# SoftUni Calendar HTML5 Application – JS Apps Exam

You are assigned to design and implement an **events calendar management Web front-end application** using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend. The app keeps lectures and their lectors saved in a calendar. Users can register, login, logout, view all lectures, add, view, edit and delete their own lectures. The app should be implemented as client-side Web application in JavaScript with server-side REST services called by AJAX and returning JSON objects. Front-end JavaScript **frameworks** like AngularJS are **forbidden**!

## Create SoftUni Calendar REST Services

Register at kinvey.com and create an application to keep your data in the cloud. Create a collection **lectures** to hold **lecture** entries**(title, start, end, lecturer)**. kinvey.com will automatically create REST services to access your data:

* **User Registration (Sign Up)**
  + Endpoint: **https://baas.kinvey.com/user/:appid**, Method: POST
  + Authorization header: **[app credentials]**
  + Content-Type header: **application/json**
  + Request body (JSON):

|  |
| --- |
| {  "username":"*username*",  "password":"*password*"  } |

* + Returns (JSON):   
    {"username":"…","\_id":"…","\_kmd":{…,"authtoken":"*authTokenValue*"}}
* **User Login**
  + Endpoint: **https://baas.kinvey.com/user/:appid/login**, Method: POST
  + Authorization header: **[app credentials]**
  + Content-Type header: **application/json**
  + Request body (JSON):

|  |
| --- |
| {  "username":"*username*",  "password":"*password*"  } |

* + Returns (JSON):   
    {"username":"…","\_id":"…","\_kmd":{…,"authtoken":"*authTokenValue*"}}
* **User Logout**
  + Endpoint: **https://baas.kinvey.com/user/:appid/\_logout**, Method: POST
  + Authorization header: **[user credentials/session authorization]**
* **List All Lectures**
  + Endpoint: **https://baas.kinvey.com/appdata/:appid/lectures/**, Method: GET
  + Authorization header: **[user credentials/session authorization]**
  + Returns (JSON):   
    [{"\_id":"…","title":"…","start":"…","end":"…","\_acl":{"creator":"…"}}, …]
* **List Your Lectures**
  + Endpoint: **https://baas.kinvey.com/appdata/:appid/lectures/?query={"\_acl.creator":":userId"}**, Method: GET
  + Authorization header: **[user credentials/session authorization]**
  + Returns (JSON):   
    [{"\_id":"…","title":"…","start":"…","end":"…","\_acl":{"creator":"…"}}, …]
* **Add Lecture**
  + Endpoint: **https://baas.kinvey.com/appdata/:appid/lectures/**, Method: POST
  + Authorization header: **[user credentials/session authorization]**
  + Request body (JSON):

|  |
| --- |
| {  "title":"*lectureTitle*",  "start":"*lectureStartDateTime",*  "end":"*lectureEndDateTime*",  "lecturer":"*username*"  } |

* + Returns (JSON): {"\_acl":{"creator":"*userId*"},"\_id":"…","title":"…","start":"…",…}
* **Edit Lecture**
  + Endpoint: **https://baas.kinvey.com/appdata/:appid/lectures/:lectureId**, Method: POST
  + Authorization header: **[user credentials/session authorization]**
  + Request body (JSON):

|  |
| --- |
| {  "title":"*lectureTitle*",  "start":"*lectureStartDateTime",*  "end":"*lectureEndDateTime*",  "lecturer":"*username*"  } |

* + Returns (JSON): {"\_acl":{"creator":"*userId*"},"\_id":"…","title":"…","start":"…",…}
* **Delete Lecture**
  + Endpoint: **https://baas.kinvey.com/appdata/:appid/lectures/:lectureId**, Method: POST
  + Authorization header: **[user credentials/session authorization]**

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

* Authorization: *Basic [base64 string]*
  + App credentials authorization
  + The base64 string is encoded from *appId:appSecret* concatenated string
* Authorization: *Kinvey [base64 string]*
  + User session authorization
  + The bas64 string is given as a response after register/login

Notes about **users and authentication**:

* After register / login, pass the session token as HTTP request header to **authenticate your requests**:
  + Authorization: Kinvey *session\_token\_returned\_by\_login\_or\_register*

## SoftUni Calendar 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].
  + **url**: "#/"
  + **template**: "welcome-guest.html"

2 score

* **Register user** – by username, password and confirm password the app should register a new user in the system. After a confirming (repeating) the password, you should send a registration request and a notification message should be displayed. After register, 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.
  + **url**: "#/register/"
  + **template**: "register.html"

6 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.
  + **url**: "#/login/"
  + **template**: "login.html"

6 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 (shown as menu on the top).

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

* + **url**: "#/"
  + **template**: "welcome-user.html"

2 score

* **Display all lectures** – after clicking the "Calendar" link at the menu, successfully logged users should be able to view all people's lectures. The lectures should be listed as shown in the Web design. You are required to use a library called fullcalendar.js. More information about how to use it can be found in the additional file in the assignment archive. In case of an 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>*".

* + **url**: "#/calendar/list/"
  + **template**: "calendar.html" – **Do not modify the template!**
  + **template**: "modal.html" – **modify this template but do not add it to the html of the page**

14 score

* **Display user's lectures** – successfully logged users after clicking the "My Lectures" link at the menu should be able to view all lectures created by the current user. The lectures should be listed as shown in the Web design. You are required to use a library called fullcalendar.js. More information about how to use it can be found in the additional file in the assignment archive. 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>*".

* + **url**: "#/calendar/my/"
  + **template**: "calendar.html" – **Do not modify the template!**
  + **template**: "modal.html" – **modify this template but do not add it to the html of the page**

20 score

* **Add new lecture** – successfully logged in users should be able to add new lectures to the calendar by entering a **title**, **start** (**2016-03-20T09:00:00** format), **end** (**2016-03-20T09:00:00** format) and **lecturer**. Afterwards clicking the [Add] button. After successful lecture creation, a notification message should be displayed and the "My Lectures" 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.
  + **url**: "#/calendar/add/"
  + **template**: "add-lecture.html"

6 score

* **Edit existing lecture** – successfully logged in users should be able to edit their lectures by choosing a lecture, clicking [Edit], editing the lecture’s title, start and end, and then clicking the [Edit] button. At success, a notification message should be displayed and the "My Lectures" page should be shown. In case of error, an appropriate error message should be displayed and the "My Lectures" page should be shown.
  + **url**: "#/calendar/edit/*:lectureId*"
  + **template**: "edit-lecture.html"

8 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.
  + **url**: "#/calendar/delete/*:lectureId*"
  + **template**: "delete-lecture.html"

6 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.
  + **url**: "#/logout/"

2 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.

4 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. The "menu-home.html" should be displayed if the user has been logged in. An attempt for anonymous access to these screens should redirect the user to the welcome screen. If the user is not logged, display the "menu-login.html".

4 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

* **Total** – 80 (90 with bonus) pts.