# SoftUni Properties HTML5 Application – JS Apps Exam

You are assigned to design and implement a **real estate managment Web front-end application** using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend. The app keeps users and their products. Users can register, login, view everyone's products, add products, edit and delete their own products and logout. The app should be implemented as client-side Web application in JavaScript with server-side REST services called by AJAX and returning JSON objects.

## Create SoftUni Properties REST Services

Register at Parse.com and create an application to keep your data in the cloud. Create a class **Category(name)** to hold the category name and a class **Estate(name, category, price)** to hold estates. There are only 4 category types: **'Condo', 'Penthouse', 'Apartment', 'Studio'**. You are not allowed to add/edit/delete any category. Parse.com will automatically create REST services to access your data:

* **User Registration (Sign Up)**
  + Endpoint: <https://api.parse.com/1/users>, Method: POST
  + Request body (JSON): {"username":"*user*", "password":"*pass*"}
  + Returns (JSON): {…, "sessionToken":"*session\_token*"}
* **User Login**
  + Endpoint: <https://api.parse.com/1/login>, Method: GET
  + URL parameters: username=*user*, password=*pass*
  + Returns (JSON): {…, "sessionToken":"*session\_token*"}
* **List All Estates**
  + Endpoint: [https://api.parse.com/1/classes/](https://api.parse.com/1/classes/Phone)Estate, Method: GET
  + URL parameters: include=category *(for max points use pointers)*
  + Returns (JSON): {"results":[{"name":"…","category":"…",…}, …]}
* **Add Estate**
  + Endpoint: <https://api.parse.com/1/classes/Estate>, Method: POST
  + Request body (JSON): {"name":"*estate\_name*", "category":{…}, "price":*price*, "ACL":{…}}
  + Returns (JSON): {"createdAt":"…", "objectId":"…"}
* **Edit Estate**
  + Endpoint: [https://api.parse.com/1/classes/Estate/*estate\_objectId*](https://api.parse.com/1/classes/Estate/estate_objectId), Method: PUT
  + Request body (JSON): {"name":"*estate\_name*", "price":*price*}
  + Returns (JSON): {"updatedAt":"…", "objectId":"…"}
* **Delete Estate**
  + Endpoint: [https://api.parse.com/1/classes/Estate/*estate\_objectId*](https://api.parse.com/1/classes/Estate/estate_objectId), Method: DELETE
  + Returns (JSON): { }

All Parse.com REST services require the following **HTTP request headers**:

* X-Parse-Application-Id: *your\_parse\_app\_id*
* X-Parse-REST-API-Key: *your\_parse\_rest\_api\_key*

Notes about **users and authentication**:

* After register / login, pass the session token as HTTP request header to **authenticate your requests**:
  + X-Parse-Session-Token: *session\_token\_returned\_by\_login\_or\_register*
* When creating new objects, pass the following **ACL** to restrict the access to the user who created the object:
  + "ACL":{"*user\_оbjectId*":{"write":true,"read":true},"\*":{"read":true}}
  + Thus the object will be visible to everybody, but modifiable only by the user who created it

3 score

## Products Web Design

You аre given the sliced Web design of the SoftUni Properties application (see SoftUniProperties.pdf) as HTML5 + CSS3 files. Pixel-perfect layout and responsive design are not required. You do not need to match exactly sizes / fonts / colors. Implement the site navigation and all screens using the provided site assets.

4 score

## Products Client-Side Web Application

Design and implement a client-side web app for managing the products with the following functionality:

* **Welcome screen** – when no user is logged in, the app should display the "Welcome" screen holding three buttons: **[Home]**, **[Login]** and **[Register]**.

3 score

* **Register user** – by username, password and confirm password the app should register a new user in the system. After a successful registration, a notification message 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.

5 score

* **Login user** – by username and password the app should be able to login an existing user. After a successful login, a notification message 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 login again.

5 score

* **User home screen** – after successful login, the app should display the user's home screen holding a welcome message + the username of the current user + navigation links **[Home]**, **[Estates]**, **[Add Estate]** and **[Logout]** (shown as menu on top).

Ensure you handle property all HTML special characters, e.g. the person name could be "*<peter><br>*".

5 score

* **Display all estates** – successfully logged users after clicking the "Estates" link at the menu should be able to view all estates. The estates should be listed as shown in the Web design. In case of error (e.g. Internet connection lost), an error message should be displayed.

*For maximum points use parse* ***queries*** *with* ***categories as pointers****. ("https://.../?include=category")*

15 score

* **Add new estate** – successfully logged in users should be able to add new estates by entering a name, category and price, and clicking the [Add] button. After successful estate creation, a notification message should be displayed and all estates should be shown. In case of error, an appropriate error message should be displayed and the user should be able to try to add an estate again.

*For maximum points use parse* ***categories as pointers****. {"category":{"\_\_type":"Pointer","className":"Category","objectId":categoryId}}*

10 score

* **Edit existing estate** – successfully logged in users should be able to edit their estates by choosing an estate, clicking **[Edit]**, editing the estate name and price, and clicking the **[Edit]** button. While editing an estate, you should **not** be **able** to **change** the **category**. At success, a notification message should be displayed and the estates page should be shown. In case of error, an appropriate error message should be displayed and all estates should be shown.

10 score

* **Delete existing estate** – successfully logged in users should be able to delete their own estates by choosing an estate, clicking **[Delete]**, and confirming the operation. At success, a notification message should be displayed and all products should be shown. In case of error, an appropriate error message should be displayed and all products should be shown.

10 score

* **Logout** – successfully logged in user should be able to logout from the app. After a successful logout, a notification message should be displayed and the login screen should be shown. All local information about the user should be deleted.

5 score

* **Notifications** – the application should notify the users about the result of their actions. In case of success an info notification message should be shown, which disappears automatically after 2 seconds or manually when the user clicks on it. In case of error, an error notification message should be shown which disappears automatically after 2 seconds or manually when the user clicks on it.

5 score

* **Estate ownership** – Users should be able to view all estates, but only the user who created them should be able to edit/delete them. **[Edit]** and **[Delete]** buttons should appear only if the estate belongs to the current user.

5 score

* **Authorization checks** – anonymous site visitors (without login) should be able to see the welcome, login, and register screens. All other screens should be accessible only after login. An attempt for anonymous access to these screens should redirect the user to the welcome screen.

5 score

* **Well-structured code** – high-quality JavaScript code and coding practices, use of template engines, routing libraries, promises, functionality slit into modules, etc.

10 score

* **\*Bonus: Category pointer** – implement the estates categories with **pointers** creating a **category** class in parse.com.

10 score

* **\*Bonus: filters** – successfully logged in users should be able to filter the estates in the estates screen by keyword, price range and category. The **[Filter]** button should remove all products which do not match the filters. **[Clear filters]** should restore the original products. The filtering may be performed **client-side**.

*For maximum points use parse* ***queries****. where={"price":{"$gte":minPrice,"$lte":maxPrice}, category":{"\_\_type":"Pointer","className":"Category","objectId":categoryId}}*

20 score