# Phonebook HTML5 Application – JS Apps Exam

You are assigned to design and implement a **phonebook management Web front-end application** using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend. The app keeps users and their phonebooks. Users can register, login, view their phonebooks, add, edit and delete phones 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 Phonebook REST Services

Register at Parse.com and create an application to keep your data in the cloud. Create a class **Phone(person, number)** to hold phonebook 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":"*name*"}
  + 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 Phones**
  + Endpoint: <https://api.parse.com/1/classes/Phone>, Method: GET
  + Returns (JSON): {"results":[{"person":"…","number":"…",…}, …]}
* **Add Phone**
  + Endpoint: <https://api.parse.com/1/classes/Phone>, Method: POST
  + Request body (JSON): {"person":"*person*", "number":"*number*", "ACL":{…}}
  + Returns (JSON): {"createdAt":"…", "objectId":"…"}
* **Edit Phone**
  + Endpoint: [https://api.parse.com/1/classes/Phone/*phone\_objectId*](https://api.parse.com/1/classes/Phone/phone_objectId), Method: PUT
  + Request body (JSON): {"person":"*person*", "number":"*number*"}
  + Returns (JSON): {"updatedAt":"…", "objectId":"…"}
* **Delete Phone**
  + Endpoint: [https://api.parse.com/1/classes/Phone/*phone\_objectId*](https://api.parse.com/1/classes/Phone/phone_objectId), Method: DELETE
  + Returns (JSON): { }
* **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): {"username":"*user*", "password":"*pass*", "fullName":"*name*"}
  + Returns (JSON): {"updatedAt":"…", "username":"…", "fullName":"…"}

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\_оbjectId*":{"write":true,"read":true}}
  + Thus the object will only be visible and accessible by the specified user and invisible for all others

3 score

## Phonebook Web Design

You аre given the sliced Web design of the Phonebook application (see <Phonebook.pptx>) 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

## Phonebook Client-Side Web Application

Design and implement a client-side web app for managing the phonebook 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 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 person name could be "*<peter>*".

5 score

* **Display user's phonebook** – successfully logged users after clicking the "Phonebook" link at the menu should be able to view all phones created by the current user. The phones should be listed as table 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 phonebook person could be "*<peter>*".

15 score

* **Add new phone** – successfully logged in users should be able to add new phones to their phonebook by entering a person name and phone number and clicking the [Add] button. After successful phonebook creation, a notification message should be displayed and the phonebook 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 phone again.

10 score

* **Edit existing phone** – successfully logged in users should be able to edit their phones by choosing a phone, clicking [Edit], editing the person name and phone number and clicking the [Edit] button. At success, a notification message should be displayed and the phonebook should be shown. In case of error, an appropriate error message should be displayed and the phonebook should be shown.

10 score

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

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

* **Phonebook ownership** – each registered user should have his own phonebook. A user should view / edit / delete his phones but should not view and access any other user's phones.

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

* **\*Bonus: edit user profile** – successfully logged in users should be able to edit their profiles (change their username, password and full name). At success show info notification and the user home screen. At error, show an error message and leave the user try editing his profile again.

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.

10 score