2702ICT Intelligent Media Systems – Assignment 2 Web Mashup

Individual Assignment (50%) **DUE:** Monday 9am Week 12

Create a web application that combines information or functionalities from two or more web APIs in a useful and sensible way. This is called a Web Mashup.

You are free to decide on the **purpose** of your application, and which Web APIs you use. However, the Web APIs you have selected need to work together to fulfil the purpose of your application.

An example:
An intelligent Flickr photo browser that displays photos related to user's interest. This can be done by fetching user's "like" from Facebook web API, and then use this information to retrieve Flickr photos (using Flickr search) related to user's like. (Note: A good implementation of this application would be at a pass level)

In a more sophisticated implementation, an analysis of user's Facebook posts could be done to determine user's interest. You could implement your own analysis or use text analysis web API.

You are not restricted to using Flickr and Facebook APIs. In fact, students aiming for more than just a pass grade are expected to come up with creative/innovative and new ideas and demonstrate that they are able to learn and use Web APIs that have not been covered in detail in this course.

Further Requirements
The following are the basic requirements that all students should be able to implement:

- There is a description of your web application. This description should include:
    - A clear description of the functionality of your app, i.e. what is its purpose and/or what does it do.
    - Which Web APIs it accesses, and what kind of data is retrieved, or functionality used.
    - Which Web API require user authentication, and what extra permissions are required (if any).

  This description should be appropriately placed within your application and users should be able to see the description without having to perform user authentication.
- CSS animation should be used in at least one element in your web application. The amination should be consistent with the overall theme/functionality of this web application.
- One of the Web APIs used should require user authentication (oAuth). So your application should allow user to login and logout. The basic requirement is to use Facebook's Login Button. But more marks will be awarded for using SDK (basic to intermediate) or implement your own oAuth (hard).
- The main data for your application retrieved from Web API should be at least as complex as an array of objects. E.g. An array of likes, or array of photos.
- Your code should be appropriately structured into MVC. A bundler, i.e. webpack, should be used to bundle all the modules together.
- Templating (e.g. Handlebars.js or JSX etc) should be used to handle the display of dynamic content (and structure your code).

The following are intermediate to hard requirements. Some of these requirements requires independent learning and are designed to challenge high achieving students:

- Perform multiple fetches by using fetch with Promise.all(). Partial marks will be given for using fetch, without Promise.all().
- A simple proxy should be implemented to relay data from a Web API. The proxy should be able to also obtain input from user, and relay user input to the Web API. Partial marks will be given for using external proxy.
- Use the Javascript Framework React to implement the functionalities of your application.

UI Design

The look and feel of your web application should fit the theme of your application. You should create and implement your design to impress your client (the marker). In addition, attention to detail is expected. In coming up with your design, you should apply the design principles covered in the lecture.

Note: You should not use an existing CSS framework (such as Bootstrap and Foundation) to implement your layout and design.

Documentations

Write no more than ONE page of text containing the following:

1. **Design documentation**: You are required to describe *three* design principles you have focused on in you design. You should describe how your design is reflected on your implementation. To do this you should provide screen shots of your application in this page and refer to the screen shots in your description. Note: if you could not implement your design, you can include a sketch of your design instead screen shots.
2. **Technical documentation**: Describe which Web APIs you have used in your application. Describe the most difficult technical challenge that you had to overcome in implementing your application.
3. **Reflection documentation**: Reflect on the process you have applied to develop your solution. E.g. What improvements from assignment 1 have you made? How did you get started, did you do any planning? How often did you test your code? How did you solve the problems you come across? What do you think you could have done better for this assignment?

Video

Create a video (with audio) to demonstrate the functionality of your application. In the video, you need state your name and student number, and then explain what your application does, and demonstrate each of the features. You do not need to discuss your code in this video (we will ask you to do this during in class demonstration). Your video should be no longer than 3 minutes.

For further details of the requirements, refer to the marking rubric. **All requirements from both the assignment specification and marking rubric must be satisfied.**

Submission Requirements

Your submission should consist of:

1. A compressed file (.zip) containing your application (including all source code). See Week 1 lecture regarding how to zip a directory and download file.
2. A PDF file containing your documentations.
3. A link/URL to your demonstration video. To do this, when submitting your assignment, click on 'Write Submission' and paste the link.

These files and link must be submitted via Learning@Griffith through the Assignment 2 link.

Note: to reduce the size of your submission, you can remove the *node_modules* directory (before you download from Elf). This directory and its content can be easily restored by running the command: *npm install*

Assignment Demonstration and Marking

After you have completed your peer assessment, you must demonstrate and explain your work to your tutor in Week 12 lab to have your submission marked by your tutor.

If you do not *demonstrate your assignment* to your tutor, your submission will be regarding as incomplete, hence you will not receive a mark for this assessment item!

During the demonstration, you need to show the last modified date of your file (by running the command: *ls -la* in your Javascript directory).

Warning: We take student academic misconduct very seriously.