# Office Notes HTML5 Application – JS Apps Exam

You are assigned to design and implement an **office notes management Web front-end application** using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend. The app keeps users and their tasks written on notes. Users can register, login, logout, view today's notes of the whole office, add, view, edit and delete their own notes. 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 Office Notes REST Services

Register at Parse.com and create an application to keep your data in the cloud. Create a class **Note(title, text, author, deadline)** to hold notes entries. 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*", "fullName":"Pesho"}
  + 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*"}
* **User Logout**
  + Endpoint: <https://api.parse.com/1/logout>, Method: POST
  + Returns (JSON): {…, "sessionToken":"*session\_token*"}
* **List Notes with a deadline today**
  + Endpoint: [https://api.parse.com/1/classes/Note?where={"deadline":"28/04/2015"](https://api.parse.com/1/classes/Note?where=%7b%22deadline%22:%2228/04/2015%22)}, Method: GET
  + Returns (JSON): {"results":[{"author":"Kaloyan","deadline":"28/04/2015",…},…]}
* **List Your Notes**
  + Endpoint: [https://api.parse.com/1/classes/Note?where={"author":"Pesho"](https://api.parse.com/1/classes/Note?where=%7b%22author%22:%22Pesho%22)}, Method: GET
  + Returns (JSON): {"results":[{"author":"Pesho","deadline":"30/04/2015",…},…]}
* **Add Note**
  + Endpoint: <https://api.parse.com/1/classes/Note>, Method: POST
  + Request body (JSON): {"title":"*Tests*", "text":"*Make tests*", "author":"Pesho", "deadline":"01/05/2015", ACL":{…}}
  + Returns (JSON): {"createdAt":"…", "objectId":"…"}
* **Edit Note**
  + Endpoint: [https://api.parse.com/1/classes/Note/*note\_objectId*](https://api.parse.com/1/classes/Note/note_objectId), Method: PUT
  + Request body (JSON): {"title":"*Tests*", "text":"*Make tests*", "author":"Pesho", "deadline":"01/05/2015"}
  + Returns (JSON): {"updatedAt":"…", "objectId":"…"}
* **Delete Note**
  + Endpoint: [https://api.parse.com/1/classes/Note/*note\_objectId*](https://api.parse.com/1/classes/Note/note_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 created the object:
  + "ACL":{  
    "*user\_Id*":{"write":true,"read":true}, "\*":{"write":false,"read":true }}
  + Thus the object will only be accessible by the specified user but visible for all others

3 score

## Office notes Client-Side Web Application

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

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

2 score

* **Register user** – by username, password and full name 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.

10 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.

10 score

* **User home screen** – after successful login, the app should display the user's home screen holding a welcome message + the full name and username of the current user + navigation links (shown as menu on the left).

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

5 score

* **Display office notes** – after clicking the "Office notes" link at the menu, successfully logged users should be able to view all people's notes with a deadline – today's date. The notes should be listed as shown in the Web design. In case of error (e.g. Internet connection lost), an error message should be displayed.

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

15 score

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

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

15 score

* **Add new note** – successfully logged in users should be able to add new notes to their notes page by entering a title, text, author (the current user's full name) and deadline and afterwards clicking the [Add] button. After successful note creation, a notification message should be displayed and the "My notes" page should be shown. In case of error, an appropriate error message should be displayed and the user should be able to try to add a note again.

10 score

* **Edit existing note** – successfully logged in users should be able to edit their notes by choosing a note, clicking [Edit], editing the note's title, text and deadline and clicking the [Edit] button. At success, a notification message should be displayed and the "My notes" page should be shown. In case of error, an appropriate error message should be displayed and the "My notes" page should be shown.

10 score

* **Delete existing note** – successfully logged in users should be able to delete their notes by choosing a note, clicking [Delete], and confirming the operation. At success, a notification message should be displayed and the "My notes" page should be shown. In case of error, an appropriate error message should be displayed and the "My notes" page 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 welcome screen should be shown.

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 when the user clicks its [x] button or automatically after 4 seconds.

5 score

* **Notes ownership** – each registered user should have his own notes. A user should view / edit / delete his notes but should not edit / delete any other user's notes. However the user can view all other user's notes.

5 score

* **Authorization checks** – anonymous site visitors (without login) should be able to see the welcome, login and register screen. 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

* **\*Bonus: implement pagination** – when showing the "Office notes" or "My notes" page, you should implement pagination that shows only 10 notes per page. *Read the pagination documentation that was given to you in the exam archive for more information. You are allowed to use any pagination library, not only the given one.*

20 score

* **\*Bonus: well-structured code** – high-quality JavaScript code and coding practices, use of template engines, routing libraries, promises, functionality split into modules, etc.

10 score