

JS Apps Exam – Airline-Administration Application

You are assigned to implement a **Web application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Kinvey). The **app** that keeps **users** (airline administrators) that manage **flights**. Users can **register**, **login**, **logout**, view a page with **all public** flights, **create** a flight, **edit** and **delete** their own flights, view a **detailed** page of a flight and view their **own** flights only.

You are **allowed** to use libraries like **jQuery**, **Handlebars** and **Sammy**. Frameworks and libraries like React, Angular, Vue are **not permitted**.

Problem 1. Create a Kinvey REST Service

Register at **Kinvey.com** and create an application to keep your data in the cloud.

Create a collection **flights**. Each flight has a **destination** airport, **origin** airport, **departure time**, number of **seats** in flight, **cost** per seat, flight **image**, and information whether the flight **is public** or not.

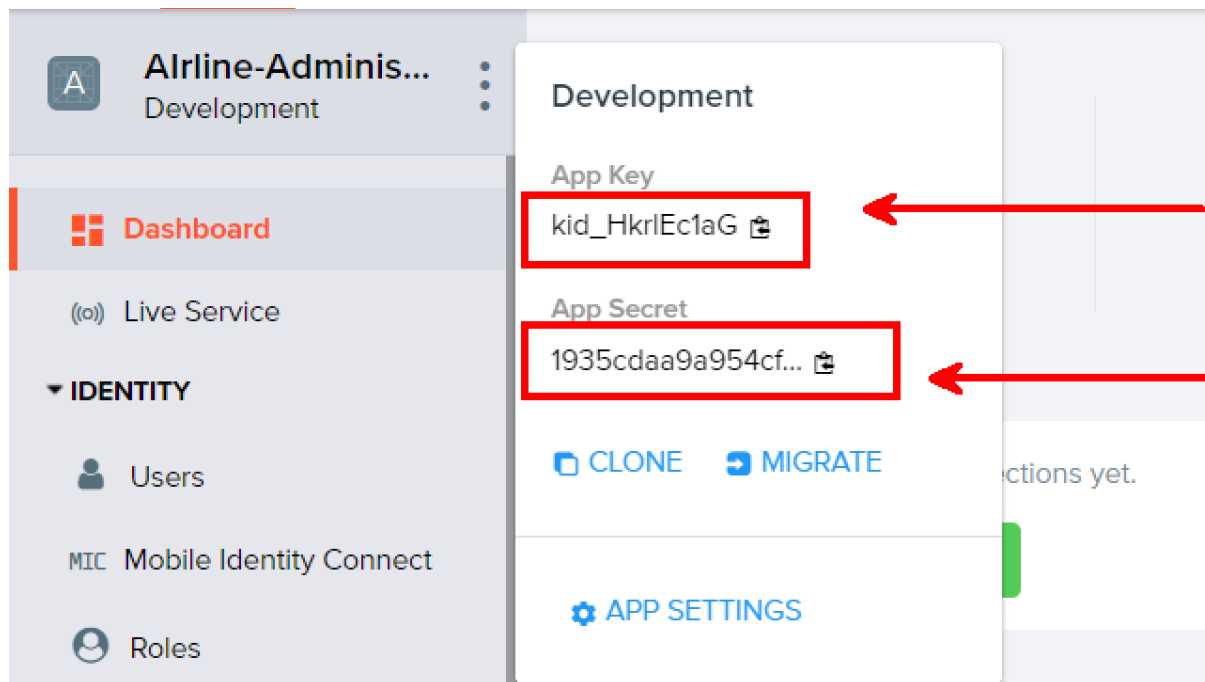
Problem 2. Test the Kinvey REST Services

Common Responses

Note: When creating or updating records, the response will contain the **entire record** body, as it appears in the database. It's advisable if you observe network traffic via Postman or using your browser's dev-tools, to view details about each request.

Response Code	Response Body
200 OK	<code><Record data></code>
201 Created	<code><Record data></code>
204 No Content	<code><Empty></code>
401 Unauthorized	<pre>{ "error": "InvalidCredentials", "description": "Invalid credentials. ...", "debug": "" }</pre>
404 Not Found	<pre>{ "error": "EntityNotFound", "description": "This entity not found in the collection", "debug": "" }</pre>
Error response 409 Conflict	<pre>{ "error": "UserAlreadyExists", "description": "This username is already taken. ...", "debug": "" }</pre>

Using **Postman** or other HTTP client tool (you can use Kinvey's built-in **API Console**), test the REST service endpoints:



User Registration (Sign Up)

POST https://baas.kinvey.com/user/ <i>app_key</i> /	
Request headers	Authorization: Basic base64(app_id:app_secret) Content-Type: application/json
Request body	{ "username": "testuser", "password": "testuserpass890" }

The request needs “**Basic**” authentication. Use the Kinvey **App Key** and Kinvey **App Secret** as credentials.

User Login

POST https://baas.kinvey.com/user/ <i>app_key</i> /login	
Request headers	Authorization: Basic base64(app_id:app_secret) Content-Type: application/json
Request body	{ "username": "testuser", "password": "testuserpass890" }

Successful login returns an “**authtoken**” which is later used to authenticate the CRUD operations.

User Logout

POST https://baas.kinvey.com/user/app_key/_logout

Request headers	Authorization: Kinvey authtoken
-----------------	--

To logout, you need to provide the “**authtoken**” given by login / register as “**Kinvey**” authorization header.

Get Published Flights

GET [https://baas.kinvey.com/appdata/app_key/flights?query={"isPublished":"true"}](https://baas.kinvey.com/appdata/app_key/flights?query={)

Request headers	Authorization: Kinvey authtoken
-----------------	---------------------------------

Create Flight

POST https://baas.kinvey.com/appdata/app_key/flights

Request headers	Authorization: Kinvey authtoken Content-Type: application/json
-----------------	---

Request body	<pre>{ "destination": "Las Vegas", "origin": "New York", "departure": "2017-02-02", "seats": 25, "cost": 15, "image": "http://air.....jpg ", "isPublished": true }</pre>
--------------	--

Edit Flight

PUT https://baas.kinvey.com/appdata/app_key/flights/flight_id

Request headers	Authorization: Kinvey authtoken Content-Type: application/json
-----------------	---

Request body	<pre>{ "destination": "Las Vegas", "origin": "New York", "departure": "2017-02-02", "seats": 25, "cost": 15, "image": "http://air.....jpg ", "isPublished": true }</pre>
--------------	--

Delete Flight

DELETE https://baas.kinvey.com/appdata/ app_key /flights/ flight_id	
Request headers	Authorization: Kinvey authtoken

Flight Details

GET https://baas.kinvey.com/appdata/ app_key /flights/ flight_id	
Request headers	Authorization: Kinvey authtoken

My Flights

GET https://baas.kinvey.com/appdata/ app_key /flights?query={"_acl.creator":" user_id "}	
Request headers	Authorization: Kinvey authtoken

Use the ID of the currently **logged in user**.

Problem 3. HTML and CSS

You are given the Web design of the application as **HTML + CSS** files.

- Initially all views and forms are shown by the HTML. Your application may **hide** by CSS (display: none) or **delete** from the DOM all unneeded elements or just display the views it needs to display.
- You may render the views / forms / components with **jQuery** or **Handlebars**.

Important: don't change the elements' **class name** and **id**. Don't rename form fields / link names / ids. You are **allowed** to add **data attributes** to any elements. You may modify **href attributes** of links and add **action/method attributes** to forms, to allow the use of a routing library.

Problem 4. Client-Side Web Application

Design and **implement** a client-side front-end app (SPA). Implement the functionality described below.

Notifications (10 pts)

The application should notify the users about the result of their actions.

- In case of successful action an **informational (green) notification message** should be shown, which disappears automatically after 3 seconds or manually when the user clicks it.

Logout successful.

- In case of **error**, an **error notification message (red)** should be shown which disappears on user click.

Error: Invalid credentials. Please retry your request with correct credentials

- During the AJAX calls a **loading notification message (blue)** should be shown. It should disappear automatically as soon as the AJAX call is completed.

Loading ...

Points for notifications are awarded separately for each section.

Navigation System (10 pts)

Implement a **navigation system** for the app: navigation links should correctly change the current screen (view).

- Clicking on the links in the **menu** or **individual** links should display the view behind the link (views are sections in the HTML code).
- The given „**Navigation**“ menu should be visible **only** for logged in users. Anonymous users can **only** view the **login/register** section and logged in users can view **flights** section.

Register User Screen (5 pts)

By given **username**, **password** and **repeat password** the app should register a new user in the system.

- After a **successful registration**, a notification message “User registration successful.” should be displayed and the user should be **redirected** to the home view.
- You **need** to validate the **input**. A username **should** be a string with at **least** 5 characters **long**. Passwords **input** fields shouldn’t be **empty**. Both passwords **should** match.
- In case of **error** (eg. invalid username/password), an appropriate error **message** should be displayed and the user should be able to **try** to register again.
- Keep the user session data in the browser’s **session storage**.
- Clear **all** input fields after **successful** register.

Create your account:

Username:

Password:

Repeat Password:

Login User Screen (5 pts)

By given **username** and **password** the app should be able to login an existing user.

- After a **successful login**, a notification message “Login successful.” should be displayed and the user should be **redirected** to the home view.
- In case of **error**, an appropriate error message should be displayed and the user should be able to fill the login form again.
- Form validation** should be the **same** as register.
- Keep the user session data in the browser’s **session storage**.
- Clear **all** input fields after **successful** login.

Login to your account:

Username:

Password:

Logout (5 pts)

Successfully logged in user should be able to **logout** from the app.

- After a **successful** logout, a **notification** message “Logout successful.” should be displayed.
- After successful logout, the **Sign In screen** should be shown.
- The “**logout**” **REST service** at the back-end should be obligatory called at logout.
- All local information in the browser (**user session data**) about the current user should be deleted.

Home Screen (List all published flights) (20 points)

Whenever the user opens the home screen a list of all published flights (public flights by all users) should be shown in the following format:



Add Flight (10 points)

Clicking on **[Add Flight+]** button should **redirect** to form where the admin creates a flight.

Add new flight

Destination:

Destination

Origin:

Origin

Departure Date:

mm / dd / yyyy

Departure Time:

--:-- --

Number of Seats:

0

Cost per seat:

0.00

Image:

Is Public:

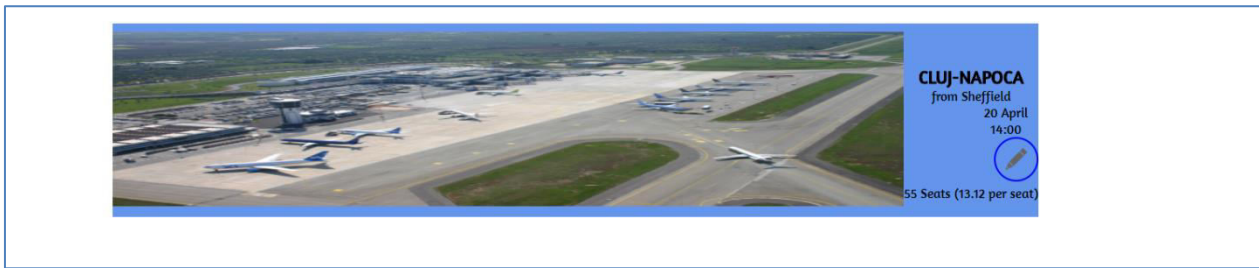
☐

Create

Each flight has a **destination** station, **origin** station (both strings), a **departure** time, number of **seats** per flight, **cost** per seat (both should be validated that they are valid numbers) an image **url** and information whether the flight is **public** or not. Destination and origin station should be **non-empty** strings. Number of seats and cost per seat should be **positive** numbers. (after successful creation of a flight redirect to the home screen and display a message “Created flight.”)

Flight Details (10 points)

Clicking on each **individual** flight on the home screen or My Flights, **redirects** to a flight details page where **additional** information for each flight is shown (departure hour, departure minutes, number of seats, cost per seat). If the user is the creator of a flight, display the option to edit it (pencil icon).



Edit flight (10 points)

If the user is the **creator** of a specific flight he should be able to **edit** it. Clicking on the edit button **redirects** to a form where the user can **modify** the given flight (all validations in **create a flight** should be followed).

Edit existing flight

Destination:

CLUJ-NAPOCA

Origin:

Sheffield

Departure Date:

04 / 28 / 2018

Departure Time:

02:00 PM

Number of Seats:

55

Cost per seat:

13.12

Image:

http://airportcluj.ro/files/ro/userfiles

Is Public:

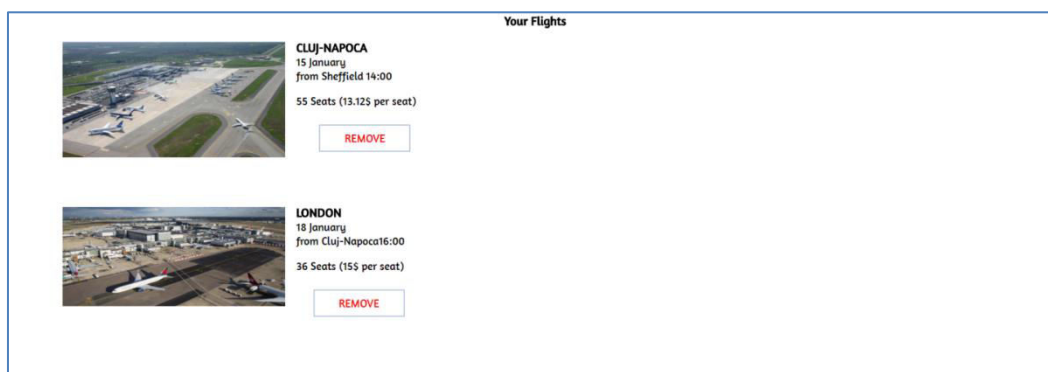
☒

Save Changes

After **successfully** editing a flight the user should be **redirected** to the **details** page of the flight a message "Successfully edited flight." should be shown.

My Flights (10 points)

All **authenticated** users can view their **own** flights by clicking on the **[Flights]** button in the navigation.



Delete Flight (5 points)

In the **My Flights** section users can **delete** their **own** flights. Deleting is done **instantly**. When the user **successfully** deletes a flight the message "Flight deleted." should be shown and the user should be **redirected** to the **same** page.

Problem 5. Submitting Your Solution

Place in a ZIP file your project folder. Exclude the **node_modules** folder. Upload the archive to the Judge.