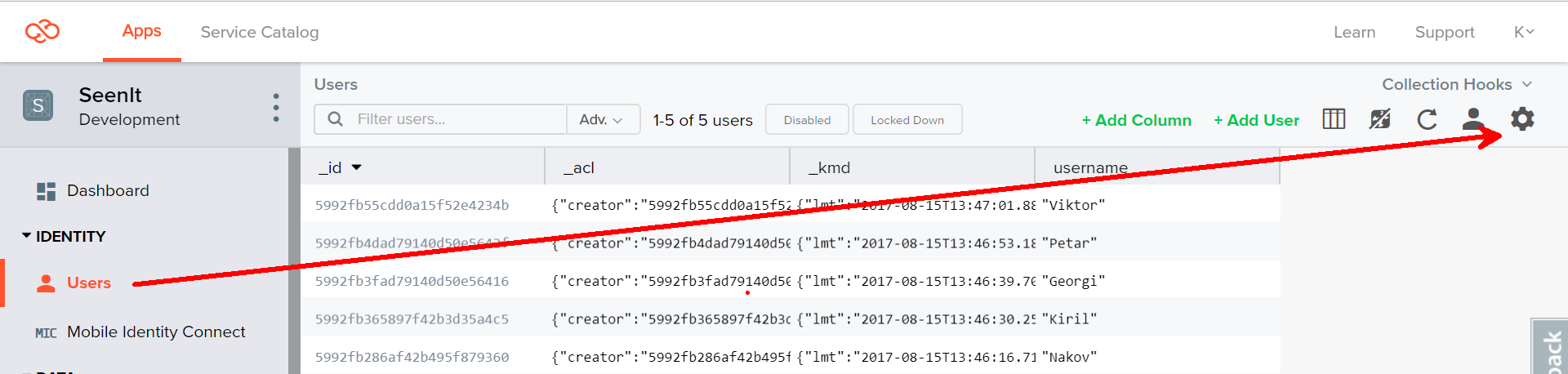
# Exercise: React SeenIt

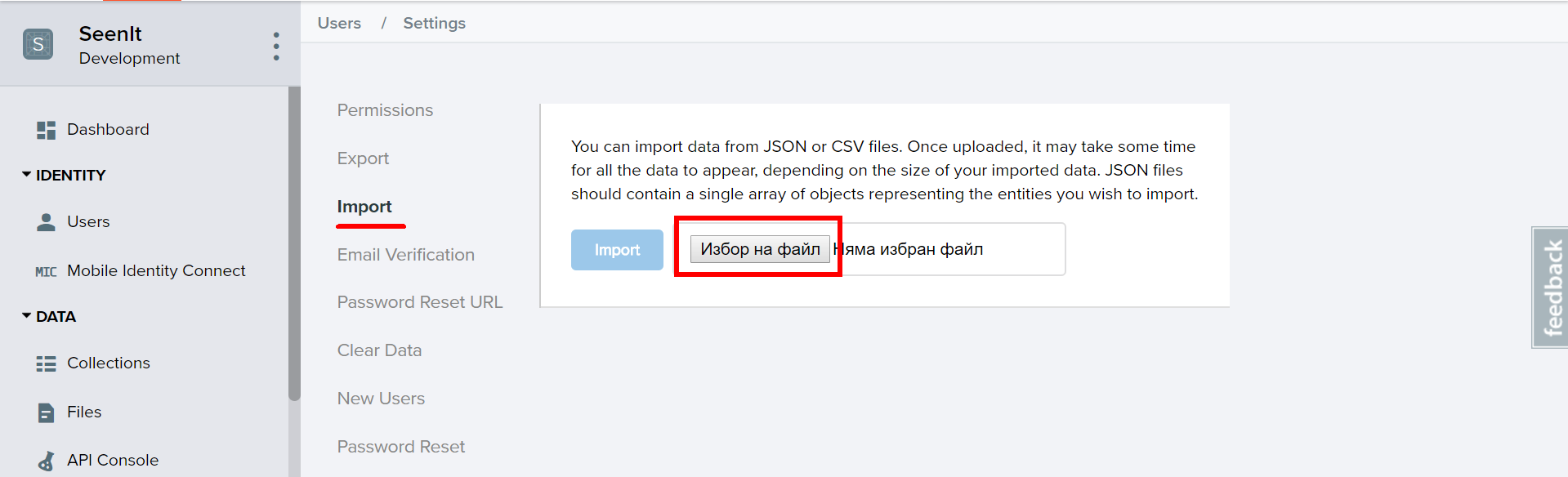
For this exercise, you must implement the application described below, using React and React-Router. Follow the steps, outlined as best practices in the lecture:

* Start with a mock of the application – you are provided with HTML, the API mock is based on Kinvey, which takes very little to set up
* Break the UI into a component hierarchy
* Build a static version in React
* Identify the minimal representation of UI state
* Identify where your state should live
* Add inverse data flow

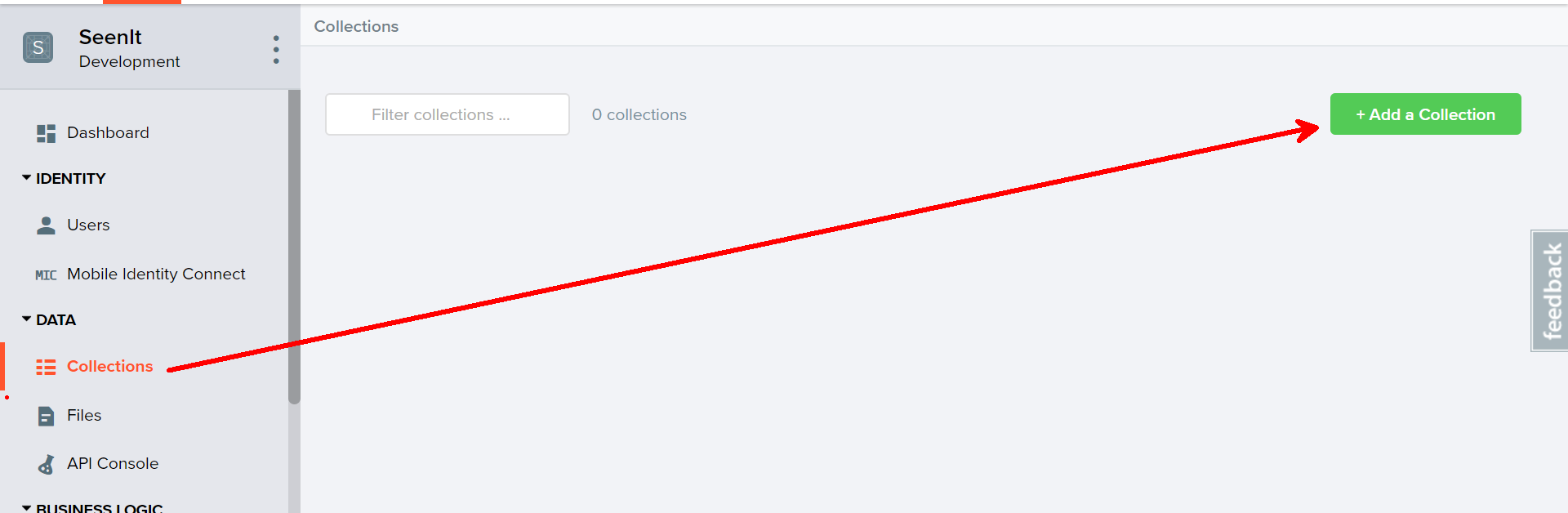
## Create a Kinvey REST Service

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

In the **Users** collection, import the provided JSON file with sample users to get started with template data. In the **Kinvey Console**, select **Users** from the navigation of the left, click **Settings** in the upper right then scroll down to the **Import** section: 



Create a collection **posts**. Each **post** has a **url** (a link to website), **title**, **imageUrl**, **description** and an **author** (username of the author).



Create a collection **comments**. Each comment has **postId**, **content** and **author** (username of the author).

You are also given **two** more json files to **import** data for **posts** and **comments**.

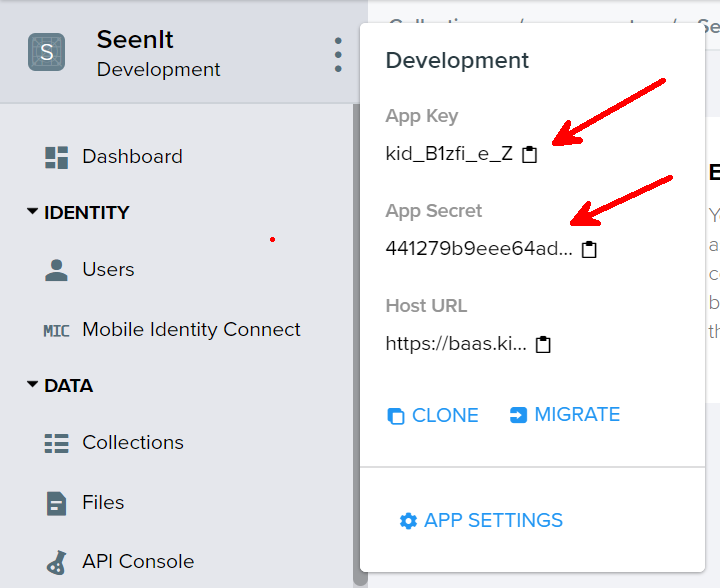
## Test the Kinvey REST Services

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/***app\_id***/](https://baas.kinvey.com/user/app_id/) | |
| Request headers | Authorization: Basic base64(app\_id:app\_secret)  Content-Type: application/json |
| Request body | {  "username": "testuser",  "password": "testuserpass890"  } |
| Response  201 Created | {  "\_id": "59930c78a743e20c7d3fca77",  "username": "testuser",  "password": "testuserpass890"  } |
| Error response  409 Conflict | { "error": "UserAlreadyExists", "description": "This username is already taken. Please retry your request with a different username", "debug": "" } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

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



### User Login

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_id***/login](https://baas.kinvey.com/user/app_id/login) | |
| Request headers | Authorization: Basic base64(app\_id:app\_secret)  Content-Type: application/json |
| Request body | {  "username": "testuser",  "password": "testuserpass890"  } |
| Response  200 OK | {  "\_id": "59930c78a743e20c7d3fca77",  "username": "testuser"  "\_kmd": {  "**authtoken**": "**8e6471bc-3712-4cfb-b92e-50e62a0c80….Duj5fHdM /7XHIe6KdY=**"  …  },  …  } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

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

### User Logout

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_id***/\_logout](https://baas.kinvey.com/user/app_id/_logout) | |
| Request headers | Authorization: Kinvey **authtoken** |
| Response  204 No Content |  |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

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

### List all Posts (Catalog – sorted by post time, descending)

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/***app\_id***/posts](https://baas.kinvey.com/appdata/app_id/posts)?query={}&sort={"\_kmd.ect": -1} | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "599307d36af42b495f87bad2",  "author": "Kiril",  "description": "My GitHub profile",  "url": "https://github.com/k1r1L",  "title": "GitHub",  "imageUrl": "https://pbs.twimg.com/profile\_images/616309728688238592/pBeeJQDQ.png",  "\_acl": {  "creator": "kid\_B1zfi\_e\_Z"  },  "\_kmd": {  "lmt": "2017-08-15T14:40:19.182Z",  "ect": "2017-08-15T14:40:19.182Z"  }  },...  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### Create Post

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/appdata/***app\_id***/posts](https://baas.kinvey.com/appdata/app_id/posts) | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "author": "Kiril",  "title": "Test Post",  "description": "Test Post Description",  "url": "https://en.wikipedia.org/wiki/Santorini",  "imageUrl":"http://sailing-santorini.com/wp-content/uploads/2015/10/sunset-oia 500x500.jpg"  } |
| Response  201 Created | {  "\_id": "59931398996ab5127d2a84d1",  "author": "Kiril",  "title": "Test Post",  "description": "Test Post Description",  "url": "https://en.wikipedia.org/wiki/Santorini",…  } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### Edit Post

|  |  |
| --- | --- |
| **PUT** [https://baas.kinvey.com/appdata/***app\_id***/posts/**post\_id**](https://baas.kinvey.com/appdata/app_id/posts/post_id) | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "author": "Kiril",  "title": "Test Post **Edited**",  "description": "Test Post Description **Edited**",  "url": "https://en.wikipedia.org/wiki/Santorini",  "imageUrl":"http://sailing-santorini.com/wp-content/uploads/2015/10/sunset-oia 500x500.jpg"  } |
| Response  200 Ok | {  "\_id": "59931398996ab5127d2a84d1",  "author": "Kiril",  "title": "Test Post **Edited**",  "description": "Test Post Description **Edited**",  "url": "https://en.wikipedia.org/wiki/Santorini",…  } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### Delete Post

|  |  |
| --- | --- |
| **DELETE** [https://baas.kinvey.com/appdata/***app\_id***/posts/***post\_id***](https://baas.kinvey.com/appdata/app_id/posts/post_id) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | {  "count": 1  } |
| Error response  404 Not Found | { "error": "EntityNotFound", "description": "This entity not found in the collection", "debug": "" } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### My Posts (View all posts by logged in user, sorted by post time, descending)

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/**app\_id**/posts?query={"author":"**username**"}](https://baas.kinvey.com/appdata/app_id/posts?query=%7b%22author%22:%22username%22%7d)&sort={"\_kmd.ect": -1} | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "599307d36af42b495f87bad2",  "author": "Kiril",  "description": "My GitHub profile",  "url": "https://github.com/k1r1L",  "title": "GitHub",  "imageUrl": "https://pbs.twimg.com/profile\_images/616309728688238592/pBeeJQDQ.png",  "\_acl": {  "creator": "kid\_B1zfi\_e\_Z"  },  "\_kmd": {  "lmt": "2017-08-15T14:40:19.182Z",  "ect": "2017-08-15T14:40:19.182Z"  }  },  {  "\_id": "5993178dbce993cf532bbe29",  "author": "Kiril",  "title": "Test Post",  "description": "Test Post Description",  "url": "https://en.wikipedia.org/wiki/Santorini",  "imageUrl": "http://sailing-santorini.com/wp-content/uploads/2015/10/sunset-oia-500x500.jpg",  "\_acl": {  "creator": "59930c78a743e20c7d3fca77"  },  "\_kmd": {  "lmt": "2017-08-15T15:47:25.376Z",  "ect": "2017-08-15T15:47:25.376Z"  }  }  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### Post Details (View Post Comments)

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/**app\_id**/posts/**post\_id**](https://baas.kinvey.com/appdata/app_id/posts) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "599307d36af42b495f87bad2",  "author": "Kiril",  "description": "My GitHub profile",  "url": "https://github.com/k1r1L",  "title": "GitHub",  "imageUrl": "https://pbs.twimg.com/profile\_images/616309728688238592/pBeeJQDQ.png",  "\_acl": {  "creator": "kid\_B1zfi\_e\_Z"  },  "\_kmd": {  "lmt": "2017-08-15T14:40:19.182Z",  "ect": "2017-08-15T14:40:19.182Z"  }  }  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

**Note: Keep in mind that the quotes used in the query must be exactly double quotes(").**

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/**app\_id**/comments?query={"postId":"**post\_id**"}](https://baas.kinvey.com/appdata/app_id/posts)&sort={"\_kmd.ect": -1} | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "599309bdbce993cf532b9643",  "postId": "599307d36af42b495f87bad2",  "author": "Nakov",  "content": "Very cool profile",  "\_acl": {  "creator": "kid\_B1zfi\_e\_Z"  },  "\_kmd": {  "lmt": "2017-08-15T15:50:59.486Z",  "ect": "2017-08-15T14:48:29.551Z"  }  },  {  "\_id": "59930a34133f6d253dfe31c9",  "postId": "599307d36af42b495f87bad2",  "content": "Nice!",  "author": "Viktor",  "\_acl": {  "creator": "kid\_B1zfi\_e\_Z"  },  "\_kmd": {  "lmt": "2017-08-15T14:52:46.189Z",  "ect": "2017-08-15T14:50:28.536Z"  }  },..  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### Create Comment

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/appdata/**app\_id**/comments](https://baas.kinvey.com/appdata/app_id/comments) | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "postId": "5993178dbce993cf532bbe29",  "content": "Very nice post",  "author": "Kiril"  } |
| Response  201 Created | {  "postId": "5993178dbce993cf532bbe29",  "content": "Very nice post",  "author": "Kiril",  "\_acl": {  "creator": "59930c78a743e20c7d3fca77"  },...  } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### Delete Comment

|  |  |
| --- | --- |
| **DELETE** [https://baas.kinvey.com/appdata/**app\_id**/comments/**comment\_id**](https://baas.kinvey.com/appdata/app_id/comments/comment_id) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | {  "count": 1  } |
| Error response  404 Not Found | { "error": "EntityNotFound", "description": "This entity not found in the collection", "debug": "" } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

## SeentIt–HTML and CSS

You аre given the Web design of the Messages 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 **Mustache** **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.

## SeenIt Client-Side Web Application

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

### Notifications

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.



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



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



### Navigation System (5 pts)

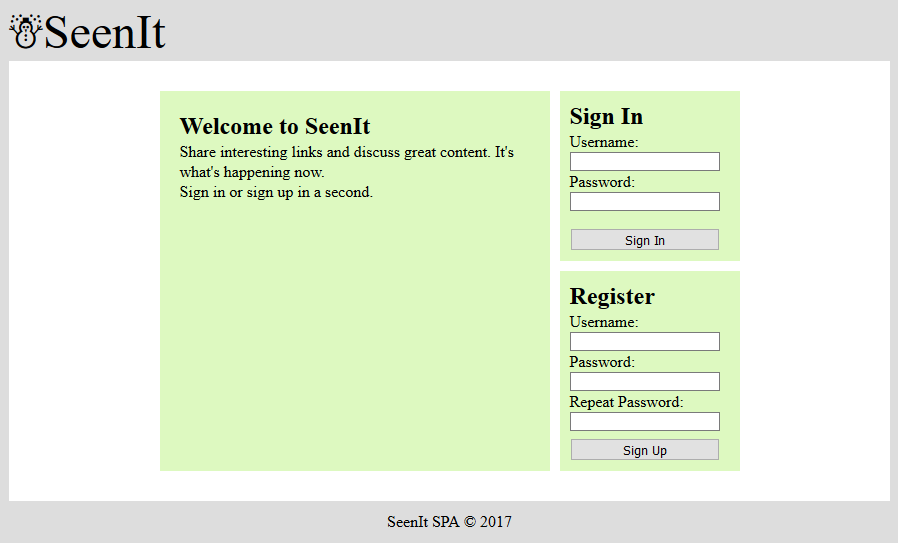
Implement a **navigation system** for the app: navigation links should correctly change the current screen (view).

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

### Register User (10 pts)

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

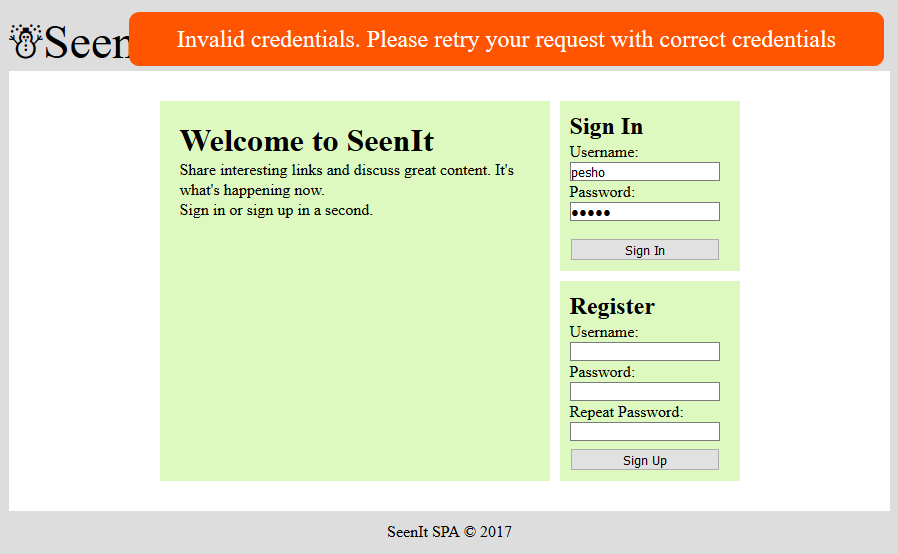
* After a **successful registration**, a notification message “User registration successful.” should be displayed and the **catalog** with all **posts** should be shown.
* You **need** to validate the **input**. A username **should** be at **least** 3 characters **long** and should contain only english alphabet **letters**. A user‘s password should be at **least** 6 characters **long** and should contain only english alphabet **letters** and **digits**. 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** login.



### Login User (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 home screen should be displayed.
* 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.



### Logout (5 pts)

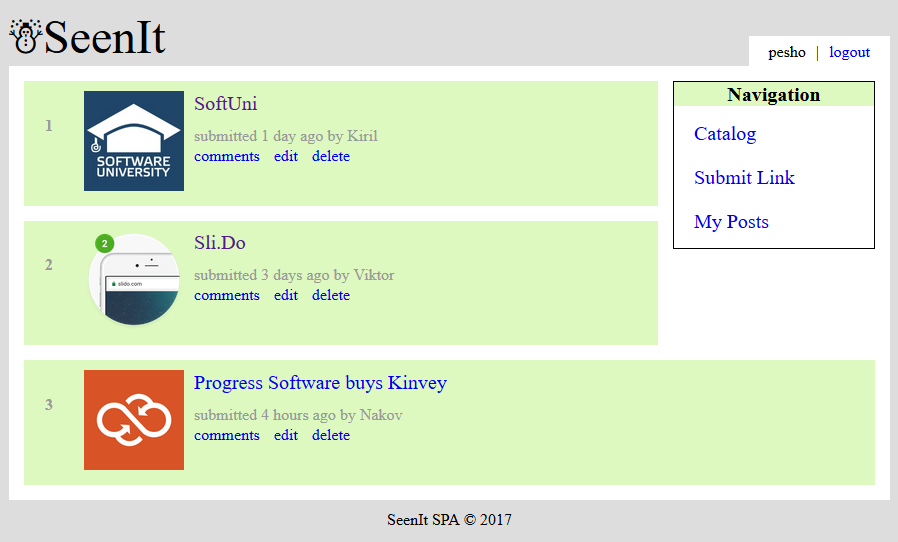
Successfully logged in user should be able to **logout** from the app.

* 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.

### Catalog Screen (20 pts)

Successfully logged users should be welcomed by the **catalog** screen which can be accessed **after** clicking the **[Catalog]** link in the **sidebar**. The catalog contains **all** posts on the site.

* The **posts** should be listed in the **format** as shown in the Web design (see the screenshot below).
* Each **post** has **title**, **url**, **imageUrl** (optional), **description** (optional), **rank** (order number) and information about **when** it was **created**. For the **time created** information you will use a helper function given **below**. It requires a **parameter** which is the entity creation date of the post (or **\_kmd.ect** property of the post object).
* When the **comments** link is clicked, your app should **redirect** to the details **section** of the post with comments.
* Post **authors** can edit/delete their **own** posts.
* In case of **error** (e.g. Internet connection lost), an error message should be displayed.
* In case of **no posts**, display ‘No posts in database’.



* Format the entity creation **date** of a **post** as shown in the screenshot above. You may **use** the following helper function:

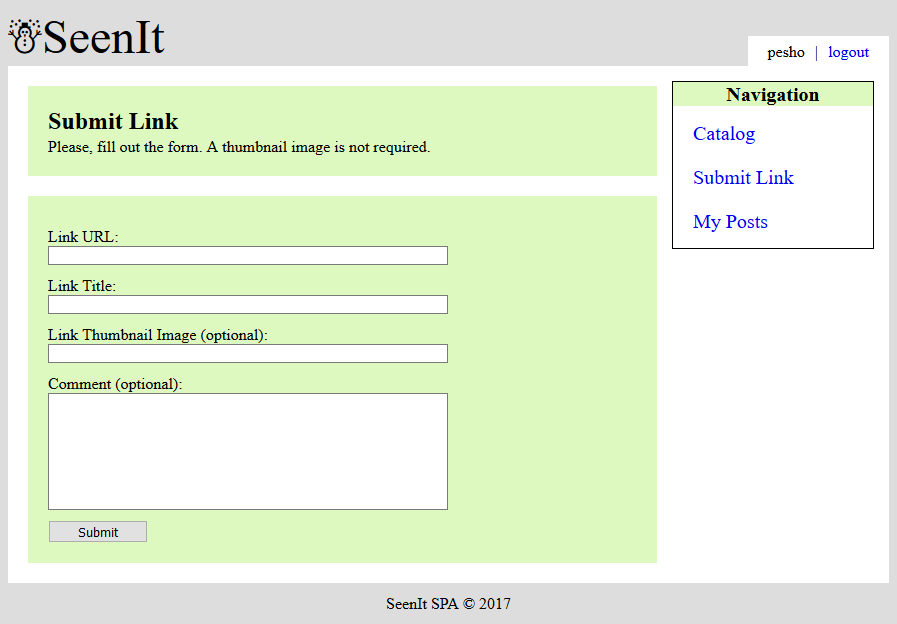
|  |
| --- |
| **function** *calcTime*(dateIsoFormat) {  **let** diff = **new** Date - (**new** Date(dateIsoFormat));  diff = ***Math***.floor(diff / 60000);  **if** (diff < 1) **return 'less than a minute'**;  **if** (diff < 60) **return** diff + **' minute'** + *pluralize*(diff);  diff = ***Math***.floor(diff / 60);  **if** (diff < 24) **return** diff + **' hour'** + *pluralize*(diff);  diff = ***Math***.floor(diff / 24);  **if** (diff < 30) **return** diff + **' day'** + *pluralize*(diff);  diff = ***Math***.floor(diff / 30);  **if** (diff < 12) **return** diff + **' month'** + *pluralize*(diff);  diff = ***Math***.floor(diff / 12);  **return** diff + **' year'** + *pluralize*(diff);  **function** *pluralize*(value) {  **if** (value !== 1) **return 's'**;  **else return ''**;  } } |

* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".

### Create Post Screen (10 pts)

Logged in users can **create** posts. Clicking the **[Submit Link]** link should open a form. A post has **author**, **url**, **title**, **imageUrl** (optional), **description** (optional).

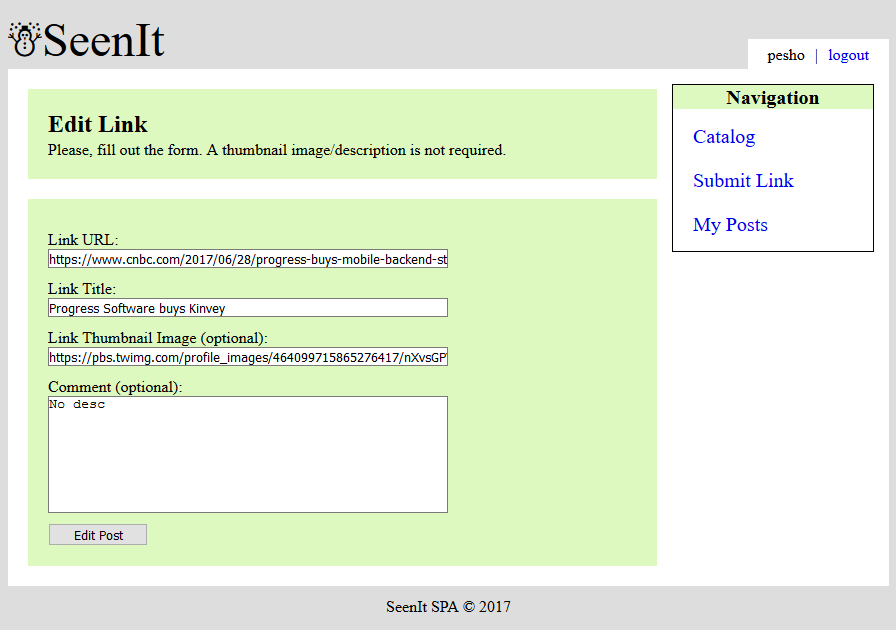
* After a **successful** post creation, a notification message “Post created.” should be displayed and the **catalog** should be shown.
* **Link url** and **title** are **not** optional, also link url should always start with “**http**”. In case of **empty/incorrect** input fields an appropriate **error** message should be **displayed** and the user should be able to fill the **form** again.
* Clear **all** input fields after **successful** creation.
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".



### Edit Post Screen (10 pts)

Authors **should** be able to **edit** their **own** posts. Clicking the [**Edit**] link on each post should open an edit **form**. Inside the edit form all input fields should be **filled** with the **current** data of each post.

* After a **successful** post update, a notification message “Post {postTitle} updated.” should be displayed and the **catalog** should be shown.
* **Link url** and **title** are **not** optional, also link url should always start with “**http**”. In case of **empty/incorrect** input fields an appropriate **error** message should be **displayed** and the user should be able to fill the **form** again.
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".



### Delete Post (5 pts)

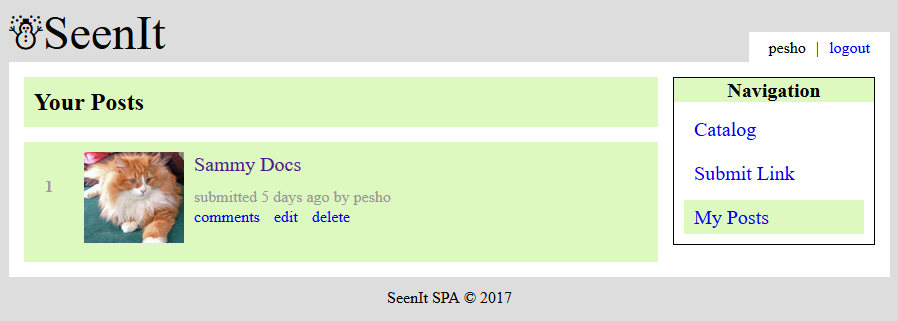
Authors **should** be able to **delete** their **own** posts by clicking the [**Delete**] button.

* After **successful** post delete a notification message “Post deleted.” should be displayed and the **catalog** should be **shown**.
* In case of **error** (e.g. Internet connection lost / unauthorized request / missing message), an error message should be displayed.
* Deleting works immediately, with **no confirmation**.

### My Posts (5 pts)

Each **user** should be able to view his own posts by clicking [**My Posts**].

* The **posts** should be listed like in the **catalog** section.
* In case of **error** (e.g. Internet connection lost), an error message should be displayed.
* In case of **no posts**, display ‘No posts in database’.



### Post Details (15 pts)

Each **post** has a details view. By clicking the [**Comments**] link **partial** information about each post should be listed (post **image**, post **title**, post **description**), and most importantly each comment **connected** to the post.

* In case of **error** (e.g. Internet connection lost), an error message should be displayed.
* If there is **no** description add “No description” in it’s place.
* If there are **no** comments under the post display “No comments yet.”
* Each comment author has the **ability** to **delete** their own comment.
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".



### Create Comment (5 pts)

Every **logged** in user has the ability to **create** a comment under a post by clicking [**Add Comment**].

* After a **successful** comment creation, a notification message “Comment created.” should be displayed and the post details should be shown **again** with the newly added comment.
* Clear the **input** field after **successful** creation.
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".

### Delete Comment (5 pts)

Every **author** has the ability to delete his own comment under a post by clicking [**Delete**]

* After a **successful** comment delete, a notification message “Comment deleted.” should be displayed and the post details should be shown **again** without the deleted comment.
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".