# Homework 8: Responsive Web Design

Facebook Graph Search

(AJAX/JSON/Bootstrap/jQuery/AngularJS/Cloud Exercise)

## 1. Objectives

- Become familiar with the AJAX and JSON technologies
- Use a combination of HTML5, CSS, Bootstrap, jQuery, AngularJS and PHP
- Get hands-on experience in Google Cloud App Engine or Amazon Web Services
- Get hands-on experience on how to use Bootstrap to enhance the user experience
- Provide an interface to perform Facebook search using Facebook Graph APIs

# 2. Background

## 2.1 AJAX and JSON

Ajax (short for Asynchronous JavaScript and XML) is a mechanism which enables the communication between the client and the server without the need for a post-back or a complete page refresh. It can also be defined as a method of exchanging data with a server, and updating some parts of a web page – without reloading the entire page.

In this exercise, you have to use the AJAX component provided in JQuery. Please see the class slides at:

http://cs-server.usc.edu:45678/slides/jQueryTutorial.ppt

JSON, short for JavaScript Object Notation, is a lightweight data interchange format. Its main application is in AJAX web application programming, where it serves as an alternative to the use of the XML format for data exchange between client and server. See the class slides at:

http://cs-server.usc.edu:45678/slides/JSON1.ppt

## 2.2 Bootstrap

Bootstrap is a free collection of tools for creating responsive websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. To learn more details about Bootstrap please refer to the lecture material on Responsive Web Design (RWD). See the class slides at:

http://cs-server.usc.edu:45678/slides/Responsive.pdf

and

http://getbootstrap.com/

## 2.3 Facebook Graph API

The Graph API is the primary way to get data out of, and put date into Facebook's platform. It's a low-level HTTP-based API that you can use to programmatically query data, post new stories, manage ads, upload photos, and perform a variety of other tasks that an app might implement.

To learn more about the Facebook Graph API visit the page
https://developers.facebook.com/docs/graph-api

## 2.4 Hosting Cloud Services

You have two options to host your page either in Amazon Web Services (AWS) or Google App Engine (GAE).

## 2.4.A. Amazon Web Services

AWS is Amazon's implementation of cloud computing. AWS includes Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You can simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS 7.5 for .NET, and Apache Tomcat for Java.

The Amazon Web Services homepage is available at: http://aws.amazon.com/

## 2.5.B. Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and noSQL databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

To learn more about GAE support for PHP visit the page:

https://cloud.google.com/appengine/docs/php/

# 3. Homework Description

Similar to the homework #6, in this exercise you will create a webpage that allows users to query about users, pages, groups, events and places.

The difference being, in this homework you will create a PHP script to return a JSON formatted data stream to the front-end. The client parses the JSON data and renders it in a nicer-looking, responsive UI (i.e., using Bootstrap).

A user first opens a page as shown below in Figure 1. The page includes a navigation menu at the top and the users' information table at the bottom. By default, the result for the input value (*USC*) has to be displayed.

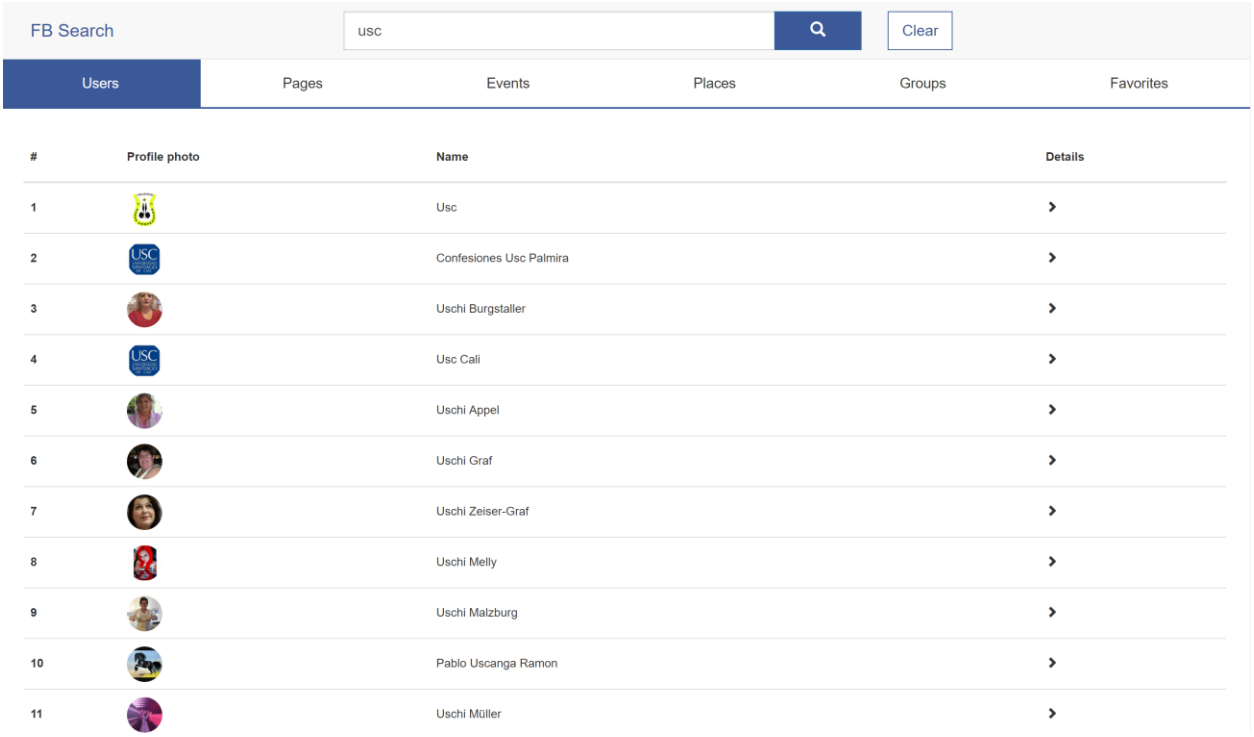



**Figure 1 – Users' page**

Once the user opens this page, all of the queried data is loaded using the AJAX component in JQuery or Angular JS. The AJAX code calls your PHP script. The webpage must then use JavaScript to extract data from the JSON response and display the results on the same webpage. We explain how to display the results in the Hints section.

## 3.1 Page Header

At the top of the webpage, there is a page header which contains an input field with a search button and clear button and the title to the left. The header is shown in Figure 2.

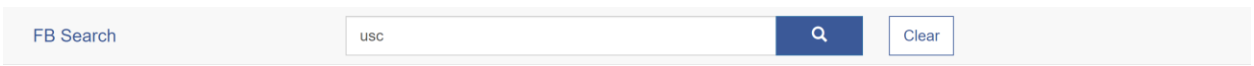


**Figure 2 – Page Header**

- The image is linked to the same page.
- On clicking the clear button, the search field along with the results have to be cleared.

## 3.2 Navigation Bar

The navigation bar should contain 6 tabs (Users, Pages, Events, Places, Groups and Favorites). The navigation bar is shown in Figure 3.

**Figure 3 – Navigation Bar**

When the user clicks on a tab in this navigation bar, the corresponding content should be loaded. For example, when the user clicks on Pages, the pages information should be loaded to the content part.

## 3.3 The Users Tab

### 3.3.1 Design

You must replicate the table displayed in Figure 4 using a **Bootstrap table**.

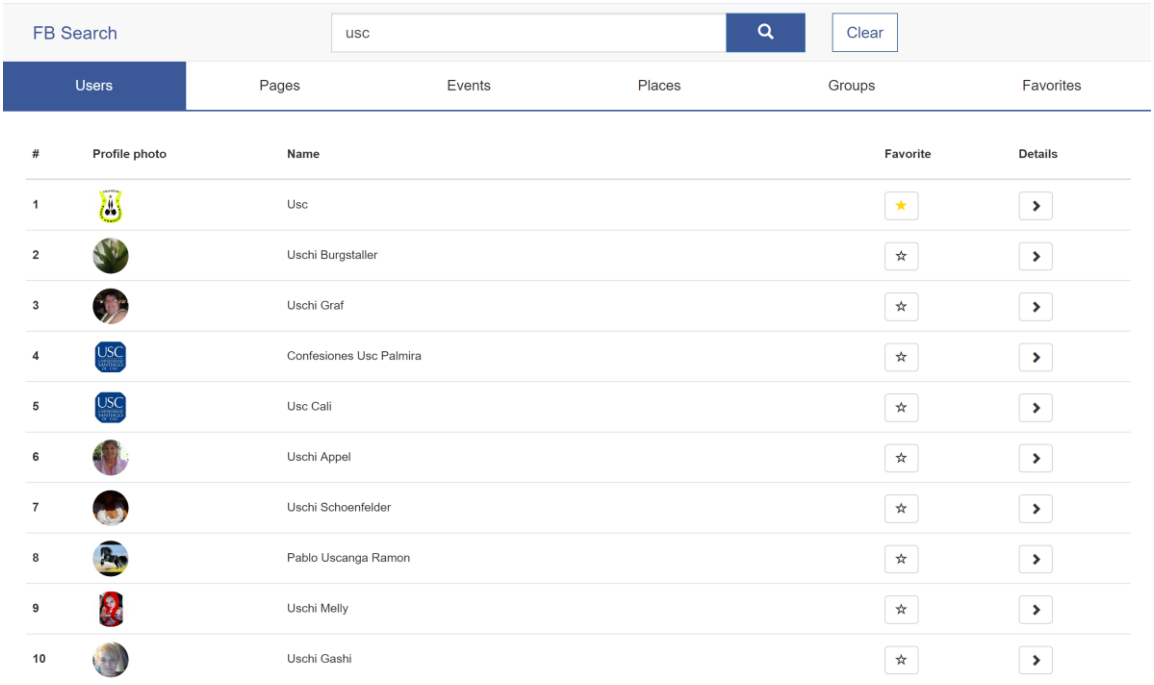



**Figure 4 – User's Table**

- The page starts with a table having the following columns as '#', 'Profile Photo', 'Name', 'favorite', 'Details'.

- You need to display all the first 25 records returned by the Facebook service.
- A record can be marked favorite by the user by clicking the favorite button.
- When the results are loaded if a record is in the favorite list of the user, it has to be colored.

You should replicate the result table as shown in Figure 4.

The table contains the information shown in Table 1.

| Table Field | Description |
|---|---|
| **#** | Displays the row number |
| **Profile Photo** | Displays the Profile Picture of the User. |
| **Name** | Displays the name for the User. |
| **Favorite** | A button that shows whether the record is present in the favorite list. |
| **Details** | Displays a button. Information about this button will be enclosed in later section. |

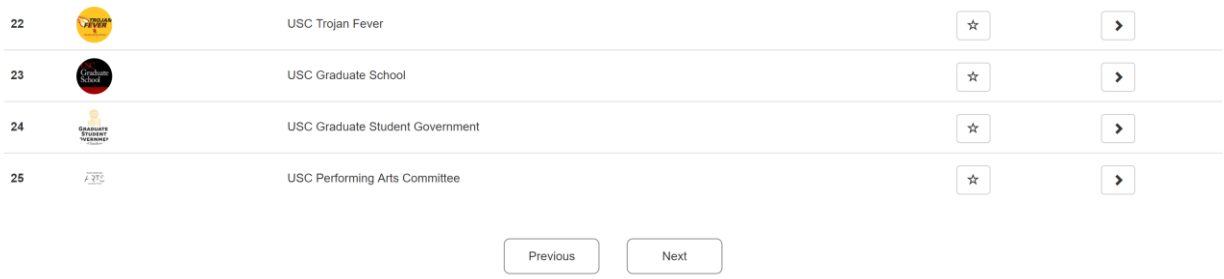
**Table 1: Users table description**

### 3.3.2 Pagination

The table should be paginated. Each page displays 25 records. The previous and the next pages can be obtained from the pages block of the returned JSON object. The following figure shows the pagination block returned from the API

```
"paging": {
  "next": "https://graph.facebook.com/v2.8/search?fields=id,name&limit=25&offset=50&
  "previous": "https://graph.facebook.com/v2.8/search?fields=id,name&limit=25&offset
}
```

The next records are present at “next” attribute of the “paging” block. The previous records are present at “previous” attribute of the corresponding block. The links to the previous and next page looks as shown in the figure.




## 3.3.3 Details

When the “Details” button is clicked, the page should slide to the detail page of the corresponding record. Please check the video in order to know more about the sliding functionality. In order to implement the ‘sliding’ mechanism, you must use Angular JS.

The detail page is shown in Figure 5.

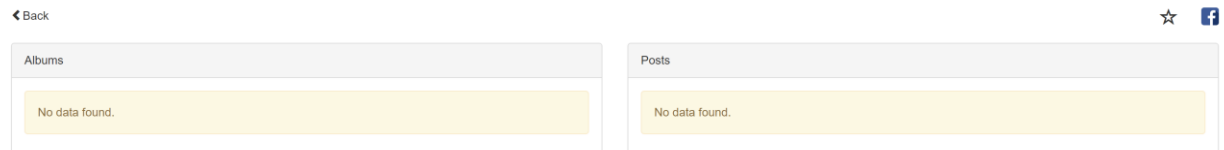



**Figure 5a – User Details when no albums and posts.**

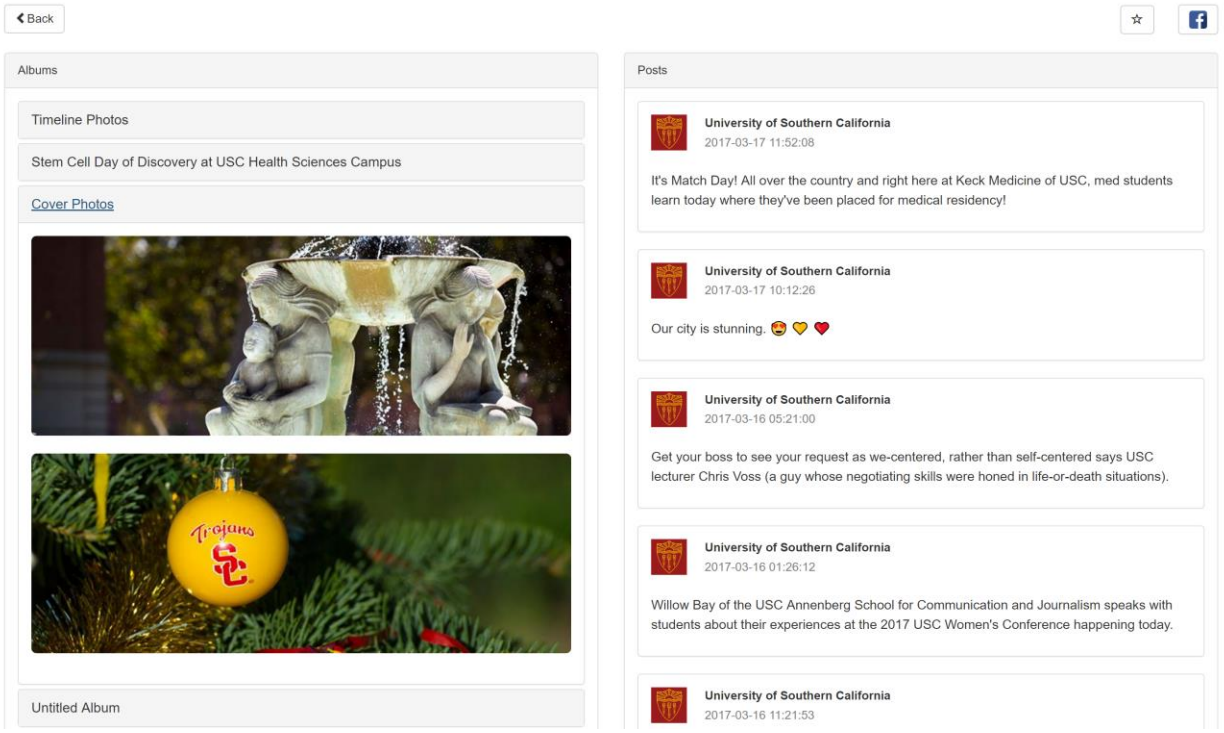



**Figure 5b – User Details when albums and posts are present and cover photos is clicked.**

The detail page contains 2 parts:

- A table containing the top 5 albums of the user
- A table containing the top 5 posts of the user

### 3.3.3.1 Albums

The albums table contains at most 5 albums of the user where each album contains at most two pictures. The albums has to be displayed as shown in the figure 6.

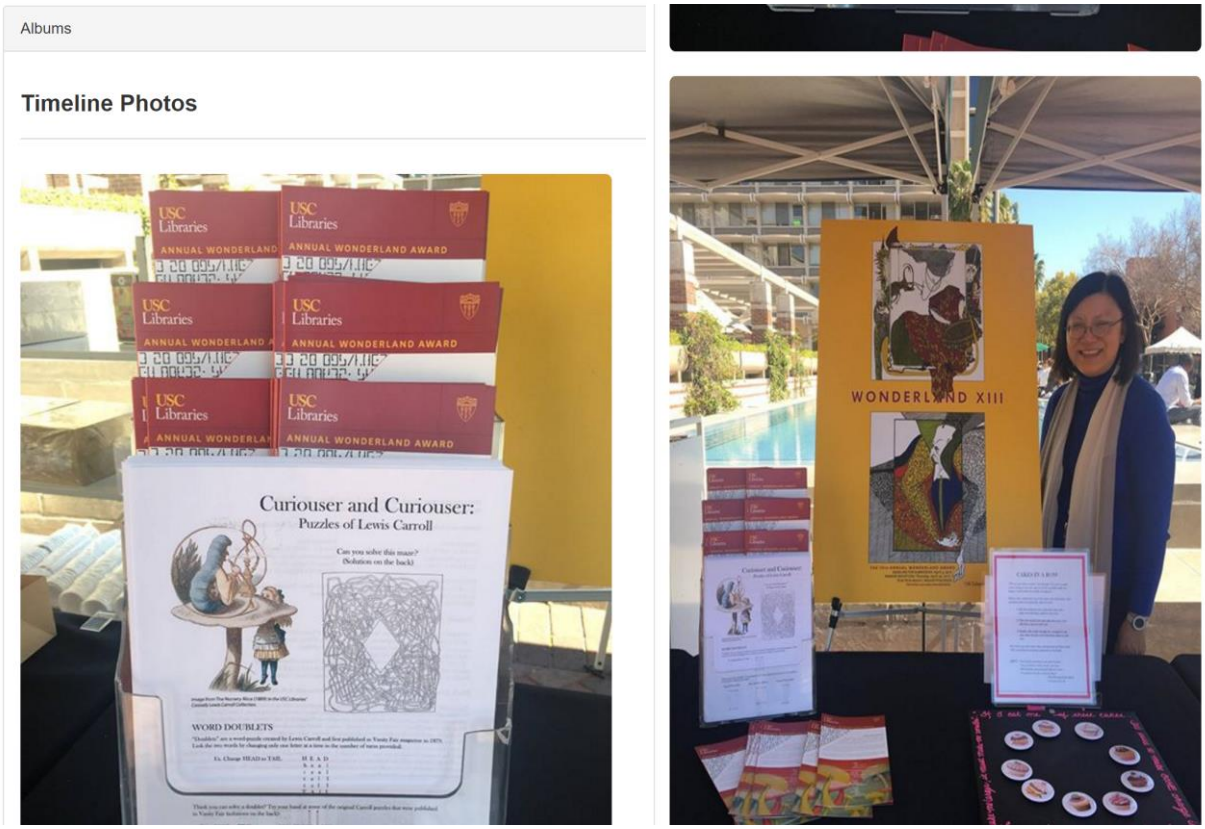



**Figure 6 - Albums in details section**

Each album should have “Title of the album” followed by the two pictures as shown in the figure.

Similar to the Homework #6, an album is to be displayed only when it is clicked.

By default the first album is opened and is displayed. When an album is clicked, the photos has to open in a sliding fashion. Please refer to the video in order to understand this mechanism.

### 3.3.3.2 Posts

The posts table contains at most 5 posts of the user. The posts have to be presented as shown in Figure 7.

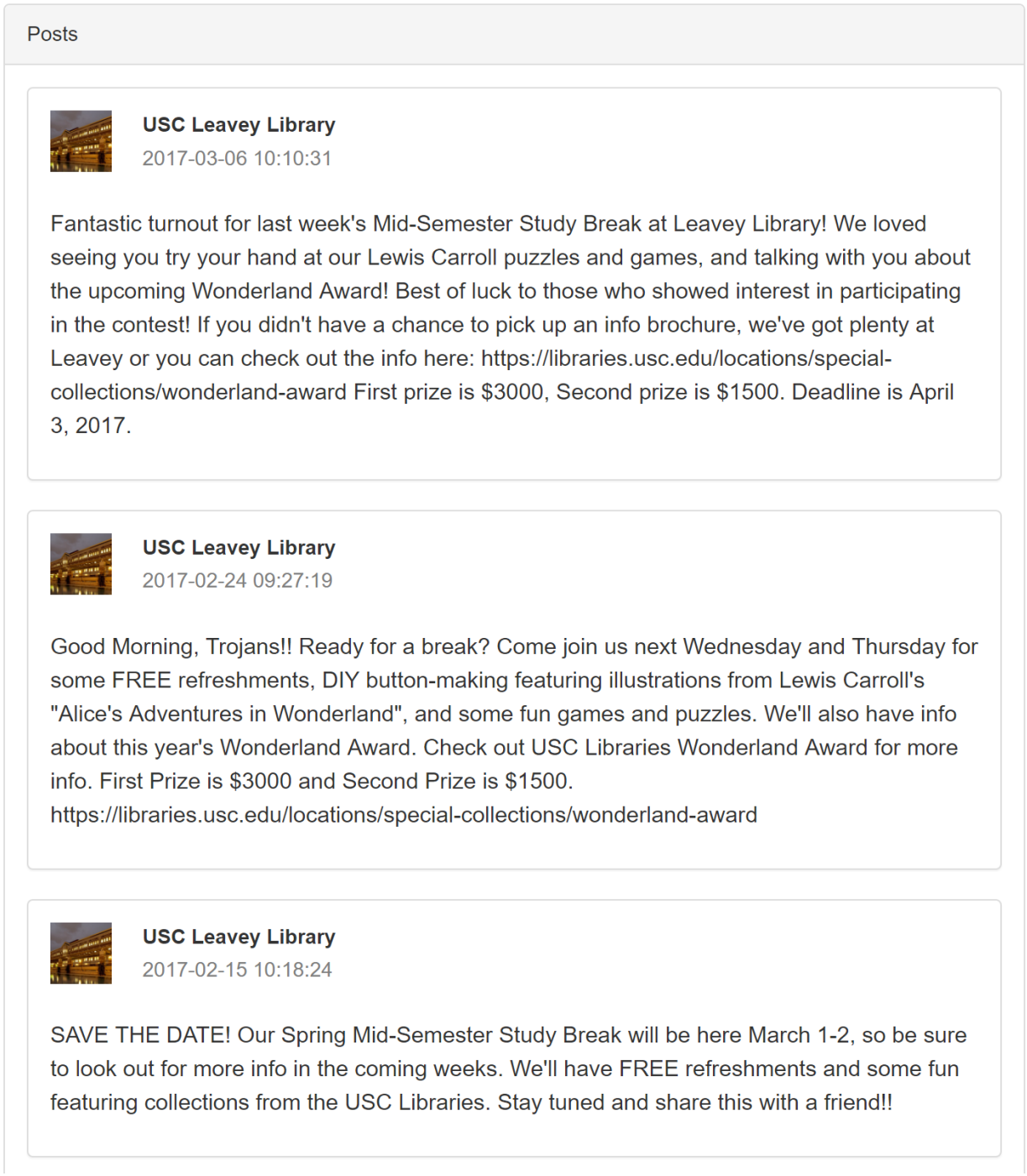



**Figure 7: Posts in details section.**

- Each post should have “The name of the user”, “Date and Time of the post” and “the message” as shown in Figure 7.
- The Date and Time can be calculated with the help of Moment.js. More Hints are provided in the Hints section.

For more information, please refer to the video for the homework.

### 3.3.4 Back Button

At the top left of the detail page, there is a back button. When clicked, the page should go back to the Users table ( See Figure 8).

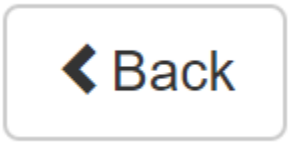



**Figure 8 – Back Button**

### 3.3.5 Favorite Button

At the top right of the detail page, there is a favorite button. When clicked, the corresponding user record is added to the Favorites list. If clicked again, the user has to be removed from the Favorites list.

**Figure 9 - Before and after click**

- The favorite button should allow the user to add the user to the Favorites list and store it in the browser's local storage.
- If the record is in the Favorites list, the button should have a yellow star, otherwise a white star (see Figure 8).
- When clicking on the Favorite button, the record should be stored in HTML5 local storage collection. Through this storage, you can store items in the persistent local storage belonging to the client browser (i.e., when closing the browser, you can still retrieve the information). For more information, please refer to section 5.6.

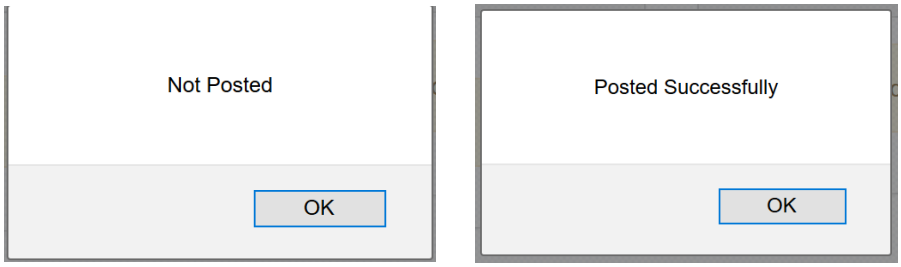
### 3.3.6 Facebook button

Next to the favorite button, a Facebook logo is to be displayed. Upon clicking the Facebook logo, there has to be a pop up in order to post about the user in to the clients' timeline as shown in Figure 10.

The background image of the pop up is the profile picture of the "user". With the name as the name of the "user" and the caption has to be "***FB SEARCH FROM USC CSCI571***"

**Figure 10: Posting to client's timeline**

A Message has to be displayed whether the post is shared on client's timeline. Sample message when the post is made are as shown below.




More hints are provided in the Hints section.

### 3.3.7 Progress bar

Whenever data is being fetched, a progress bar has to be displayed as shown in the figure 11. Please check the video in order to implement this functionality.

You can use the progress bar component of Bootstrap to implement the feature. You can find hints about the bootstrap components in the Hints section.

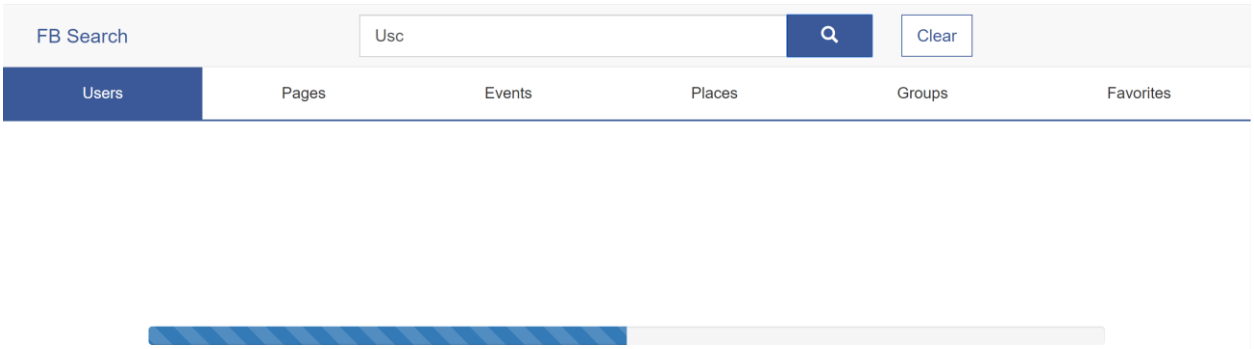



**Figure 11: Loading the data from the server**

# 3.4 The Pages Tab

## 3.4.1 Design

You must replicate the table displayed in Figure 12 using a **Bootstrap table**.

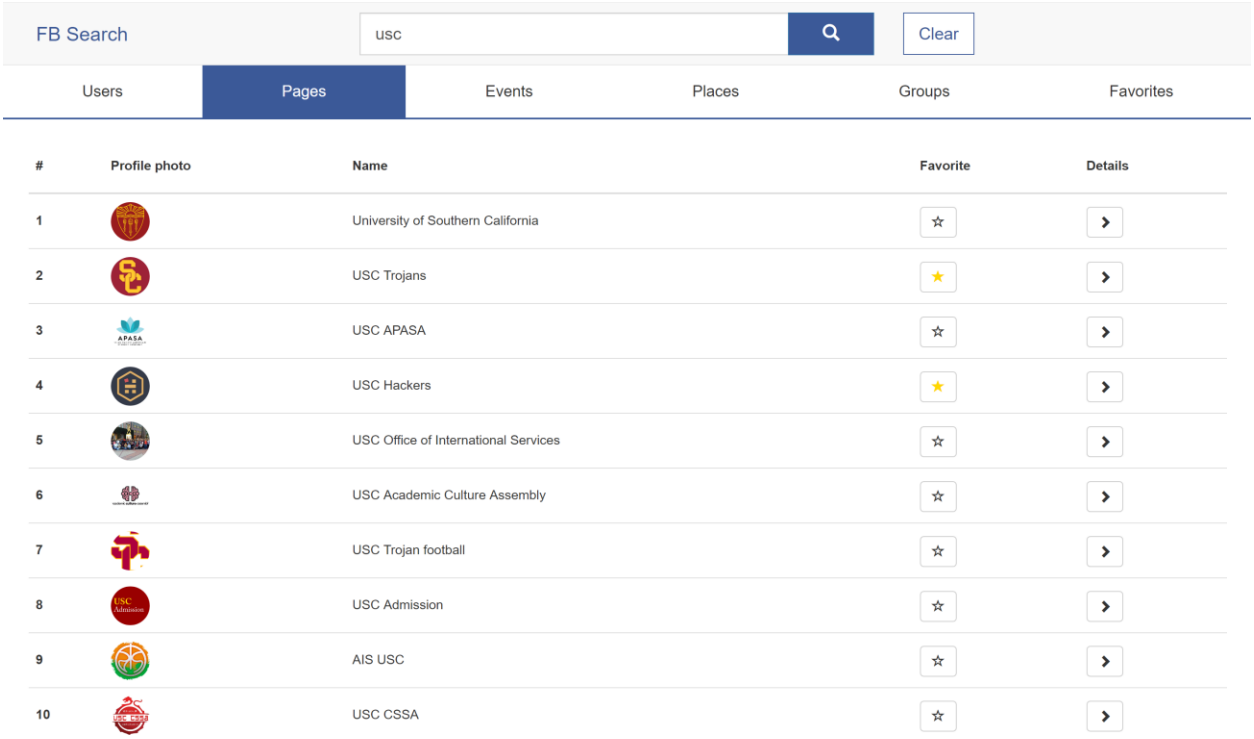


**Figure 12 – Pages' table**

- The Pages' table also contains 4 fields: "#", "Profile Photo", "Name", "Favorites" and "Details"
- The table has to contain 25 records and pagination has to be provided.
- A record can be marked favorite by the user by clicking the favorite button.
- When the results are loaded if a record is in the favorite list of the user, it has to be colored

You must replicate the table displayed in Figure 12. The table fields are listed in Table 2.

| Table Field | Description |
|---|---|
| **#** | The row number. |
| **Profile Photo** | The Profile Picture of the "page". |
| **Name** | The name of the page. |
| **Favorite** | A button that shows whether the record is present in the favorite list. |
| **View Details** | View Details button. When clicked, the page should navigate to the detail page of the "Page". |

**Table 2: "Pages" table description**

### 3.4.2 Pagination

The result table should be paginated. Details can be looked up in Section 3.3.2.

### 3.4.3 View Details

When the "Details" button is clicked, the page should slide to the detail page of the corresponding record. Please check the video in order to know more about the sliding functionality.

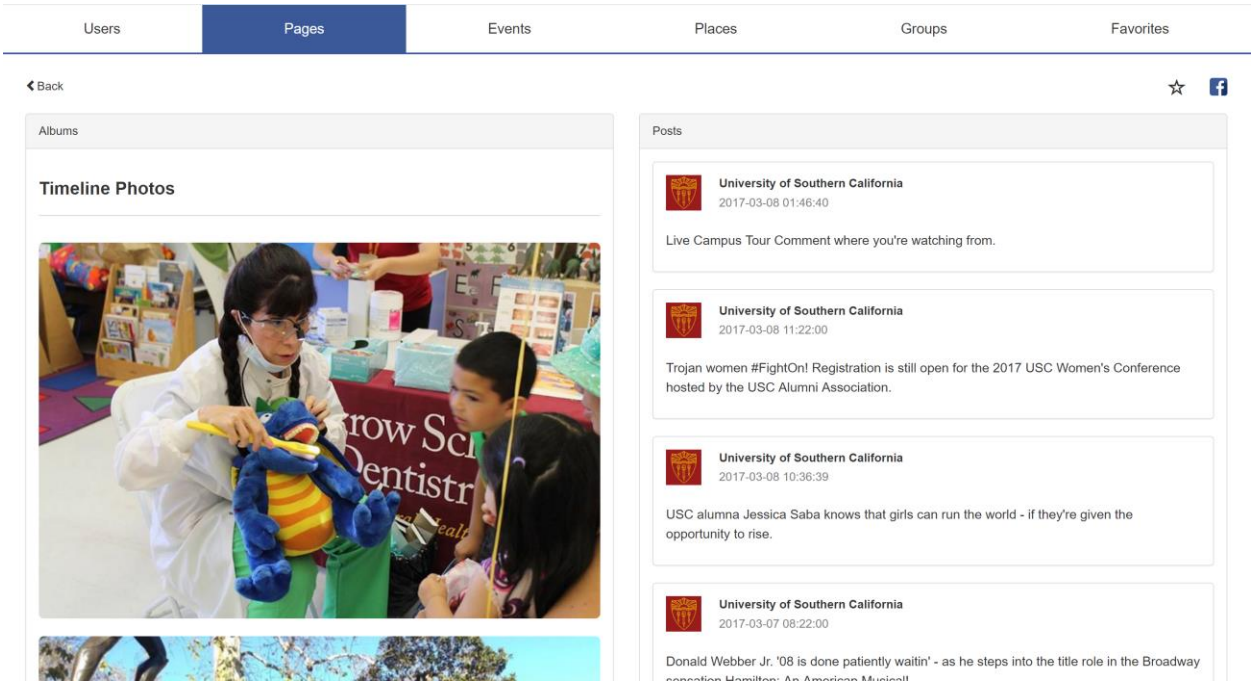



**Figure 13 – "Pages" Detail page**

Similar as described in Section 3.3.3, you need to display the albums and posts for the "page" selected.

## 3.4.4 Back button

Similar as described in Section 3.3.4. When clicked, it should navigate back to the "Pages" table.

## 3.4.5 Favorite button

Similar as described in Section 3.3.5. If this "Pages" is in the Favorites list, the user should see a yellow button when entering this page's detail page.

## 3.4.6 Facebook button

Similar as described in Section 3.3.6. On clicking the Facebook logo, there has to be a pop up in order to post about the user in to the clients' timeline. More hints are provided in the Hints Section.

## 3.4.7 Progress bar

Whenever data is being fetched, a progress bar has to be displayed as shown in the figure 14. Please check the video in order to implement this functionality.

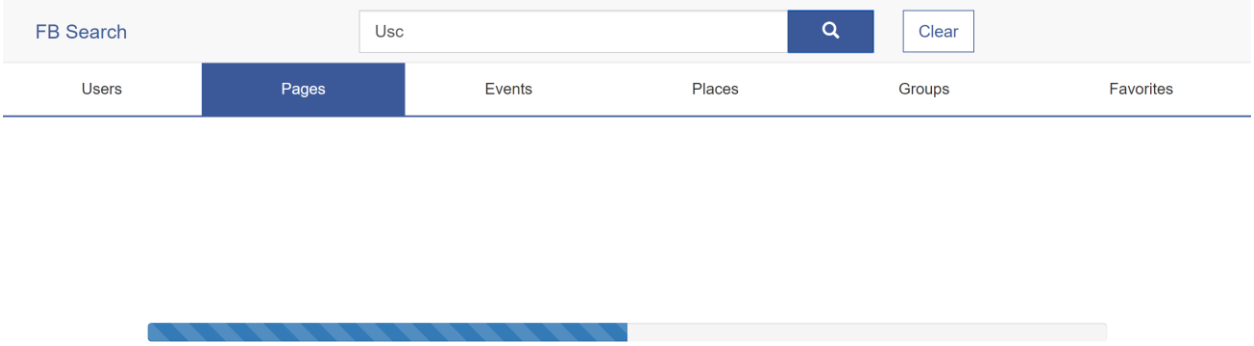



**Figure 14: Loading the data from the server**

# 3.5 The Events Tab

## 3.5.1 Design

You must replicate the table displayed in Figure 15 using a **Bootstrap table**.

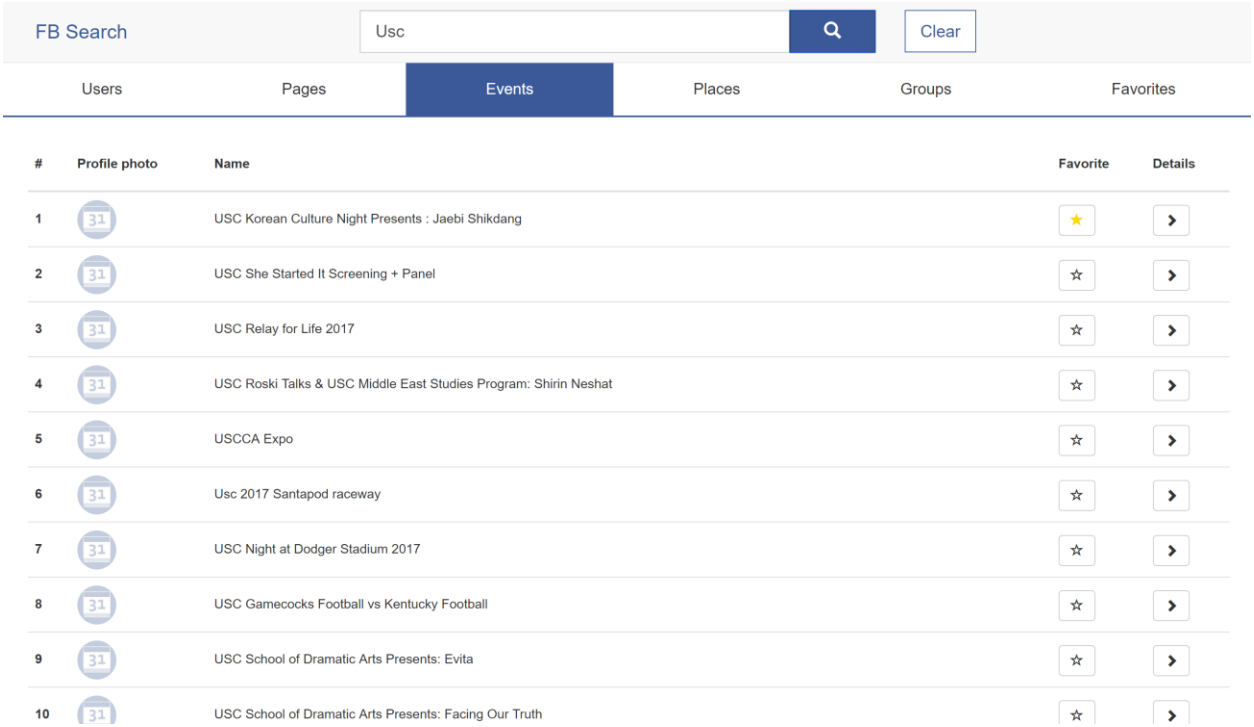



**Figure 15 – Places page**

- The Pages' table also contains 5 fields: "#", "Profile Photo", "Name", "Favorites" and "Details"
- There has to be 25 records for every page. Pagination has to be provided to move from pages.

- A record can be marked favorite by the user by clicking the favorite button.
- When the results are loaded if a record is in the favorite list of the user, it has to be colored
- You must replicate the table displayed in Figure 15. The table fields are as shown in Table 3.

| Table Field | Description |
|---|---|
| **#** | The row number. |
| **Profile Photo** | The Profile Picture of the “event”. |
| **Name** | The name of the event. |
| **Favorite** | A button that shows whether the record is present in the favorite list. |
| **View Details** | View Details button. When clicked, the page should slide to the detail page of the corresponding record. |

**Table 3: Events table description**

### 3.5.2 Pagination

The result table should be paginated. Details can be looked up in Section 3.3.1.2.

### 3.5.3 Details

When the “details” button is clicked, the page should slide to the details of the event as shown in Figure.

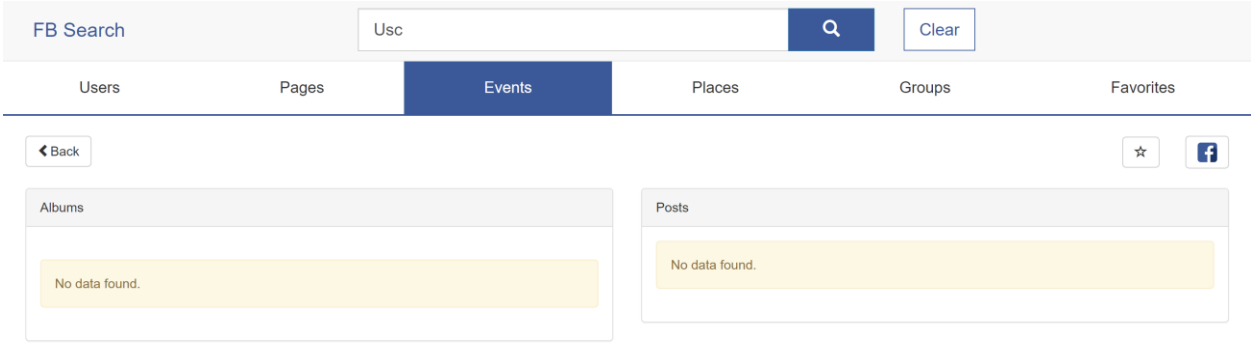



**Figure 16 – Details section for events**.

Similar as described in Section 3.3.3, you need to display the albums and posts for the “events” selected See Figure 16.

### 3.5.4 Back button

Similar as described in Section 3.3.4. When clicked, it should navigate back to the “Places” table.

### 3.5.5 Favorite button

Similar as described in Section 3.3.5. If this “Places” is in the Favorites list, the user should see a yellow button when entering this page’s detail page.

### 3.5.6 Facebook button

Similar as described in Section 3.3.6. On clicking the Facebook logo, there has to be a pop up in order to post about the “Place” in to the clients’ timeline. More hints are provided in the Hints Section.

### 3.5.7 Progress bar

Whenever data is being fetched, a progress bar has to be displayed as shown in the figure. Please check the video in order to implement this functionality.

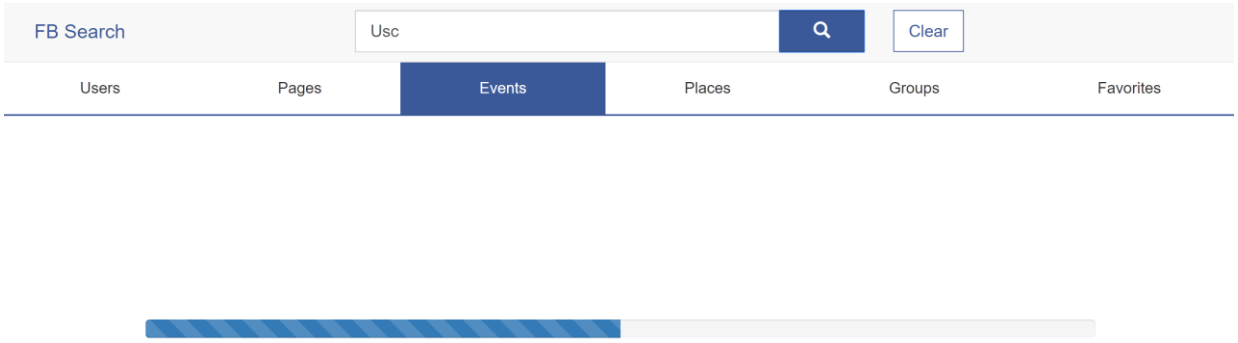



**Figure 17: Loading the data from the server**

## 3.6 The Places Tab

### 3.6.1 Design

You must replicate the table displayed in Figure 18 using a **Bootstrap table**.

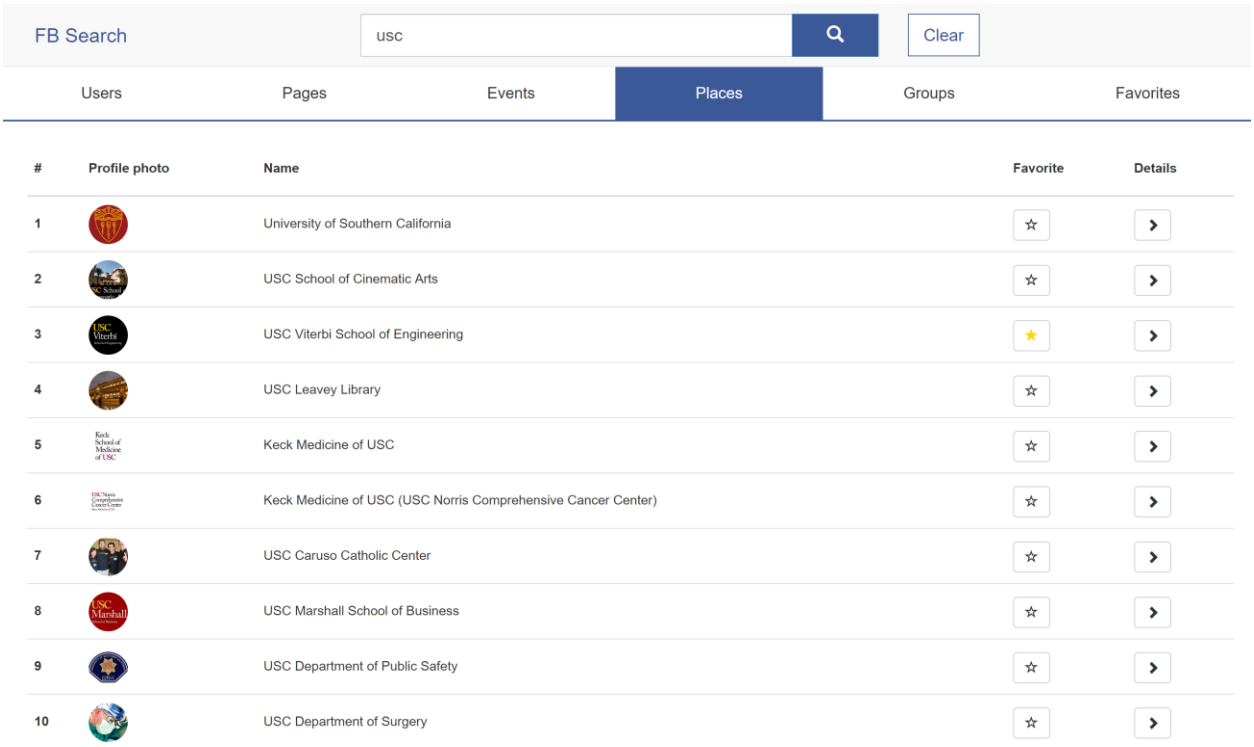



**Figure 18 – Places page**

- The Pages' table also contains 5 fields: "#", "Profile Photo", "Name", "Favorites" and "Details"
- There has to be 25 records for every page. Pagination has to be provided to move from pages.
- A record can be marked favorite by the user by clicking the favorite button.
- When the results are loaded if a record is in the favorite list of the user, it has to be colored
- You must replicate the table displayed in Figure 18. The table fields are as shown in Table 4.

| **Table Field** | **Description** |
|---|---|
| **#** | The row number. |
| **Profile Photo** | The Profile Picture of the "page". |
| **Name** | The name of the page. |
| **Favorite** | A button that shows whether the record is present in the favorite list. |
| **View Details** | View Details button. When clicked, the page should navigate to the detail page of the "Page". |

**Table 4: Places description**

## 3.6.2 Getting User's Location:

Instead of asking client for the location and distance, you need to get the client's location using JavaScript. It can be done using the attributes of "navigator" object.

There is a method in JavaScript that can be used in order to get the current location of the client. Please refer to the following website for more details.

https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/getCurrentPosition

## 3.6.3 Pagination

The result table should be paginated. Details can be looked up in Section 3.3.1.2.

## 3.6.4 Details

When the "Details" button is clicked, the page should slide to the detail page of the corresponding record. Please check the video in order to know more about the sliding functionality.

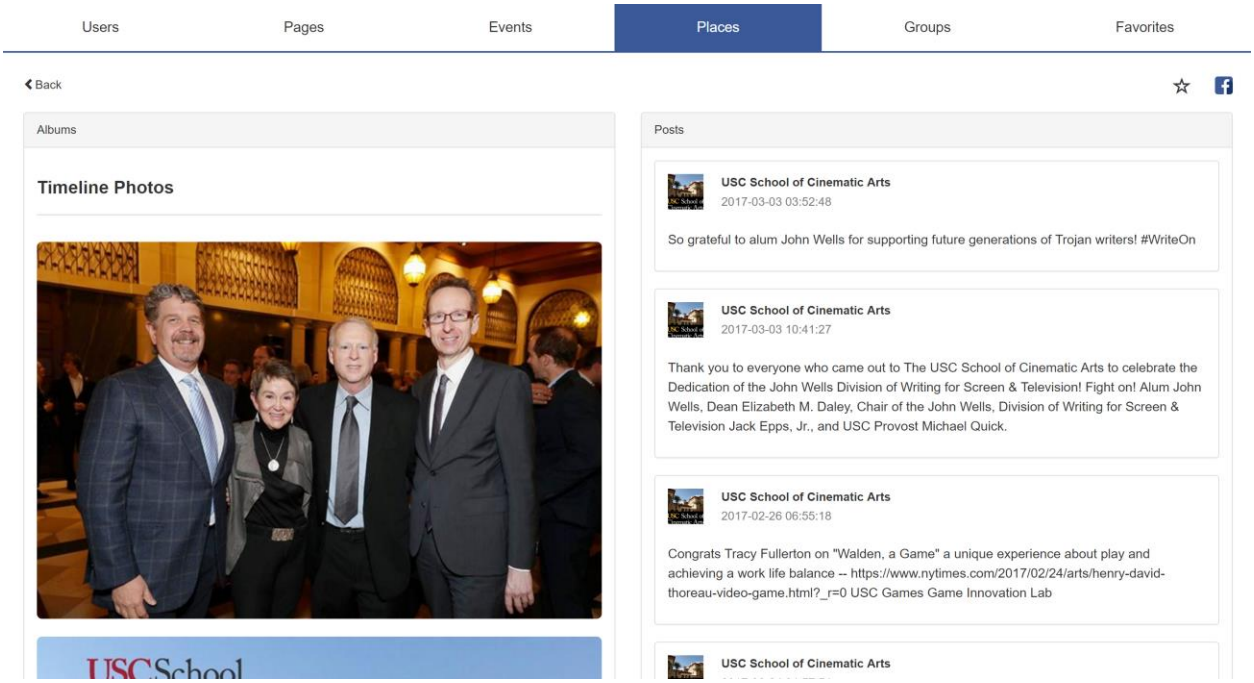



**Figure 19 – Details section for Places**.

Similar as described in Section 3.3.3, you need to display the albums and posts for the "Places" selected See Figure 19.

## 3.6.5 Back button

Similar as described in Section 3.3.4. When clicked, it should navigate back to the “Places” table.

### 3.6.6 Favorite button

Similar as described in Section 3.3.5. If this “Places” is in the Favorites list, the user should see a yellow button when entering this page’s detail page.

### 3.6.7 Facebook button

Similar as described in Section 3.3.6. On clicking the Facebook logo, there has to be a pop up in order to post about the “Place” in to the clients’ timeline. More hints are provided in the Hints Section.

### 3.6.8 Progress bar

Whenever data is being fetched, a progress bar has to be displayed as shown in the figure. Please check the video in order to implement this functionality.

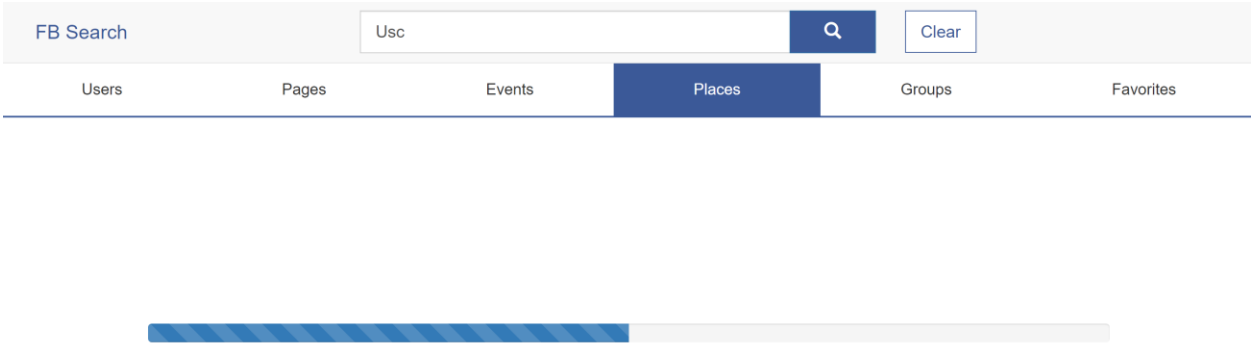



**Figure 20 – Loading the data from the server**

## 3.7 The Groups Tab

### 3.7.1 Design

You must replicate the table displayed in Figure 21 using a **Bootstrap table**.

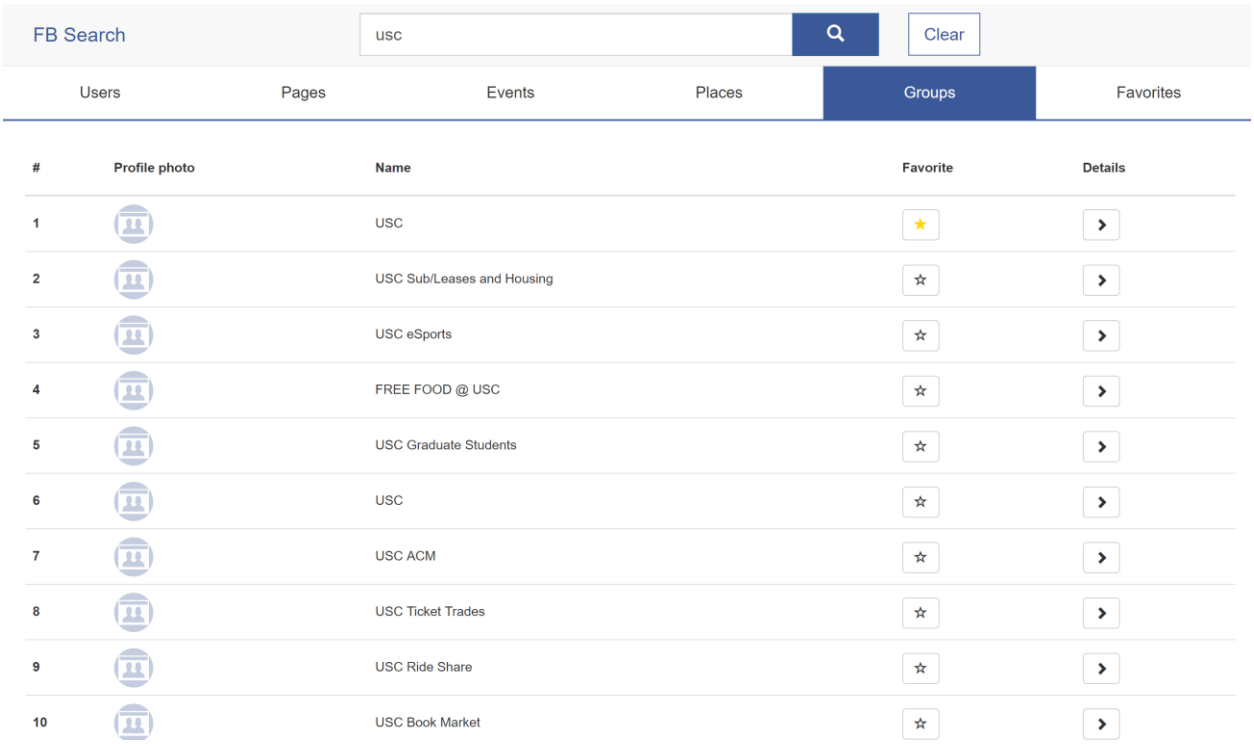



**Figure 21 – Groups' table**

- The Groups' table also contains 5 fields: "#", "Profile Photo", "Name", "Favorite" and "Details"
- The table has to contain 25 records and pagination has to be provided.
- A record can be marked favorite by the user by clicking the favorite button.
- When the results are loaded if a record is in the favorite list of the user, it has to be colored

You must replicate the table displayed in Figure 21. The table fields are listed in Table 5.

| **Table Field** | **Description** |
|---|---|
| **#** | The row number. |
| **Profile Photo** | The Profile Picture of the "group". |
| **Name** | The name of the group. |
| **View Details** | View Details button. When clicked, the page should navigate to the detail page of the "group". |

**Table 5: "Groups" table description**

## 3.7.2 Pagination

The result table should be paginated. Details can be looked up in Section 3.3.2.

## 3.7.3 View Details

When the “Details” button is clicked, the page should slide to the details page of the corresponding group (as shown in Figure 22).

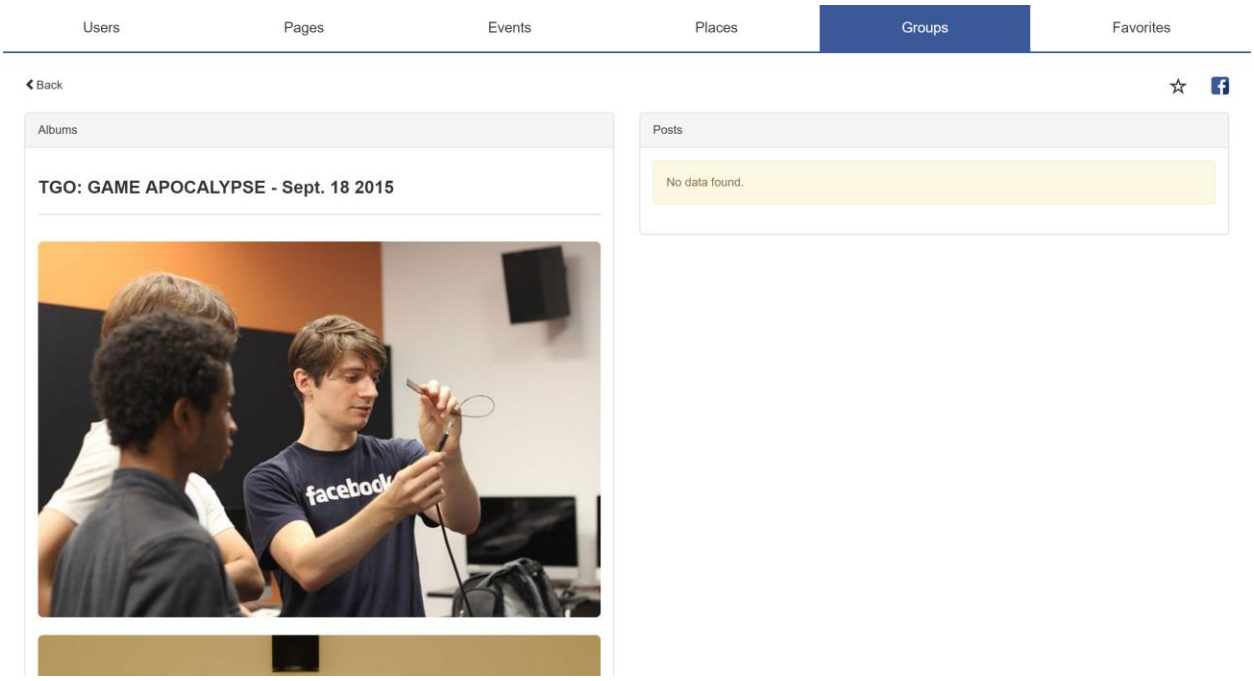



**Figure 22 – “Groups” Detail page**

Similar as described in Section 3.3.3, you need to display the albums and posts for the “groups” selected See Figure 14.

## 3.7.4 Back button

Similar as described in Section 3.3.4. When clicked, it should navigate back to the “groups” table.

## 3.7.5 Favorite button

Similar as described in Section 3.3.5. If this “groups” is in the Favorites list, the user should see a yellow button when entering this page’s detail page.

## 3.7.6 Facebook button

Similar as described in Section 3.3.6. On clicking the Facebook logo, there has to be a pop up in order to post about the group in to the clients’ timeline. More hints are provided in the Hints Section.

## 3.7.7 Progress bar

Whenever data is being fetched, a progress bar has to be displayed as shown in the figure 23. Please check the video in order to implement this functionality.

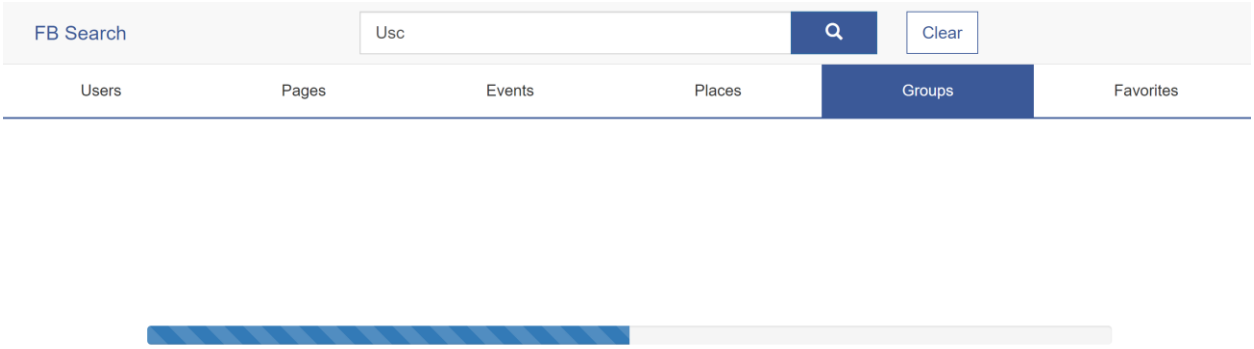



**Figure 23 – Loading the data from the server**

# 3.8 Favorites

## 3.6.1 Design

You must replicate the table displayed in Figure 24 using a **Bootstrap table**.

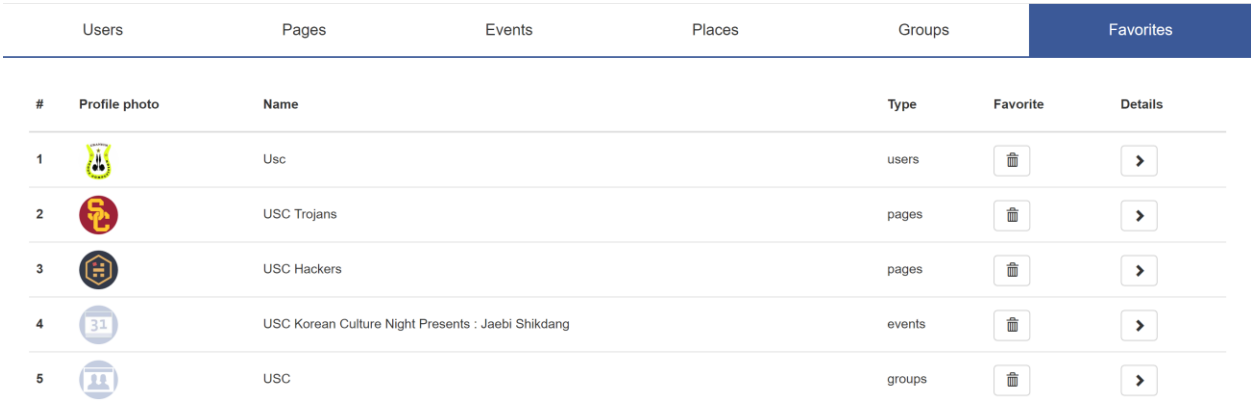



**Figure 24 Favorites table**

- The Favorites page contains the following columns: “#”, “Profile Photo”,“Name”, “Type”, “Favorite”, “Details”.
- There has to be a delete button as shown in the figure. Upon clicking the delete button, the corresponding record has to be deleted from the local storage of the user and from the favorites list
- You don’t need to implement pagination for the table.
- On clicking the details button, you need to show the albums and posts for the selected record similar to the previous tabs.
- You should use the LocalStorage of the browser in order to implement Favorites. More hints on usage of LocalStorage are provided in Section 5.6.

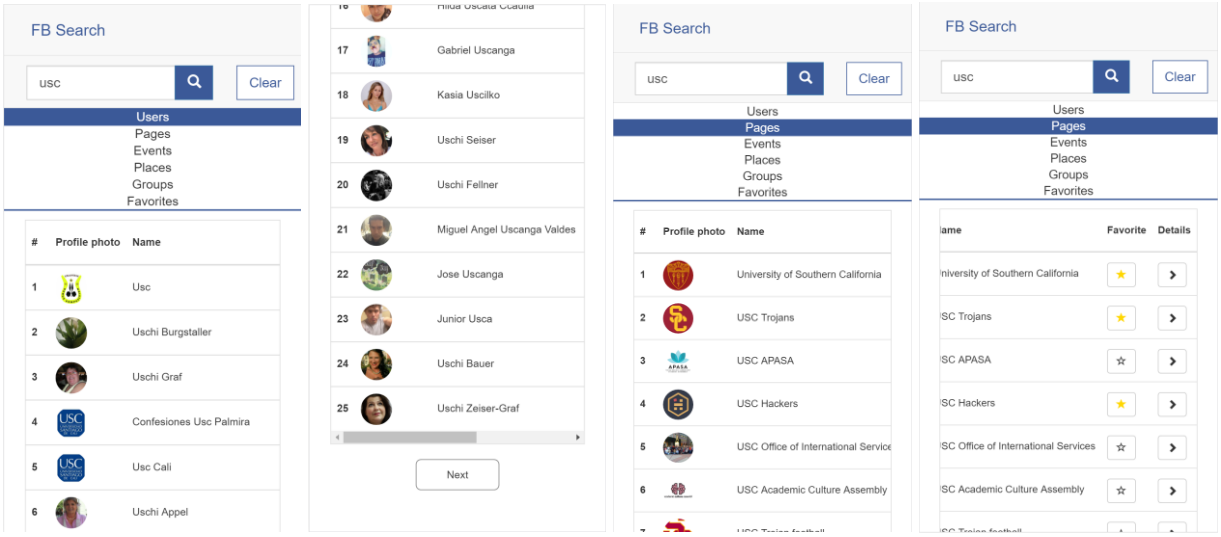
You must replicate the table displayed in Figure 24. The table fields are listed in Table 6.

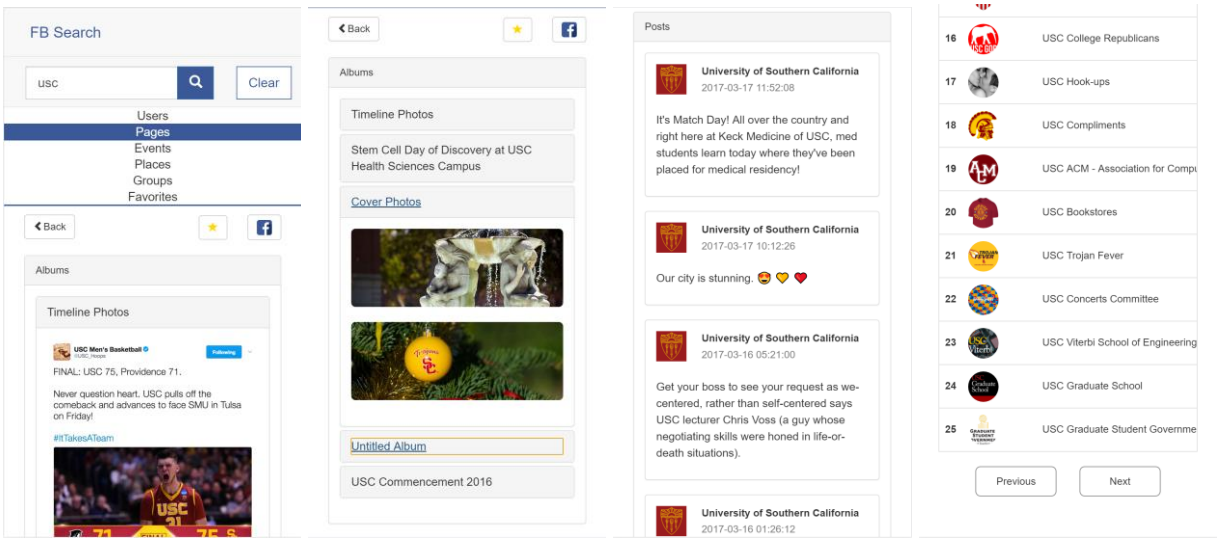
| Table Field | Description |
|---|---|
| **#** | The row number. |
| **Profile Photo** | The Profile Picture of the object. |
| **Name** | The name of the object. |
| **Type** | The type of the record/object |
| **Favorite** | A button to remove the object from favorite list |
| **View Details** | View Details button. When clicked, the page should navigate to the detail page of the "group". |

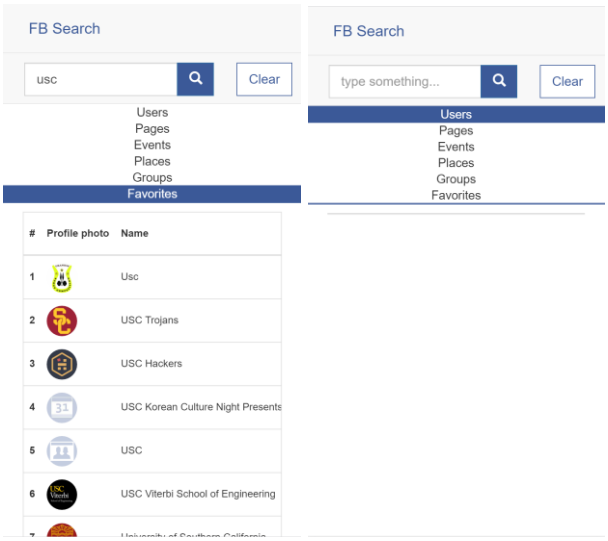
**Table 6: "Favorites" table description**

## 3.9 Responsive Behavior

The following are snapshots of the major screens taken on an iPhone 6 plus.






Some requirements in the mobile view are listed here:

- The navigation bar should display the types in a vertical way on smaller screens.
- Each table content for all the types can be scrolled horizontally.
- The albums and photos table has to be aligned vertically on smaller screens.

# 4. API Documentation

The same APIs that were used for homework 6 can be used for homework 8 as well.

## 4.1 User's Facebook API

The API to query all of the Users is:

https://graph.facebook.com/v2.8/search?q=The_Keyword_to_be_searched&type=user&fields=id,name,picture.width(700).height(700)&access_token=Your_Access_Token

The response is as follows:


```
{
  "data": [
    {
      "id": "616003678589526",
      "name": "Usc",
      "picture": {
        "data": {
          "height": 720,
          "is_silhouette": false,
          "url": "https://scontent.xx.fbcdn.net/v/t1.0-1/p720x720/16708
          "width": 720
        }
      }
    },
    {
      "id": "146546019196481",
      "name": "Uschi Burgstaller",
      "picture": {
        "data": {
          "height": 479,
          "is_silhouette": false,
          "url": "https://scontent.xx.fbcdn.net/v/t1.0-1/c0.0.479.479/1
          "width": 479
        }
      }
    },
    {
      "id": "868003976636470",
      "name": "Uschi Graf",
      "picture": {
        "data": {
          "height": 362,
```


**Figure 25**

## 4.2 Pages Facebook API

The API to query all of the Pages is:

https://graph.facebook.com/v2.8/search?q=The_Keyword_to_be_searched&type=page&fields=id,name,picture.width(700).height(700)&access_token=Your_Access_Token

The response is as follows:

```
{
  "data": [
    {
      "id": "141156592643493",
      "name": "USC Special Events Committee",
      "picture": {
        "data": {
          "height": 720,
          "is_silhouette": false,
          "url": "https://scontent.xx.fbcdn.net/v/t1.0-1/p720x720/11143458_887615124
          "width": 720
        }
      }
    },
    {
      "id": "134972803193847",
      "name": "University of Southern California",
      "picture": {
        "data": {
          "height": 720,
          "is_silhouette": false,
          "url": "https://scontent.xx.fbcdn.net/v/t31.0-1/c104.117.979.979/s720x720/
          "width": 720
        }
      }
    },
    {
      "id": "1747339718826918",
      "name": "AIS USC",
      "picture": {
        "data": {
          "height": 720,
```


**Figure 26**

## 4.3 Events Facebook API

The API to query the pages is:

`https://graph.facebook.com/v2.8/search?q=The_Keyword_to_be_searched&type=event&fields=id,name,picture.width(700).height(700)&access_token=Your_Access_Token`

The response is as follows:


```
{
  "data": [
    {
      "id": "1851364488410335",
      "name": "USC Storms La Hacks",
      "picture": {
        "data": {
          "height": 126,
          "is_silhouette": true,
          "url": "https://scontent.xx.fbcdn.net/v/t1.0-1/c37.0.126.126/255
          "width": 126
        }
      }
    },
    {
      "id": "170876643403120",
      "name": "USC Storms Hacktech",
      "picture": {
        "data": {
          "height": 126,
          "is_silhouette": true,
          "url": "https://scontent.xx.fbcdn.net/v/t1.0-1/c37.0.126.126/255
          "width": 126
        }
      }
    },
    {
      "id": "1274387602644787",
      "name": "USC Korean Culture Night Presents : Jaebi Shikdang",
```


**Figure 27**

## 4.4 Places Facebook API

The API to query the places is:

```
https://graph.facebook.com/v2.8/search?q=The_Keyword_to_be_searched&type=place&fields=id,name,picture.width(700).height(700)&access_token=Your_Access_Token
```

The response is as follows:


```
{
  "data": [
    {
      "id": "134972803193847",
      "name": "University of Southern California",
      "picture": {
        "data": {
          "height": 720,
          "is_silhouette": false,
          "url": "https://scontent.xx.fbcdn.net/v/t31.0-1/c104.117
          "width": 720
        }
      }
    },
    {
      "id": "113219012202328",
      "name": "Keck Medicine of USC",
      "picture": {
        "data": {
          "height": 720,
          "is_silhouette": false,
          "url": "https://scontent.xx.fbcdn.net/v/t31.0-1/p720x720,
          "width": 720
        }
      }
    },
    {
      "id": "45313835647",
      "name": "USC Viterbi School of Engineering",
      "picture": {
        "data": {
          "height": 720,
```


**Figure 28**

## 4.5 Groups Facebook API

The API to query the groups is:

```
https://graph.facebook.com/v2.8/search?q=The_Keyword_to_be_searched&type=group&fields=id,name,picture.width(700).height(700)&access_token=Your_Access_Token
```

The response is as follows:

**Figure 29**

## 4.5 Albums and Posts for each object

The API to query the albums and posts for an object is:

```
https://graph.facebook.com/v2.8/id? fields= albums.limit(5){name,photos.limit(2){name,
picture}},posts.limit(5){created_time}&access_token=Your_Access_Token
```

Where ‘id’ is the id of the object.
The response is as follows:

```
{
  "id": "141156592643493",
  "name": "USC Special Events Committee",
  "picture": {
    "data": {
      "height": 720,
      "is_silhouette": false,
      "url": "https://scontent.xx.fbcdn.net/v/t1.0-1/p720
      "width": 720
    }
  },
  "albums": {
    "data": [
      {
        "name": "The Ultimate Throwback 2017",
        "photos": {
          "data": [
            {
              "picture": "https://scontent.xx.fbcdn.net/v
              "id": "1276278475797960"
            },
            {
              "picture": "https://scontent.xx.fbcdn.net/v
              "id": "1276279089131232"
            }
          ],
          "paging": {
            "cursors": {
              "before": "MTI3NjI3ODQ3NTc5Nzk2MAZDZD",
              "after": "MTI3NjI3OTA4OTEzMTIzMgZDZD"
            },
            "next": "https://graph.facebook.com/v2.8/1276
          }
```


**Figure 30**

# 5. Hints

## 5.1 Images

The images for this homework have been provide in http://cs-server.usc.edu:45678/hw/hw8/images/

## 5.2 Get started with the Bootstrap Library

To know how to get started with Bootstrap, please refer to the page at http://getbootstrap.com/getting-started/. You need to import the necessary CSS file and JS file provided by Bootstrap.

## 5.3 Bootstrap UI Components

Bootstrap provides a complete mechanism to make Web pages responsive to different mobile devices. In this exercise you will get hands-on experience with responsive design using the Bootstrap Grid System.

At a minimum, you will need to use Bootstrap Form, Tab, Wells, Progress bar and Glyphicons to implement the required functionality. Information about these components can be found here:

| | |
|---|---|
| Bootstrap Form | http://getbootstrap.com/css/ - forms |
| Bootstrap Tabs | http://getbootstrap.com/javascript/ - tabs |
| Bootstrap Wells | http://getbootstrap.com/components/ - wells |
| Bootstrap Progress bar | http://getbootstrap.com/components/#progress |
| Bootstrap Glyphicons | http://getbootstrap.com/components/ - glyphicons |

## 5.4 Google App Engine/Amazon Web Services

You should use the domain name of the Google App Engine/Amazon Web Services you created in HW#7 to make the request. For example, if you're GAE/AWS server domain is called example.appspot.com or example.elasticbeanstalk.com, and the user performs a GET request with operation name="Users", then a query of the following type will be generated:

(GAE) - http://example.appspot.com/?operation=Users

(AWS) - http://example.elasticbeanstalk.com/?operation= Users

## 5.5 AJAX call

You can send the request to the PHP script by passing the URL to `$.ajax()`. You must use a **GET** method to request the resource since you are required to provide this link to your homework list to let graders check whether the PHP code is running on Google GAE or AWS (please refer to the grading guideline for details).

The is the JQuery AJAX call:

```
$.ajax({
  url: 'URL in GAE',
  //parameter list
  data: {            },
  type: 'GET',
  success: function(response,status,xhr){
      //parse the output
  },
  error: function(xhr, status, error){
      //parse the error
  }
});
```

**Figure 25 – JQuery AJAX call**

The is the AngularJS AJAX call:

```
// Simple GET request example:
$http({
  method: 'GET',
  url: '/someUrl'
}).then(function successCallback(response) {
    // this callback will be called asynchronously
    // when the response is available
  }, function errorCallback(response) {
    // called asynchronously if an error occurs
    // or server returns response with an error status.
  });
```

**Figure 26 – AngularJS AJAX call**

## 5.6. HTML5 Local Storage

Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance. Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server. There are two method, getItem() and setItem(), that can be used for this purpose. The local storage could only store strings. So you need to convert the data to strings before storing it in the local storage. See:

https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

http://www.w3schools.com/html/html5_webstorage.asp

## 5.7 Get Started with AngularJS

Use of Angular JS is must.

AngularJS extends HTML with new attributes.

AngularJS is perfect for Single Page Applications (SPAs).

AngularJS is easy to learn.

To get started with AngularJS. The W3SCHOOL site has easy tutorials:

http://www.w3schools.com/angular/

Or, you can follow the tutorials at the official website:

https://docs.angularjs.org/tutorial

## 5.8 Posting to Facebook

In order to post to Facebook, you need to use the SDK methods built for JavaScript.

One of the best ways to do this is using Angular JS as follows.

```
FB.ui({
        app_id: Your_App_ID,
        method: 'feed',
        link: window.location.href,
        picture: Url_of_the_Picture,
        name: Name_of_the_Object,
        caption: The_Caption,
    }, function(response){
    if (response && !response.error_message)
          Success
    else
          Failed
});
```

References:

https://developers.facebook.com/docs/sharing/reference/share-dialog

## 5.10 External Libraries

This is the list of external libraries that you may also want to use:

- JQuery – https://code.jquery.com/
- Moment JS -http://momentjs.com/ for time conversion

# 6. FAQ's

*Q1. Which server should I use, where would server be hosted*?

You can use either host it in Google App Engine or Amazon Web Services. While grading we just need to see the data. You should not use the cs-server for this purpose.

*Q2. Can I use any other way to store data?*

You are allowed only to use browsers local storage. No other way of storing data like cookies, session storage is allowed.

*Q3. How long should the local storage be persistent?*

The local storage data should be persistent even the browser is closed. Even if the page is refreshed, the local storage data should not be deleted.

*Q4. Facing issues with Cross Origin Request?*

There are several ways to avoid cross origin issues. You can use CORS. If you like to use $.ajax() you can use datatype: jsonp with callback function appended with the URL.

*Q5. If there is a scroll issue in iPhone or iPad?*

Use the following style to the parent or body tag

```
-webkit-overflow-scrolling: touch;
```

*Q6. Is it mandatory to use Facebook SDK?*

For homework 8, you can use the SDK or call the REST APIs directly.

*Q7. What browser is used for the purpose of testing?*

The testing is done only on latest version of Firefox as there is a problem with the navigator object in chrome.

# 7. Files to Submit

In your course homework page, you should update the HW8 link to refer to your new initial web page for this exercise. You will likely create 4 files: an HTML, a CSS, a JavaScript and a “pure” PHP file. Additionally, you need to provide a link to your homework page, and a link to the GAE/AWS server where the AJAX call is made with a sample parameter value. Also, use ‘submit’ to provide your files electronically to the csci571 account so that they can be graded and compared to all other students’ code using MOSS. You may have to copy the PHP file hosted in the cloud to cs-server, to submit it with the rest of your files.

****IMPORTANT**:**

All discussions and explanations in Piazza related to this homework are part of the homework description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.