# Social Network HTML5 Application – JS Apps Exam

You are assigned to design and implement a **social network Web front-end application** using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend. Users can register, login, view all users' posts, submit their own posts, edit their own profile 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.

**Note:** You are **NOT** allowed to use fullstack frameworks such as AngularJS and KnockoutJS.

## Social Network REST Services

Register at Parse.com and create an application to keep your data in the cloud. Create classes **User(username, password, name, about, gender, picture)** and **Post(createdBy, content)**. 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":"*username*",  "password":"*password*",  "name":"*fullName*",  "about":"*aboutInfo*",  "gender":"*gender*",  "picture":"*imageDataUrl*"  } |

* + 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*"}
* **Add Post**
  + Endpoint: <https://api.parse.com/1/classes/Post>, Method: POST
  + Request body (JSON):

|  |
| --- |
| {  "content":"*postContent*",  "createdBy": {  "\_\_type":"Pointer",  "className":"\_User",  "objectId":"*userObjectId*"  }  } |

* + Returns (JSON): {"createdAt":"…", "objectId":"…"}
* **List All Posts**
  + Endpoint: <https://api.parse.com/1/classes/Post>, Method: GET
  + Returns (JSON): {"results":[{"createdBy":"…","content":"…",…}, …]}

**Note:** Add **include=createdBy** to the service url as parameter to get the **user** along with the post.

* **Edit User Profile**
  + Endpoint: [https://api.parse.com/1/users/*user\_ObjectId*](https://api.parse.com/1/users/user_ObjectId), Method: PUT
  + Request body (JSON):

|  |
| --- |
| {  "password":"*password*",  "name":"*name*",  "about":"*about*",  "gender":"*gender*",  "picture":"*pictureDataUrl*"  } |

* + Returns (JSON): {"updatedAt":"…", …}
* **Get User**
  + Endpoint: : [https://api.parse.com/1/users/*user\_ObjectId*](https://api.parse.com/1/users/user_ObjectId)**,** Method: GET
  + Returns (JSON): {"about":"about","createdAt":"date","gender":"gender",…}

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*

3 score

## Social Network Web Design

You аre given the sliced Web design of the social network application (see <Social-Network-Screens.pptx>) as HTML5 + CSS3 files. Implement the site navigation and all screens using the provided site assets. Pixel-perfect layout and responsive design are not required.

4 score

## Social Network Client-Side Web Application

Design and implement a client-side web app for the social network 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].

3 score

* **Register user** – by username, password, full name, about information, gender and picture the app should register a new user in the system. The user should be able to upload only pictures up to 128KB. The uploaded picture should appear in the picture box as preview. 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, username of the current user, profile picture + navigation links (as shown on the provided screens).

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

5 score

* **Display posts** – successfully logged users should be able to view all posts. The posts 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 user's name could be "*<peter>*".

15 score

* **Add new post** – successfully logged in users should be able to submit new posts. The post button on the navigation bar should **slide down** a text box. Clicking the [Post] button again should **slide the text box up**. After successful post submission, a notification message should be displayed, the text box should be closed and the new post should be shown. In case of error, an appropriate error message should be displayed.

15 score

* **Edit user profile** – successfully logged in users should be able to edit their profiles (change their password, about info, name, gender and profile picture). At success show info notification and the user home screen. At error, show an error message and ask the user to try editing his profile again.

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 guest 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 5 seconds.

5 score

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

* **Post hover** – Hovering over the name of a post's author should display a **tooltip** holding additional information about the user. If the data has not yet been received, the tooltip should display "Loading…". Moving the cursor away should **hide** the tooltip. (See the provided screens for more details.)

10 score

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

20 score