click the export button it may take a few moments before the file starts downloading.

Export

Import

You can import data from JSON or CSV files. Once uploaded, it may take some time for all the data to appear, depending on the size of your imported data. JSON files should contain a single array of objects representing the entities you wish to import.

When imported through this dialog, entities are inserted into your collection using the master secret. As a result, the entities' `_acl.creator` property will be set to your app key.

Browse... No file selected.

Import

Generate Client Code

Download a zip file with scaffolded data model classes for iOS, Android, and a Data Link Connector based on a scan of the first few rows of your collection's data.

Generate Code

Kinvey will automatically create **REST services** to access your data.

Problem 2. Test the Kinvey REST Services

Using **Postman** or other HTTP client tool (you can use Kinvey's built-in **API Console**), test the REST service endpoints:

User Registration (Sign Up)

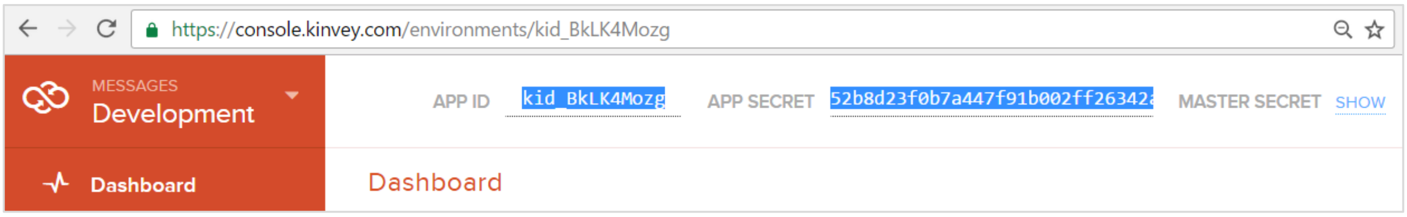
POST https://baas.kinvey.com/user/***app_id***/	
Request headers	Authorization: Basic base64(app_id:app_secret) Content-Type: application/json

Request body	{
	  "username": "new_user",
	  "password": "pass123",
	  "name": "New User"
	}

Response
201 Created
	{
	"_id": "583f53bde004a9a90983f1b7",
	"username": "new_user",
	"password": "pass123",
	"name": "New User",
	…
	}

Error response 409 Conflict	{ "error": "UserAlreadyExists", "description": "This username is already taken. Please retry your request with a different username", "debug": "" }
Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

The request needs "**Basic**" authentication. Use the Kinvey **app_id** and Kinvey **app_secret** as credentials.


https://console.kinvey.com/environments/kid_BkLK4Mozg

APP ID kid_BkLK4Mozg

APP SECRET 52b8d23f0b7a447f91b002ff26342

MASTER SECRET SHOW

User Login

POST [https://baas.kinvey.com/user/***app_id***/login](https://baas.kinvey.com/user/app_id/login)	
Request headers	Authorization: Basic base64(app_id:app_secret) Content-Type: application/json

Request body	{
	"username": "new_user",
	"password": "pass123"
	}

Response
200 OK
	{
	"_id": "583f53bde004a9a90983f1b7",
	"username": "new_user",
	"name": "New User",
	"_kmd": {
	"**authtoken**": "**8e6471bc-3712-4cfb-b92e-50e62a0c80….Duj5fHdM /7XHIe6KdY=**"
	…
	},
	…
	}

Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

Successful login returns an "**authtoken**" which is later used to authenticate the CRUD operations.

User Logout

POST [https://baas.kinvey.com/user/***app_id***/_logout](https://baas.kinvey.com/user/app_id/_logout)	
Request headers	Authorization: Kinvey **authtoken**
Response 204 No Content	
Error response	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request

401 Unauthorized	with correct credentials", "debug": "" }

To logout, you need to provide the "**authtoken**" given by login / register as "**Kinvey**" authorization header.

Get All Products (Shop)

GET https://baas.kinvey.com/appdata/***app_id***/products	
Request headers	Authorization: Kinvey authtoken

Response
200 OK
	```
	[
	 {
	  "_id": "5858699d4ad56c1314c48e96",
	  "name": "Apple",
	  "description": "An apple a day keeps the doctor away.",
	  "price": 0.5,
	  "_acl": {"creator": "kid_S1Oq7JU4g"},
	  "_kmd": {"lmt": "2016-12-19T23:13:33.195Z"…}
	 },
	 {
	  "_id": "585876b24ad56c1314c50eef",
	  "name": "Kroasan",
	  "description": "Chichipipikakao",
	  "price": 1.1,
	  "_acl": {"creator": "kid_S1Oq7JU4g"},
	  "_kmd": {"lmt": "2016-12-20T00:09:22.624Z"…}
	 }, …
	]
	```

Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

List Particular User

GET https://baas.kinvey.com/user/***app_id/user_Id***	
Request headers	Authorization: Kinvey authtoken

Response
200 OK
	```
	[
	    {
	      "_id": "5858679fe12ae039723fb628",
	      "username": "pesho",
	      "name": "Pesho Peshov",
	      "_acl": {
	        "creator": "5858679fe12ae039723fb628"
	      },
	      "_kmd": {
	        "lmt": "2016-12-20T01:09:51.514Z",
	        "ect": "2016-12-19T23:05:03.691Z"
	      },
	      "cart": {}
	    }
	]
	```

NOTE: The cart is kept in the **`Users`** collection. Every user has a **`cart`** field which corresponds to his cart. The cart is an **object** – **initally empty**. You will see how it is used later.

Update Particular User

PUT https://baas.kinvey.com/user/***app_id/user_Id***

Request headers	Authorization: Kinvey authtoken

Request body

```
{
	"_id": "5858679fe12ae039723fb628",
	"username": "pesho",
	"name": "Pesho Peshov",
	"_acl": {
		"creator": "5858679fe12ae039723fb628"
	},
	"_kmd": {
		"lmt": "2016-12-20T01:09:51.514Z",
		"ect": "2016-12-19T23:05:03.691Z"
	},
	"cart": {
		"585876e5b91c66ad2607e865": {
			"quantity": "1",
			"product": {
				"name": "Toilet Paper",
				"description": "IsSoft",
				"price": "4.10"
			}
		}
	}
}
```

Response
200 OK

```
[
	{
		"_id": "5858679fe12ae039723fb628",
		"username": "pesho",
		"name": "Pesho Peshov",
		"_acl": {
			"creator": "5858679fe12ae039723fb628"
		},
		"_kmd": {
			"lmt": "2016-12-20T01:09:51.514Z",
			"ect": "2016-12-19T23:05:03.691Z"
		},
		"cart": {
			"585876e5b91c66ad2607e865": {
				"quantity": "1",
				"product": {
					"name": "Toilet Paper",
					"description": "IsSoft",
					"price": "4.10"
				}
			}
		}
	}
]
```

NOTE: The PUT request, **updates** the **object** in the **back-end** with whatever you send as a **body**. Note that you can send everything, but whatever you send, will **REPLACE** whatever is currently on the back-end at the **current id**.

Upon updating, make sure you first send a **GET** request, to **get the current entity**, and **change the data** you **receive**, for example the **cart**, and then **send** the **same data back**. Otherwise you might break the **back-end data**.

Problem 3. Market – HTML and CSS

You are given the Web design of the Market application as **HTML** + **CSS** files.

- Initially all views and forms are shown by the HTML. Your application may **hide** by CSS (display: none) or **delete** from the DOM all unneeded elements or just display the views it needs to display.
- You may render the views / forms / components with React, jQuery, Mustache or another UI library.

Important: don't change the elements' **class name** and **id**. Don't rename form fields / link names / ids.

Problem 4. Market Client-Side Web Application

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

Navigation System

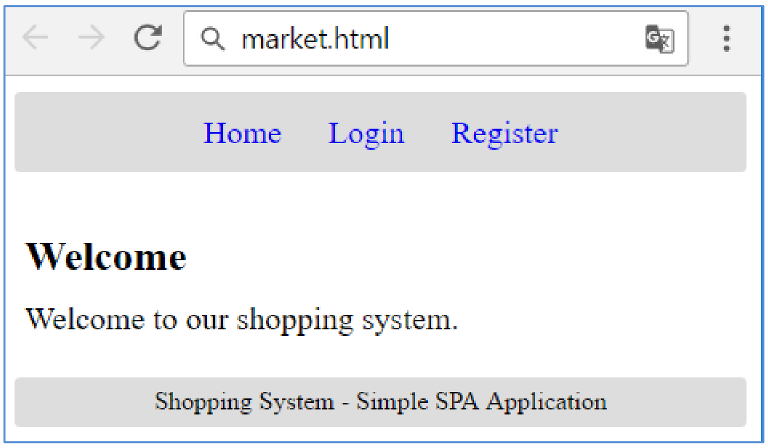
Implement a **navigation system** for the app: navigation links should correctly change the current screen (view).

- Clicking on the links in the **top navigation bar** should display the view behind the link (views are sections in the HTML code).
- Your application may **hide** by CSS (display: none) or **delete** from the DOM all unneeded elements or just display the views it needs to display.

5 score

Home Screen

When no user is logged in, the app should display the "Home" screen holding a welcome message + three links: **[Home]**, **[Login]** and **[Register]**.


market.html

Home Login Register

Welcome

Welcome to our shopping system.

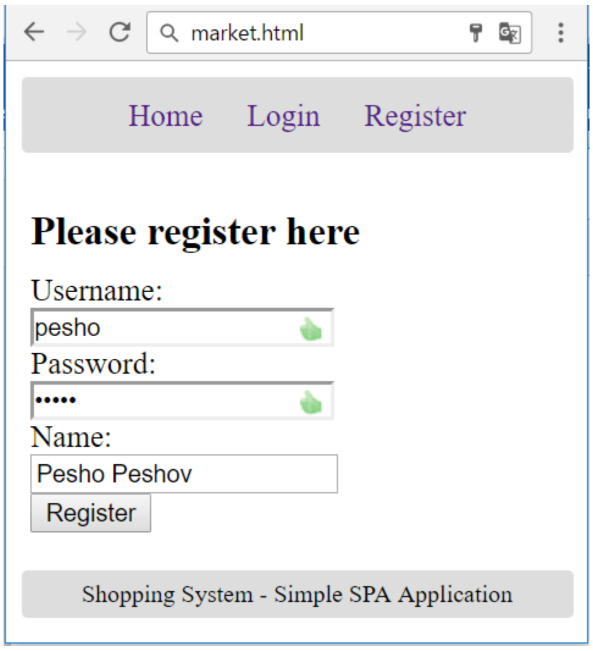
Shopping System - Simple SPA Application

5 score

Register User Screen

By given **username** + **password** + **name** the app should register a new user in the system.

- After a **successful registration**, a notification message "**User registration successful.**" should be displayed and the user home screen should be displayed.
- In case of **error**, an appropriate error message should be displayed and the user should be able to try to register again.
- **Form validation** is already implemented in the HTML, so you don't need to add it.
- Keep the user session data in the browser's **session storage**.


market.html

Home Login Register

Please register here

Username: pesho

Password: •••••

Name: Pesho Peshov

Register

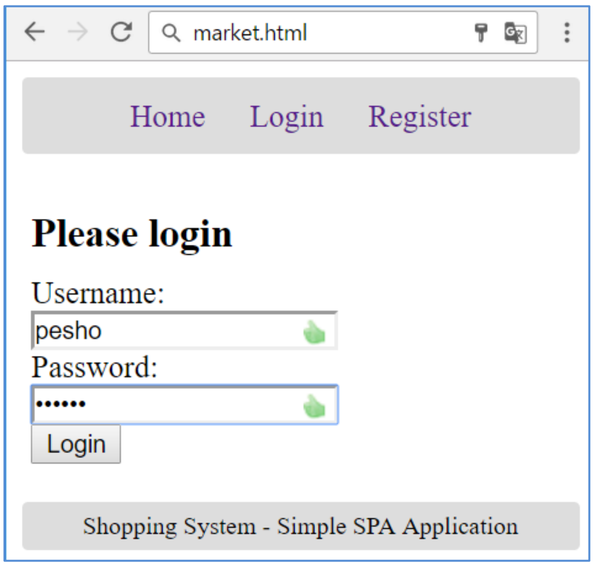
Shopping System - Simple SPA Application

10 score

Login User Screen

By given **username** and **password** the app should be able to login an existing user.

- After a **successful login**, a notification message "Login successful." should be displayed and the user home screen should be displayed.
- In case of **error**, an appropriate error message should be displayed and the user should be able to fill the login form again.
- **Form validation** is already implemented in the HTML, so you don't need to add it.
- Keep the user session data in the browser's **session storage**.


market.html

Home Login Register

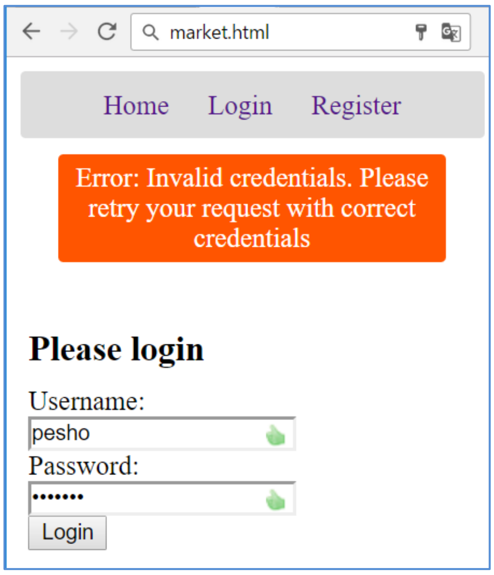
Please login

Username: pesho

Password: ••••••

Login

Shopping System - Simple SPA Application


market.html

Home Login Register

Error: Invalid credentials. Please retry your request with correct credentials

Please login

Username:

pesho

Password:

••••••

Login

5 score

Logout

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

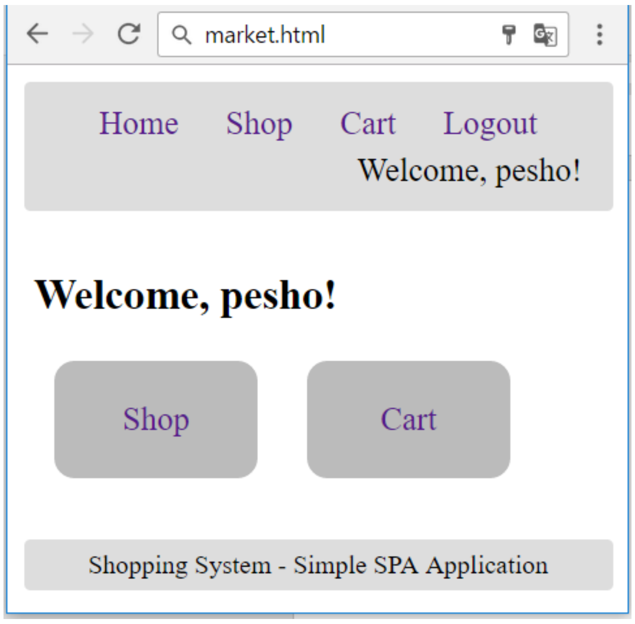
- After a **successful** logout, a **notification** message "Logout successful." should be displayed.
- After successful logout, the **Home screen** should be shown.
- The **"logout" REST service** at the back-end should be obligatory called at logout.
- All local information in the browser (**user session data**) about the current user should be deleted.

5 score

User Home Screen

After successful login, the app should display the **user's home screen**.

- It should hold a message **"Welcome, " + the username** of the current user.
- At the top navigation bar display the navigation links **[Home]**, **[Shop]**, **[Cart]** and **[Logout]** + "**Welcome, {username}**".
- At the main view area display navigation boxes **[Shop]**, **[Cart]**.


market.html

Home Shop Cart Logout

Welcome, pesho!

Welcome, pesho!

Shop Cart

Shopping System - Simple SPA Application

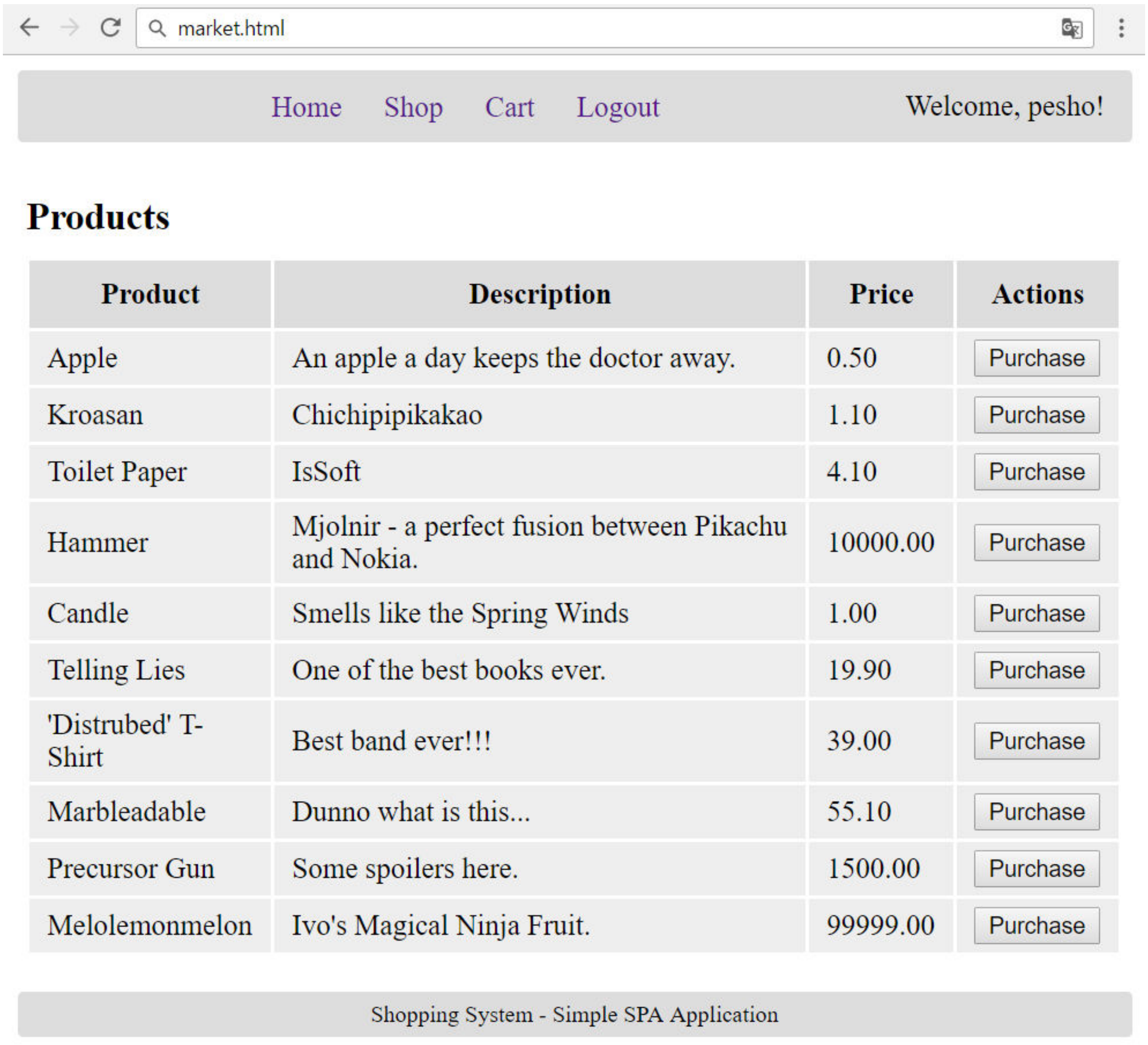
- Ensure you handle properly all HTML **special characters**, e.g. the username could be "*<pesho><br>*".

5 score

Shop Products Screen

Successfully logged users after clicking the **[Shop]** link should be able to view all products.

- The products should be listed in the **format** as shown in the Web design (see the screenshot below).
- In case of **error** (e.g. Internet connection lost), an error message should be displayed.
- Display **[Purchase]** button for each product in the shop. The button will purchase the product, adding it to the **user's cart**.
- Thought this case is quite impossible… In case of **no products**, display an empty table (header row only).
- All prices should be **rounded**, the **default way** (**`0.505`** == **`0.51`**, **`0.504`** == **`0.50`**) to the **second digit** after the **decimal point**, and printed the same way.



Products

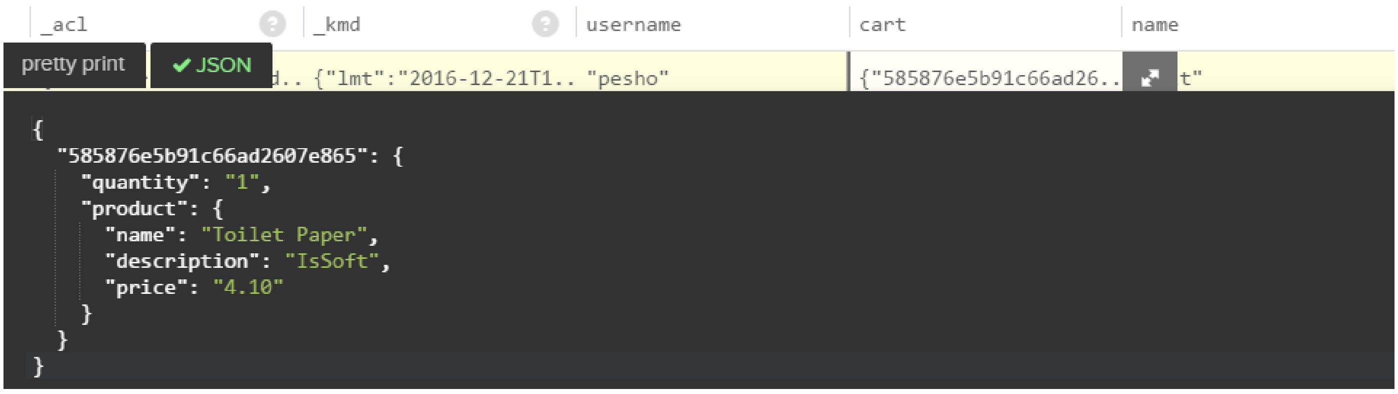
Product	Description	Price	Actions
Apple	An apple a day keeps the doctor away.	0.50	Purchase
Kroasan	Chichipipikakao	1.10	Purchase
Toilet Paper	IsSoft	4.10	Purchase
Hammer	Mjolnir - a perfect fusion between Pikachu and Nokia.	10000.00	Purchase
Candle	Smells like the Spring Winds	1.00	Purchase
Telling Lies	One of the best books ever.	19.90	Purchase
'Distrubed' T-Shirt	Best band ever!!!	39.00	Purchase
Marbleadable	Dunno what is this...	55.10	Purchase
Precursor Gun	Some spoilers here.	1500.00	Purchase
Melolemonmelon	Ivo's Magical Ninja Fruit.	99999.00	Purchase

Shopping System - Simple SPA Application

15 score

Cart Products Screen

The **Cart functionality** is quite simple for **implementing**. When you must store products in the cart, upon purchase... You should store in the **`cart`** field, which is an **object**, the **id of the particular product**, as **a key** (**property**) and the **product's quantity**, and **product data**, as a **value** (of the **property**). In the end it would look like this.



```
d.. {"lmt":"2016-12-21T1.. "pesho"   {"585876e5b91c66ad26..   t"
```

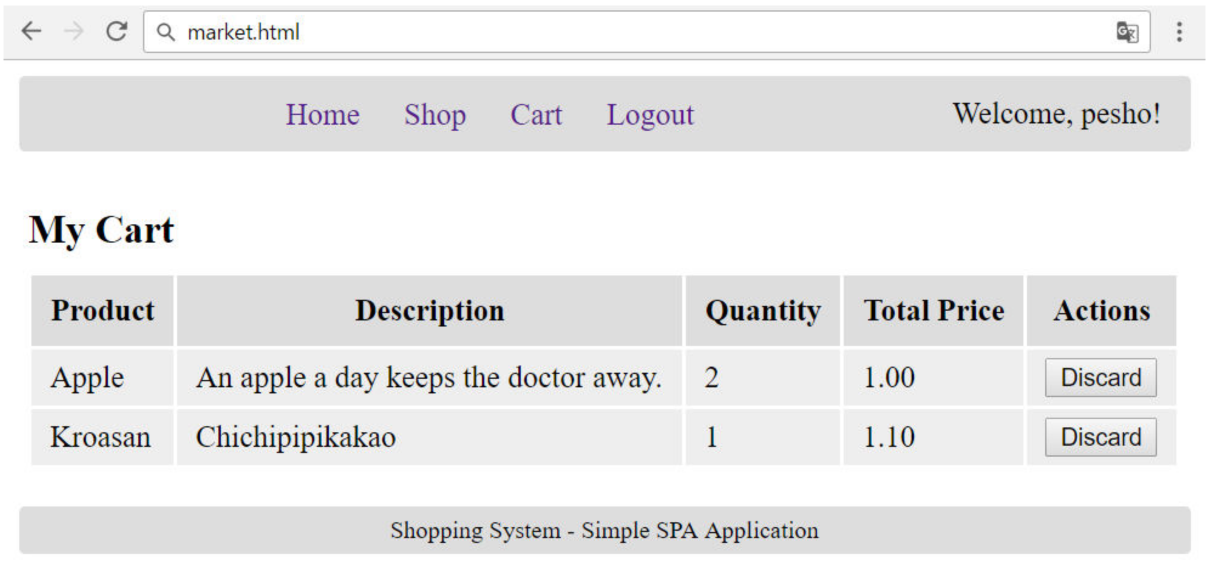
```
{
  "585876e5b91c66ad2607e865": {
    "quantity": "1",
    "product": {
      "name": "Toilet Paper",
      "description": "IsSoft",
      "price": "4.10"
    }
  }
}
```

If you have multiple products, they are to be stored, each with its **id** (**as the key**), and the **quantity** and **product data** as an **object** (**as the value**). You have been given several users with multiple products in their carts. Use them as test data.

That is why you were given an **`Update end-point`**. So you could **update** the **cart** of a **user**, **adding** or **removing** a **property** from the **`cart` object**, or just **increasing** its **quantity**.

Successfully logged users after clicking the **[Cart]** link should be able to view all products, purchased by the **current user**, i.e. **products** which are currently in the user's **cart**.

- The **products** should be listed **as shown in the Web design** (see the screenshot below).
- In case of **error** (e.g. Internet connection lost), an error message should be displayed.
- Display **[Discard]** button for each product in the cart. The button will discard the product, removing it from the **user's cart**.
- In case of **no products**, display an empty table (header row only).
- All prices should be **rounded**, the **default way** (**`0.505`** == **`0.51`**, **`0.504`** == **`0.50`**) to the **second digit** after the **decimal point**, and printed the same way.



My Cart

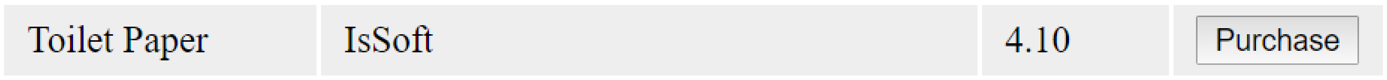
Product	Description	Quantity	Total Price	Actions
Apple	An apple a day keeps the doctor away.	2	1.00	Discard
Kroasan	Chichipipikakao	1	1.10	Discard

Shopping System - Simple SPA Application

20 score

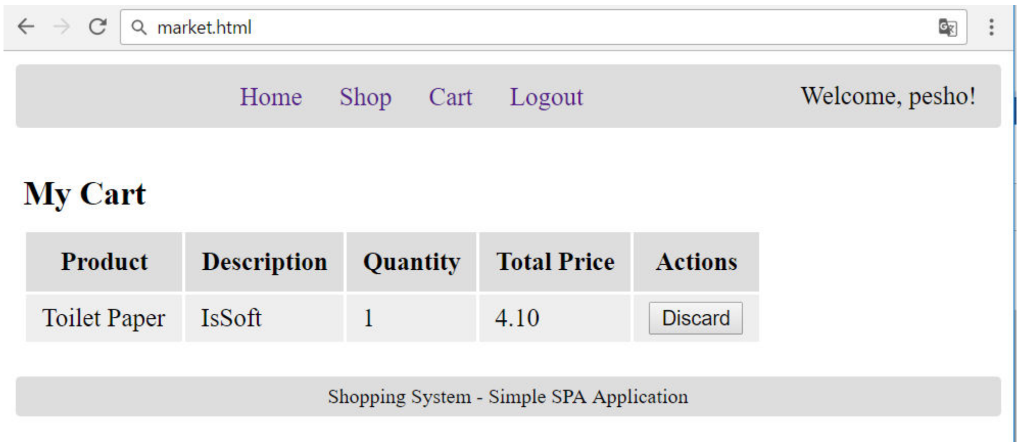
Purchase Product

Successfully logged in users should be able to **purchase products** by choosing a **product**, from the Shop products, and clicking the **[Purchase]** button.



Toilet Paper	IsSoft	4.10	Purchase

- After a **successful** product purchase, a notification message “**Product purchased.**” should be displayed and **the cart** (**user’s cart**) should be shown.
- In case of **error**, an appropriate error message should be displayed.
- Users are allowed to **purchase a product**, **more** than **once**, which is why the **`Quantity`** parameter stays in the table of the **Cart** view. The **`Total Price`** should be equal to the **`product_price * quantity`**.



My Cart

Product	Description	Quantity	Total Price	Actions
Toilet Paper	IsSoft	1	4.10	Discard

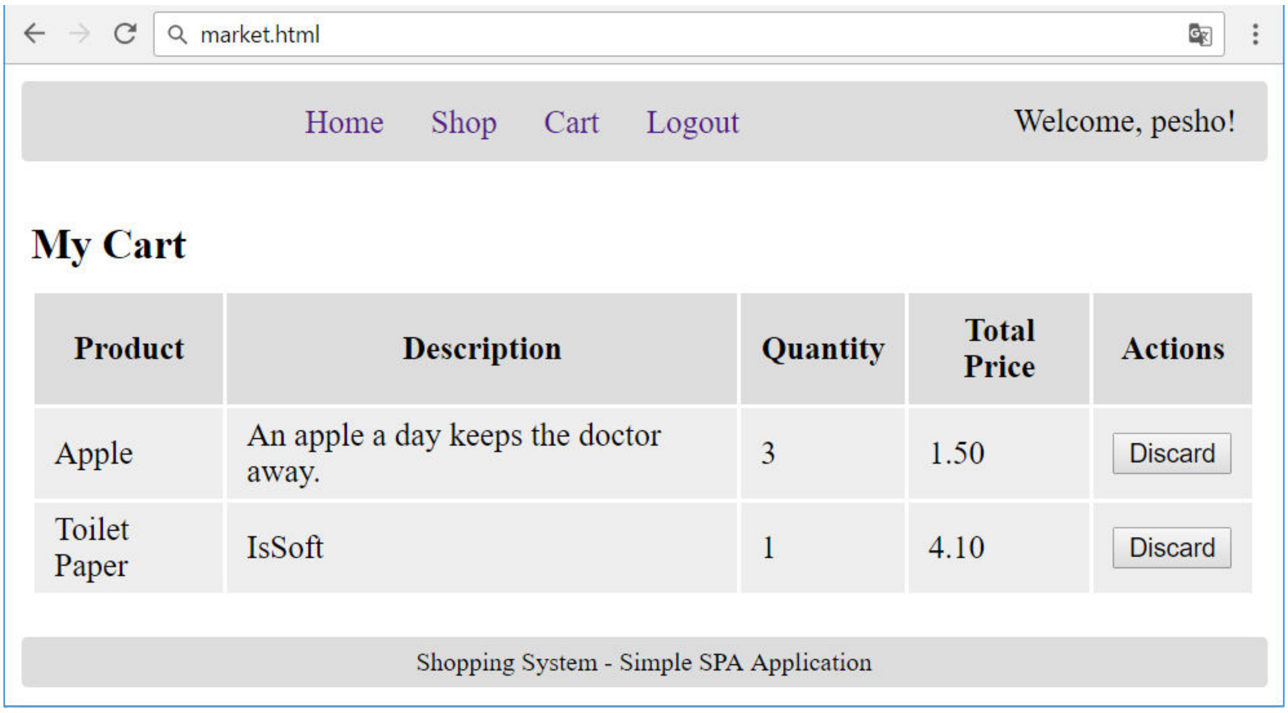
Shopping System - Simple SPA Application

15 score

Discard Product

Successfully logged in users should be able to **discard the products they purchased** by clicking on the **[Discard]** button in the table of product in the **Cart** view.

- After **successful** product discard a notification message “**Product discarded.**” should be shown.
- In case of **error** (e.g. Internet connection lost / unauthorized request / missing **product**), an error message should be displayed.
- The Deletion, should delete the **whole product**, **regardless** of its **quantity**.

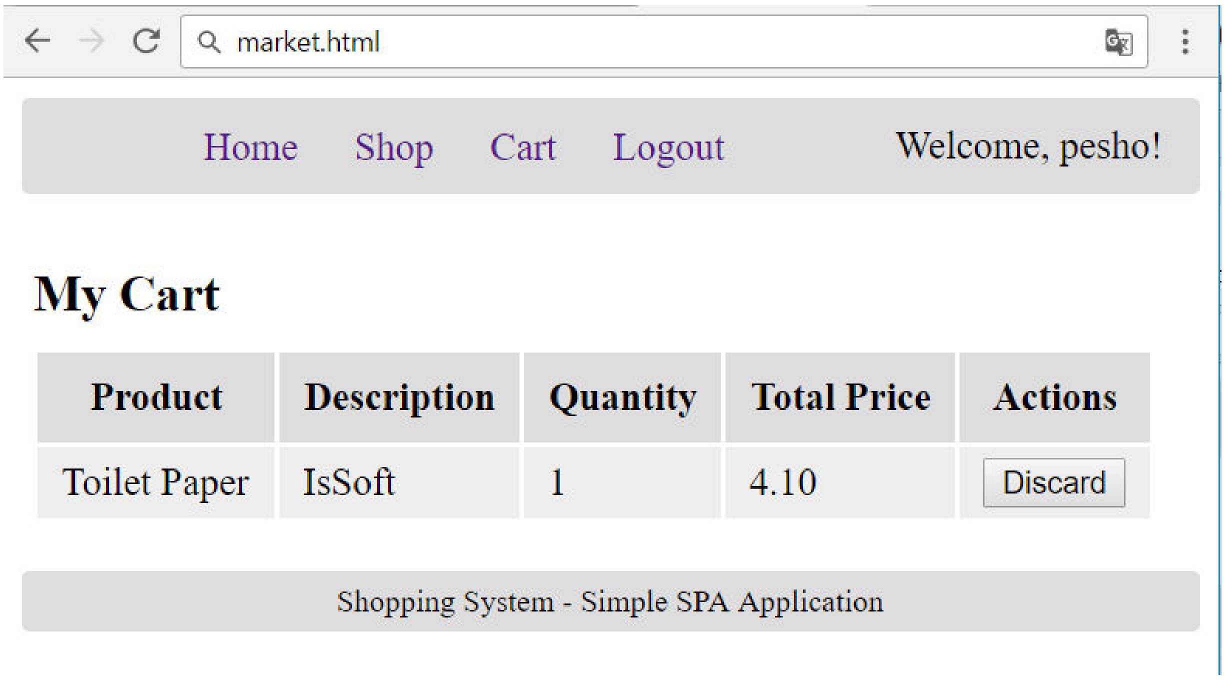

market.html

Home Shop Cart Logout Welcome, pesho!

My Cart

Product	Description	Quantity	Total Price	Actions
Apple	An apple a day keeps the doctor away.	3	1.50	Discard
Toilet Paper	IsSoft	1	4.10	Discard

Shopping System - Simple SPA Application


market.html

Home Shop Cart Logout Welcome, pesho!

My Cart

Product	Description	Quantity	Total Price	Actions
Toilet Paper	IsSoft	1	4.10	Discard

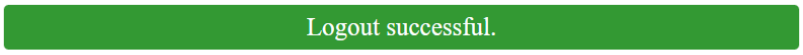
Shopping System - Simple SPA Application

15 score

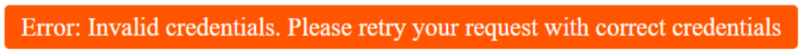
Notifications

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

- In case of successful action an **informational (green) notification message** should be shown, which disappears automatically after 3 seconds or manually when the user clicks it.


Logout successful.

- In case of **error**, an **error notification message** (red) should be shown which disappears on user click.


Error: Invalid credentials. Please retry your request with correct credentials

- During the AJAX calls a **loading notification message (blue)** should be shown. It should disappear automatically as soon as the AJAX call is completed.

Loading ...

Good luck!