# JS Apps Exam – Point Of Sale Single Page Application

You are assigned to implement a **Web application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Kinvey). The **app** that keeps **users** (cashiers), **product entires** and **receipts**. Users can **register**, **login**, **logout**, access the **main view** where a **receipt** can be **composed** (add products with their **qunatity** and **price** and save the basket to the database), list of all **receipts** and a **receipt details** view.

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

## Create a Kinvey REST Service

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

Create a collection **entries**. Each product has **type**, **qty**, **price** and **receiptId**.

Create a collection **receipts**. Each receipt has an **active** property, initially set to **true**, **productCount** and **total**.

## Test the Kinvey REST Services

### Common Responses

**Note:** When creating or updating records, the response will contain the **entire record** body, as it appears in the database. It’s advisable if you observe network traffic via Postman or using your browser’s dev-tools, to view details about each request.

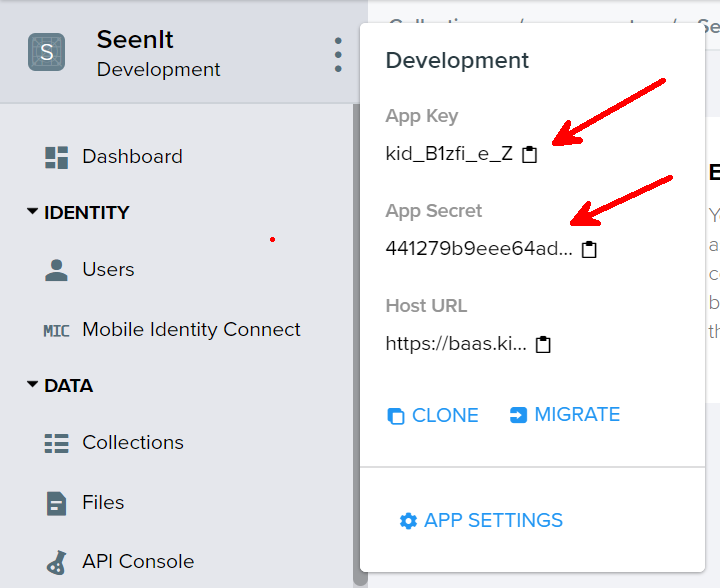
|  |  |
| --- | --- |
| **Response Code** | **Response Body** |
| 200 OK | *<Record data>* |
| 201 Created | *<Record data>* |
| 204 No Content | *<Empty>* |
| 401 Unauthorized | {  "error": "InvalidCredentials",  "description": "Invalid credentials. …",  "debug": ""  } |
| 404 Not Found | {  "error": "EntityNotFound",  "description": "This entity not found in the collection",  "debug": ""  } |
| Error response  409 Conflict | {  "error": "UserAlreadyExists",  "description": "This username is already taken. …",  "debug": ""  } |

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\_key***/ | |
| Request headers | Authorization: Basic base64(app\_id:app\_secret)  Content-Type: application/json |
| Request body | {  "username": "testuser",  "password": "testuserpass890"  } |

The request needs “**Basic**” authentication. Use the Kinvey **App Key** and Kinvey **App Secret** as credentials.



### User Login

|  |  |
| --- | --- |
| **POST** https://baas.kinvey.com/user/***app\_key***/login | |
| Request headers | Authorization: Basic base64(app\_id:app\_secret)  Content-Type: application/json |
| Request body | {  "username": "testuser",  "password": "testuserpass890"  } |

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

### User Logout

|  |  |
| --- | --- |
| **POST** https://baas.kinvey.com/user/***app\_key***/\_logout | |
| Request headers | Authorization: Kinvey **authtoken** |

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

### Get Active Receipt

|  |  |
| --- | --- |
| **GET** https://baas.kinvey.com/appdata/**app\_key**/receipts?query={"\_acl.creator":"**userId**","active":"**true**"} | |
| Request headers | Authorization: Kinvey authtoken |

This will return the receipt that’s **active** for the currently **logged in user**. Use this to populate the **Editor**, or if it’s not found – create a new receipt and set it to be active.

### Get Entries by Receipt ID

|  |  |
| --- | --- |
| **GET** https://baas.kinvey.com/appdata/**app\_key**/entries?query={"receiptId":"**receiptId**"} | |
| Request headers | Authorization: Kinvey authtoken |

You may use this query to get all entries of the currently **active receipt**, or entries for **receipt details**.

### Create Receipt

|  |  |
| --- | --- |
| **POST** https://baas.kinvey.com/appdata/***app\_key***/receipts | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "active": true,  "productCount": 0,  "total": 0  } |

### Add Entry

|  |  |
| --- | --- |
| **POST** https://baas.kinvey.com/appdata/***app\_key***/entries | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "type": "Apple",  "qty": 5,  "price": 0.3,  "receiptId": "59affdae3044bb86044a79bd"  } |

New entries should always be added to the **active receipt**.

### Delete Entry

|  |  |
| --- | --- |
| **DELETE** https://baas.kinvey.com/appdata/***app\_key***/entries/***entry\_id*** | |
| Request headers | Authorization: Kinvey authtoken |

### Get My Receipts

|  |  |
| --- | --- |
| **GET** https://baas.kinvey.com/appdata/**app\_key**/receipts?query={"\_acl.creator":"**userId**","active":"**false**"} | |
| Request headers | Authorization: Kinvey authtoken |

Use the ID of the currently **logged in user**. ***The user should see only his or her receipts on the overview screen.***

### Receipt Details

|  |  |
| --- | --- |
| **GET** https://baas.kinvey.com/appdata/**app\_key**/receipts/**receipt\_id** | |
| Request headers | Authorization: Kinvey authtoken |

### Commit Receipt

|  |  |
| --- | --- |
| **PUT** https://baas.kinvey.com/appdata/**app\_key**/receipts/**receipt\_id** | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "active": false,  "productCount": 0, *// Sum of all products*  "total": 0, *// Total cost of all products*  *// Other receipt properties*  } |

To mark a receipt as finalized (client has checked out), simply update it to set its **active** property to **false**. You need to send the whole receipt object, so don’t forget to fetch the receipt from the database first.

## HTML and CSS

You аre 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** 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 **jQuery** or **Handlebars**.

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

***Including the <section> elements is required for the style to display correctly!***

## Client-Side Web Application

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

### Notifications (10 pts)

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.



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



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



***Points for notifications are awarded separately for each section.***

### Navigation System (10 pts)

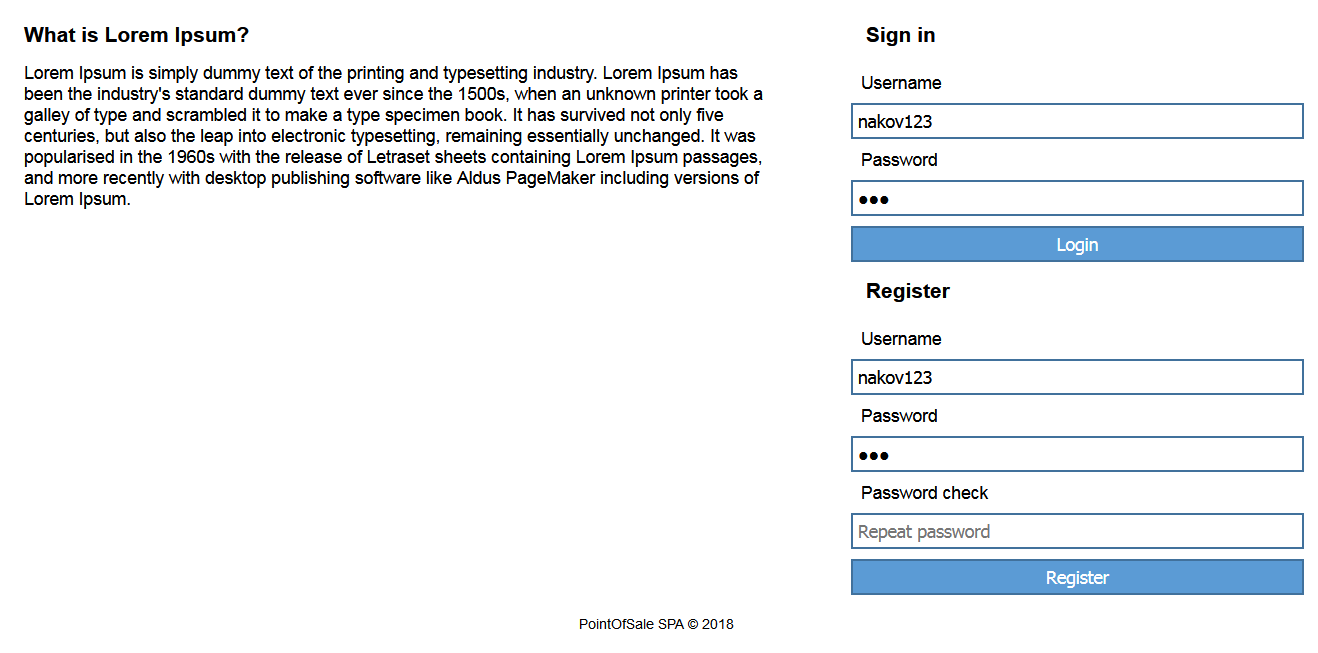
Implement a **navigation system** for the app: navigation links should correctly change the current screen (view).

* Clicking on the links in the **menu** or **individual** links should display the view behind the link (views are sections in the HTML code).
* The given „**Navigation**“ menu should be visible **only** for logged in users. Anonymous users can **only** view the **sign in/register** section.

### Register User Screen (5 pts)

By given **username**, **password and repeat password** 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 should be **redirected** to the home view.
* You **need** to validate the **input**. A username **should** be a string with at **least** 5 characters **long**. Passwords **input** fields shouldn’t be **empty**. Both passwords **should** match.
* In case of **error** (eg. invalid username/password), an appropriate error **message** should be displayed and the user should be able to **try** to register again.
* Keep the user session data in the browser’s **session storage**.
* Clear **all** input fields after **successful** register.



### Login User Screen (5 pts)

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 and the user should be **redirected** to the home view.
* 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** should be the **same** as register.
* Keep the user session data in the browser’s **session storage**.
* Clear **all** input fields after **successful** login.

### Logout (5 pts)

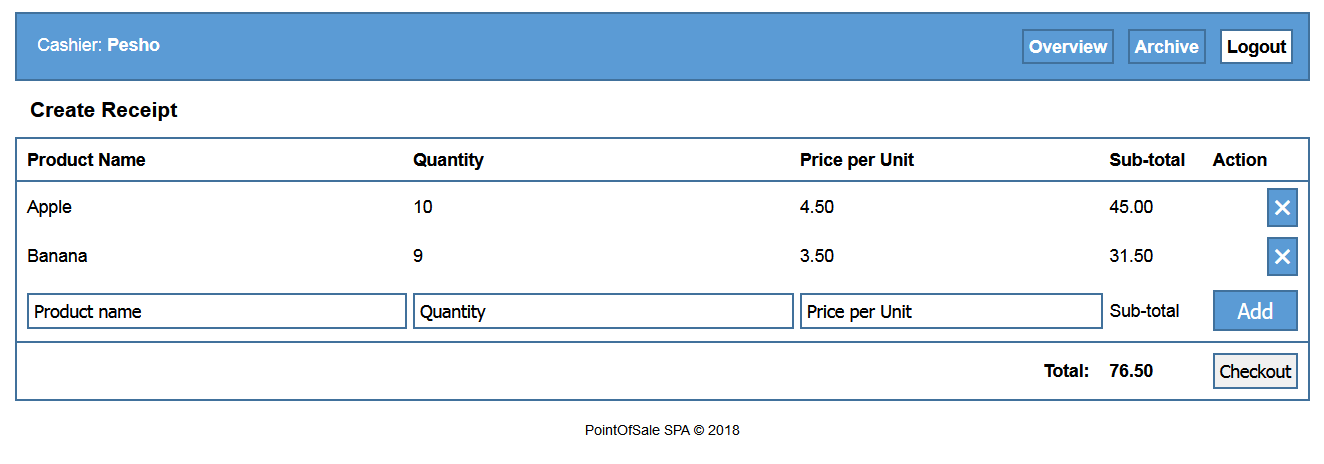
Successfully logged in user should be able to **logout** from the app.

* After a **successful** logout, a **notification** message “Logout successful.” should be displayed.
* After successful logout, the **Sign In 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.

### Home Screen (Receipt Editor) (45 pts)

##### Display Currently Active Receipt (15 pts)

Whenever the user opens the editor, you should retrieve the currently **active receipt** and all products related to it (by **receiptId**). If there is no active receipt, you must create it in the database. Note that the HTML contains **hidden input fields**, which you can use. ***There must be only one active receipt at any one time on the server!***



##### Add New Entry (10pts)

Clicking on Add creates a new entry, using the receipt ID of the currently active receipt and the data from the input fields. The fields must be validated:

* Product **name** must be a **non-empty string**
* **Quantity** must be a **number**
* **Price** must be a **number**

Update the value of **Sub-total** and **Total** in real time, whenever the user changes Quantity or Price to a valid value. Upon successfully adding the entry to the database

After successful entry creation, display a notification “Entry added”, **add the information** to the end of the list of entries and **clear all input values**.

##### Update Sub-total and Total (5 pts)

When the user enters a **valid value** for **Quantity** and **Price per Unit**, the displayed values for **Sub-total** for the new entry and **Total** for the receipt should be updated.

##### Remove Entry (5 pts)

Clicking the **delete button** next to each entry must **remove it** from the database and **delete the row** from the table. After successful deletion, **update** the value of Total. Display a notification “Entry removed” and **remove the corresponding elements** from the list of entries.

##### Checkout Receipt (10 pts)

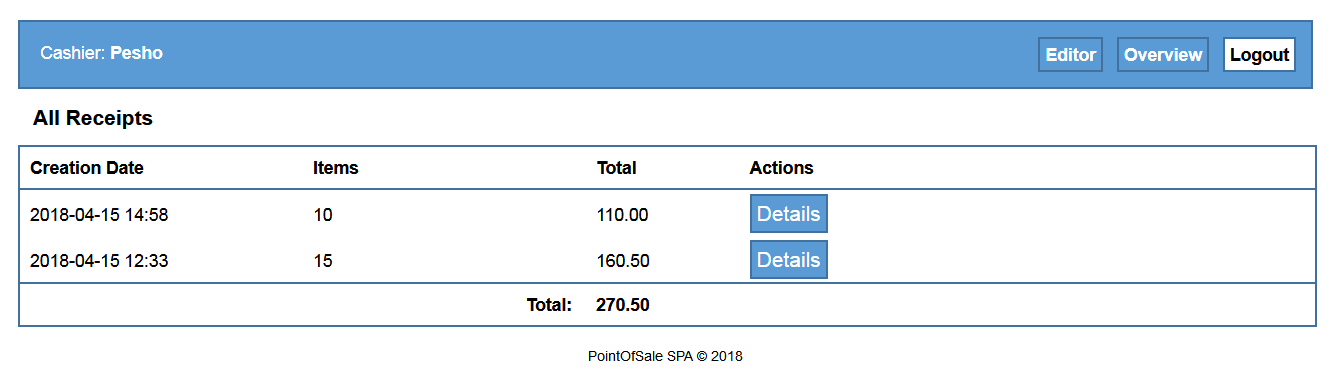
Clicking on **Checkout** should perform the following:

* Display a **notification** “Receipt checked out”
* Update the receipt in the database to have its **active** property set to **false** and the properties **productCount** and **total** populated with the correct values
* Prepare the editor for a **new receipt** by creating it in the database and **clearing the screen** of any old information.

Before carrying out any actions, make sure the receipt contains **at least one entry** – ***the user should not be able to checkout an empty receipt!***

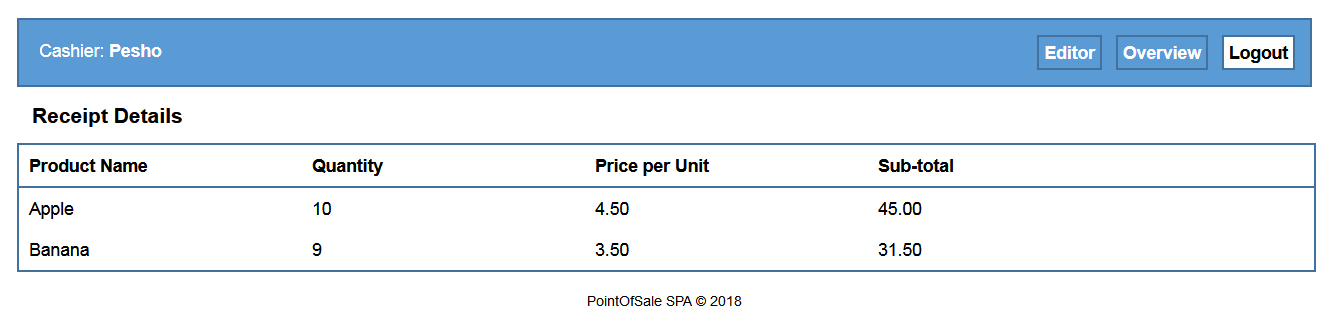
### All Receipts (10 pts)

Display a list of all receipts that the user has created. Use the stored user ID to retrieve only the relevant records. Every receipt must have a **link** that leads to its **details**. ***The user should see only his or her receipts.***



### Receipt Details (10 pts)

Display the selected receipt with a list of all entries in it. Use the receipt ID to filter only the related entries.



## Subtmitting Your Solution

Place in a ZIP file your project folder. Exclude the node\_modules folder. Upload the archive to the Judge.