Getting Started - gitlab.cse.unsw.edu.au

You must store all work on the assignment **as you complete it** in a repository at gitlab.cse.unsw.edu.au.
Don't panic this is easy to do and will ensure you have a complete backup of all work on your assignment and can return to its state at any stage.

And hopefully assignment 1 has left you familiar for git.

It will also allow your tutor to check you are progressing on the assignment as they can access your gitlab repository You need to add your CSE ssh key to your gitlab.cse.unsw.edu.au account. Here is how you do that:

1. First print your CSE ssh key. If you have one, this command should should work.

```
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAyNSzIDylSPAAGLzUXdw359UhO+tlN6wWpr
SBc9gu6t3IQ1rvHhPoD6wcRXnonY6ytb00GpS4XRFuhCghx2JNVkXFykJYt3XNr1xkPItM
mXr/DRIYrtxTs5sn9el3hHZIgELY8jJZpgIo303kgnF0MsB7XpqCzg7Iv6JGkv7aEoYC/M
Nr07hXE8iQjYIHDMdO9HxGI80GyMqb1hF+RSpQTNvXQvH56juu9VXt5OwJjOqSVa4SfsEI
Cqdn+3k9w8Z4EaD93Eeog3hz0RoTrme8h/sJenXydJ0w9ZOs0By4fjqKFYPsYEs1K6SHma
+kPByZM9COgKHZwOZHH1m24HOITQ== z5555555@williams
```

2. If you couldn't print an ssh key with the above command, you need to generate a new ssh key. You can do it like this (just hit return for each question).

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/import/kamen/3/z5555555/.ssh/id_
rsa):
Created directory '/import/kamen/3/z5555555/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /import/kamen/3/z5555555/.ssh/id
_rsa.
Your public key has been saved in /import/kamen/3/z5555555/.ssh/id_rs
a.pub.
The key fingerprint is:
b8:02:31:8b:bf:f5:56:fa:b0:1c:36:89:ad:e1:cb:ad z5555555@williams
The key's randomart image is:
...
```

3. Now add your ssh key to gitlab:

4. Go to https://gitlab.cse.unsw.edu.au/profile/keys/

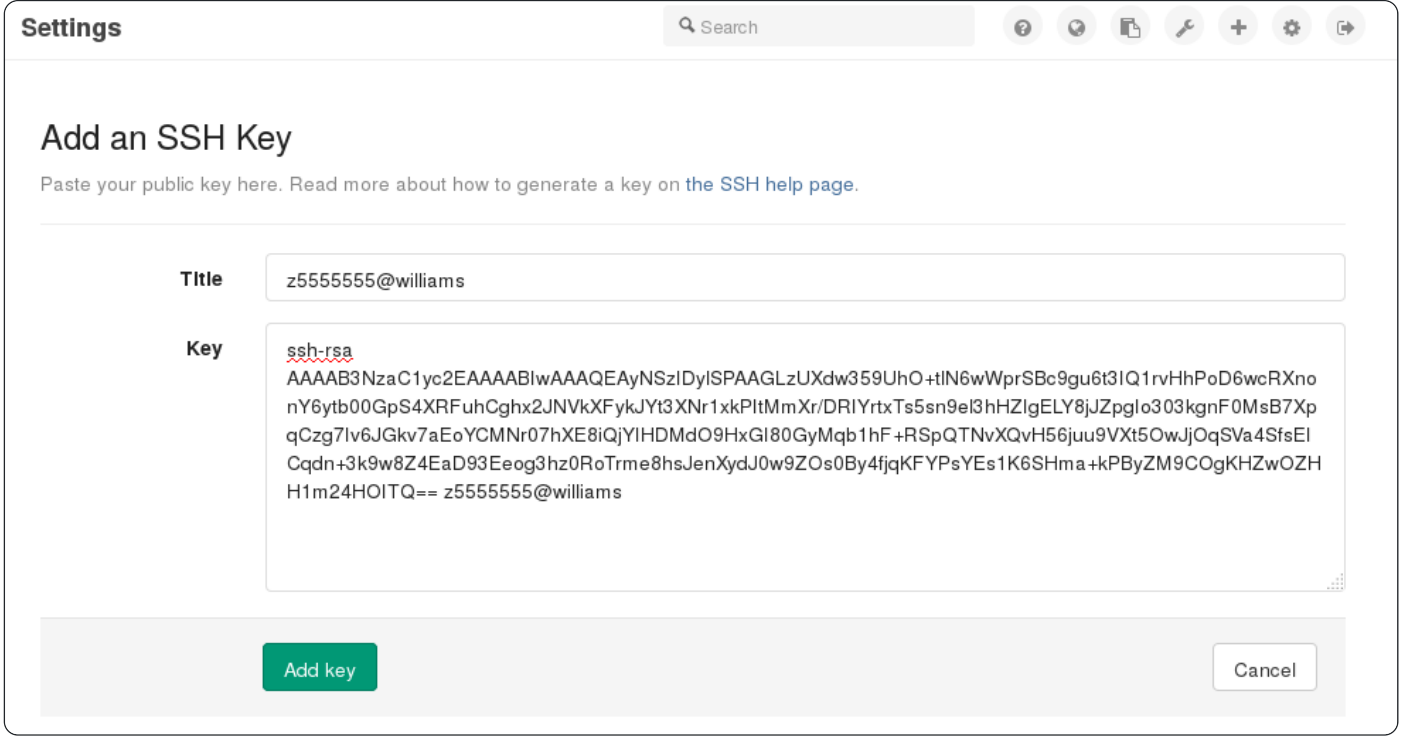
5. In the field labelled **UNSW Username** enter your zid (e.g. z5555555)

6. In the field labelled **Password** enter your zpass

7. Click on 

8. Cut-and-paste your ssh-key (the entire 200+ character line printed by `cat ~/.ssh/id_rsa.pub`) into the "**Key**" field.
   Don't cut-and paste z5555555's ssh key above - cut-and-paste your ssh-key!

9. At this point, your screen should look something like this:



10. click the green Add key button

A repository has been created for your assignment on `gitlab.cse.unsw.edu.au`.

You need to add your CSE ssh key to your `gitlab.cse.unsw.edu.au`.

After you have done that create a git repository for the assignment in your CSE account.

These commands will create a copy of the gitlab repository in your CSE account.

Make sure you replace *5555555* below by your student number!

Getting Started - Cloning the Assignment Repo

You should first clone your gitlab.cse.unsw.edu.au repository for this assignment

```
$ mkdir -m 700 -p ~/ass2
$ cd ~/ass2
$ git clone gitlab@gitlab.cse.unsw.EDU.AU:z5555555/19T2-comp2041-ass2 .
Cloning into '.'...
....
```

If the git clone above fails - redo the instructions above for adding your ssh key to gitlab.
And if that fail try with a https url instead (again replacing 5555555 with your zid) - and supplying your zid/zpass when requested:

```
$ git clone https://gitlab.cse.unsw.edu.au/z5555555/19T2-comp2041-ass2.git .
Username for 'https://gitlab.cse.unsw.edu.au': z5555555
Password for 'https://z5555555@gitlab.cse.unsw.edu.au':
```

Updating The Supplied Files

Run this command to make sure you have the latest version of the supplied files:

```
$ git remote add starting_files https://github.com/COMP2041UNSW/seddit
$ git pull starting_files master
```

Pushing Your work to gitlab.cse.unsw.edu.au

Every time you complete some work on the assignment you should commit to your git rep and push this to gitlab, for example:

```
$ vi frontend/src/main.js
$ git commit -a -m 'bugs in subset 0 fixed'
$ git push
. . . .
$ vi frontend/src/my.js
$ git add frontend/src/my.js
$ git commit -a -m 'initial version of subset 1'
$ git push
```

Seddit - Introduction

JavaScript is used increasingly to provide a native-like application experience in the web. One major avenue of this has been in the use of Single Page Applications or SPAs. SPAs are generated, rendered, and updated using JavaScript. Because SPAs don't require a user to navigate away from a page to do anything, they retain a degree of user and application state.

There are millions of websites that utilise SPAs in part of, or all of their web applications.

The assignment intends to teach students to build a simple SPA which can fetch dynamic data from a HTTP/S API. Your task will be to provide an implemention of a SPA that can provide a number of key features.

Some of the skills/concepts this assignment aims to test (and build upon):

- Simple event handling (buttons)
- Advanced Mouse Events (Swipe)
- Fetching data from an API
- Infinite scroll
- CSS Animations
- Web Workers / Service Workers
- Push Notifications (Polling)
- Offline Support
- Routing (URL fragment based routing)

API

The backend server will be where you'll be getting your data. Don't touch the code in the backend; although we've provided the source, it's meant to be a black box. Final testing will be done with our own backend so don't *assume* any single piece of data will be there. Use the instructions provided in the backend/README.md to get it started.

For the full docs on the API, start the backend server and navigate to the root (very likely to be `localhost:5000`, the exact url will be printed when you run the backend, see backend/README.md for more info). You'll see all the endpoints, descriptions and expected responses.

A Working Product

Your application should be compatible with 'modern' Chrome, Safari, and Mozilla browsers. We will assume your browser has JavaScript enabled, and supports ES6 syntax.

Writing HTML - Restrictions

You are not permitted to edit the HTML provided in *frontend/index.html* except you may change the `<head></head>` section to add your own styles and scripts.

You are not permitted to add HTML files.

You must create the HTML entirely via your JavaScript scripts. Your task is to manipulate and control your application *entirely* dynamically!

Also please use `.innerHTML` sparingly. Using `.innerHTML` amounts to the same thing as hardcoding your HTML. You'll be penalised if you rely on this method. Ask in the forum if you require more information.

You are not permitted to use Javascript written by other people.

You are not permitted to use NPM to install packages.

You are not permitted to use frameworks (React, Angular, Vue ...)

You are permitted to use small snippets of general purpose code from external sources such as Stack Overflow with proper attribution.

Getting Started

Clone the repository provided. It has a whole bunch of code, documentation, and a whole working server you'll need for developing your frontend applicaiton.

Please read the relevant docs for setup in the folders `/backend` and `/frontend` of the provided repository. Each folder outlines basic steps to get started. There are also some comments provided in the frontend source code.

Milestones

Level 0 focuses on the basic user interface and interaction with of the site. There is no need to implement any integration with the backend for this level. When demonstrating the functionality included in level 0, you are not required to prevent access to login-protected pages - all pages may be accessible by changing the URL.

Level 0

Login

```
<!--
    Before seeing the login form, a 'Log In' button
    should exist on the page. The button _must_
    have the following data attribute.
    This is important for autotests.
-->
<button data-id-login>Log In</button>
```

The site presents a login form where a user can present their credentials. Since we do not need to interact with the backend for level 0, all attempts to log in would fail. Thus, you should allow the site to inform the user that authentication has failed.

Registration

```
<!--
    Before seeing the form, a 'Sign Up' button
    should exist on the page. The button _must_
    have the following data attribute.
    This is important for autotests.
-->
<button data-id-signup>Sign Up</button>
```

An option to register for "Seddit" should be presented on the home page. This option will allow the user to create an account by entering a set of credentials (username and password) which they can then use to sign up to "Seddit". Again, since we are not required to interact with the backend for level 0, all attempts to create a new account will fail. You must perform basic input validation (think about the format of the user input you may receive) and handle cases where the input credentials should be rejected in the case of user error or (eventual) server rejection due to conflicting credential details.

Feed Interface

```
<!--
    The 'feed' _must_
    have the following id attribute.
    This is important for autotests.
-->
<ul id="feed">...</ul>
```

The application should present a "feed" of user content on the home page derived from the sample feed.json provided. The posts should be displayed in reverse chronological order (most recent posts first). You can hardcode how this works for this milestone.

Although this is not a graphic design exercise you should produce pages with a common and somewhat distinctive look-and-feel. You may find CSS useful for this.

```
<!--
    Each 'post'  _must_
    have the following data attribute.
    This is important for autotests.
-->
<li data-id-post>...</li>
```

Each post must include: 1. who the post was made by (must include `data-id-author`) 2. when it was posted 3. The image itself, if there is one present 4. How many upvotes it has (or none) (must include `data-id-upvotes`) 5. The post title (must include `data-id-title`) 6. The post description text 7. How many comments the post has 8. What suseddit this was posted to i,e `/s/meme`

While completing these tasks for level 0, consider the future inclusion of HTTP requests when designing your code - this will be helpful for future levels.

Level 1

Level 1 focuses on fetching data from the API.

Login The login form now communicates with the backend (`POST /login`) after input validation to verify whether the provided credentials are valid for an existing user. Once the user has logged in, they should see their own news feed (the home page).

NB. This is slightly different to what they will see as a non-logged in user. A non-logged in user should still see posts from `GET /post/public`.

Registration The option to register for "Seddit" (implemented in level 0) should now accept a set of credentials (a username / password pair). This user information is then POSTed to the backend to create the user in the database (`POST /signup`).

Feed Interface The content shown in the user's feed is sourced from the backend (`GET /user/feed`). Contrary to the existing popular system which "Seddit" is based off, there is only one location where all posts are to appear at this level of functionality. In the actual system called reddit posts are organised into similar groups called "sub-reddits" so all posts about plants are grouped under `r/plants` allowing plant lovers to only see those posts. seddit copies this by letting users specify the subseddit of a post when creating it so people can make posts under `s/plants` but the feed we are asking you to implement aggregates **all** posts regardless of which subseddit they were posted to.

In level 4 you will implement multiple subseddits where posts are organised into groups much like reddit.

Level 2

Level 2 focuses on a richer UX and will require some further backend interaction.

Show Upvotes Allow an option for a logged in user to see a list of all users who have upvoted a post. Possibly a modal but the design is up to you.

Show Comments Allow an option for a logged in user to see all the comments on a post. Same as above.

Upvote user generated content A logged in user can upvote a post on their feed and trigger a api request (`PUT /post/vote`) For now it's ok if the upvote doesn't show up until the page is refreshed.

In addition the user can also retract their upvote, you can do this via `DELETE /post/vote`

Post new content Logged in users can upload and post new content from a modal or seperate page via (`POST /post`). The uploaded content can either be text or text and an image.

Pagination Logged in users can page between sets of results in the feed using the position token with (`GET /user/feed`). Note users can ignore this if they properly implement Level 3's Infinite Scroll.

Profile Logged in users can see their own profile information such as username, number of posts, number of upvotes across all posts. Get this information from (`GET /user`)

Level 3

Level 3 focuses on more advanced features that will take time to implement and will involve a more rigourously designed app to execute.

Infinite Scroll Instead of pagination, users an infinitely scroll through the "subseddit" they are viewing. For infinite scroll to be properly implemented you need to progressively load posts as you scroll.

Comments Logged in users can write comments on "posts" via (`PUT /post/comment`)

Live Update If a logged in user upvotes a post or comments on a post, the posts upvotes and comments should update without requiring a page reload/refresh.

Update Profile Users can update their personal profile via (`PUT /user`) E.g: * Update email address * Update password * etc.

User Pages Let a logged in user click on a user's name/picture from a post and see a page with the users name and other info. The user should also see on this page all posts made by that person across all "subseddits". The user should be able to see their own page as well.

This can be done as a modal or as a separate page (url fragmentation can be implemented if wished.)

Follow Let a logged in user follow/unfollow another user too add/remove their posts to their feed via (`PUT user/follow`) Add a list of everyone a user follows in their profile page. Add just the count of followers / follows to everyones public user page

Delete/Update Post Let a logged in user update a post they made or delete it via (`DELETE /post`) or (`PUT /post`)

Search functionality Let a logged in user search for a post made by any user that they follow. You'll have to potentially combine a few different endpoint responses to allow this.

Level 4

This set of tasks is an extension beyond the previous levels and should only be attempted once the previous levels have been completed.

Multiple Subseddits As mentioned earlier, a subseddit is denoted by a "s/" in front of the subseddit name. Users should be able to make a post to a specific subseddit (which may or may not have been posted to before). A logged in user should also be able to view the posts made to a specific subseddit. You may choose to show this in the URL as `/s/:subseddit_name` if you wish. The original news feed - which presented the `s/all` subseddit - must still remain as the home page and must also show all posts made across all subseddits. Any post that is not posted to a specific subseddit must appear on `s/all`.

Slick UI The user interface looks good, is performant, makes logical sense, and is usable.

Push Notifications Users can receive push notifications when a user they follow posts an image. There is no endpoint or websocket provided, we expect you to figure out how to do this given the existing endpoints. Notice that since we do not give you a websocket (nor teach it in the course) we are happy for you to use polling ot achieve this, i.e check if there is anything to notify the user of every 5 or so seconds.

The delay is up to you, but remember you want it to look semi live without overwhelming the event queue.

Offline Access Users can access the "Seddit" at all times by using Service Workers to cache the page (and previous content) locally. d
Fragment based URL routing Users can access different pages using URL fragments:

```
/#profile=1
/#feed
/#profile=420
```

FAQ

1. I get a 403 (invalid auth token) on all my api requests?

Make sure you are authorized, remember you need to set your header with the token the login endpoing returned to you.

```
const options = {
    headers: {
        'Authorization' : `Token ${myToken}`
    }
}
fetch("/user", options)
```

Just for playing around you can use the `/dummy` endpoints which let you skip this step but do note you need working auth for most of the levels.

2. How do i know what a endpoint takes in and what it returns?

Run the backend server and navigate to the root, usualy 127.0.0.1:5000 or similar. There is a full set of docs which outlines every url you can hit, what method to use, a description, the paramaters it needs and the structure of the response it gives. It also outlines every possible error code it returns.

3. I'm getting a '500' error

That means that the server is crashing, this might mean your doing something really weird the backend wasn't ready to handle, in general check the forum in this case. We may have just made a oopsie.

Seddit - Frontend

Recommended Steps for tackling the assignment

1. Get something basic working with the data provided in `frontend/data`
2. Once you've got something basic working, start hitting the `/dummy/` API, which doesn't require authentication. See docs.
3. Finally, once you've got that working, transition over to use the real API.

Quickstart

To start up the frontend server which simply servers everything under the frontend folder:

```
python3 frontend_server.py
```

This will print out a username, password and launch up a static server at localhost:8080, if something else on your network is using 8080 it will start it on some other port.

```
Live at http://localhost:8080
use username 'user' and password '8tv1oz'
```

Use control-c to shut this server down.

When you visit the link you will be prompted for a username and password, use the ones printed to the console. This is to prevent other students on the same network as you being able to see your code.

In addition we've provided a basic project scaffold for you to build from. You can use everything we've given you, although there's no requirement to use anything.

```
# scaffold
data
  - feed.json  # A sample feed data object
  - users.json # A sample list of user/profile objects
  - post.json  # A sample post object

src
  - main.js   # The main entrypoint for your app
  - api.js    # Some example code for your api logic

styles
  - post.css
  - provided.css  # some sample css we've provided (add more stylesheets as you please)
```

To make sure everything is working correctly we strongly suggest you read the instructions in both backend and frontend, and try to start both servers (frontend and backend).

Seddit Backend

You are given a backend server for seddit written in Python.

Do not change any file in the backend directory.

You only submit the seddit frontend.

Any changes you make to the backend will be lost.

If you depend on changes you make to the backend your code will break.

Running the Backend @ CSE

Running the backend on a CSE machine is simple:

```
$ 2041 ass2_backend
```

Visit the url it prints (e.g. http://127.0.0.1:5000/) to see the backend docs!

Note if other students are running a sever on the same machine you will give a different URL (port).

The backend server will print out a message telling you how to run the frontend server.

This command ensures correct URL for the backend server is passed to your frontend Javascript.

Running the Backend on your Own Machine

You can use virtual env [recommended].

```
cd backend
# create a sandbox for the backend
virtualenv -p /usr/local/bin/python3 env

# enter sandbox
source env/bin/activate
# set up sandbox
pip install -r requirements.txt
# run backend! Will print out a bunch of info including a line like this:
#  * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
# visit the url on this line (e.g. http://127.0.0.1:5000/) to see the backend docs!
python backend_server.py
```

Once you are done working on the assignment run the following command to exit the sandbox

```
deactivate
```

This method creates a space in which the backend can run without clashing with any other python packages and issues on your local account. If you don't care you can run the backend in the global space as such.

```
cd backend
# on your local system this may just be pip and python not pip3 and python3
pip3 install -r requirements.txt
python3 app.py
```

User Data

in `backend/db/users.csv` there is a list of all users within the provided database, you can login as any of these users for testing or create your own account. Note that in the case that you put the database in a undesirable state such as accidently making too many accounts or comments on a post etc. simply delete the file `backend/db/test.sqlite3` and restart the backend server. The server will automatically detect the missing file and download a fresh copy.

Connecting to your A Server Running at CSE Using an SSH Tunnel

You can use ssh to forward a port from on your local machine to a port on a CSE server.
You can use this to run your browser on your local machine and your frontend & backend servers at CSE.

Alternatively you could run your backend server at CSE and your frontend server and your browser at CSE.

SSH Tunnelling on Linux/Mac

Its best if you can get your laptop or home machine setup so it can run frontend and backend servers.
If you can't, it is possible to connect a server running at CSE to your local machine.

First ssh to a CSE login server such as weber and start your backend:

```
$ ssh weber.cse.unsw.edu.au
$ 2041 ass2_backend
$ cd ass2/frontend
$ 2041 ass2_backend
....
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
....
```

Then ssh to the same CSE login server in another window and start your frontend:

```
$ ssh weber.cse.unsw.edu.au
$ cd ass2/frontend
$ python3 frontend_server.py http://127.0.0.1:5000
....
Frontend server running at:  http://localhost:8080
....
```

Then in a third window create a ssh tunnel again to the same CSE login server - replace 8080 with the actual port your frontend server is running.

```
$ ssh -L 2041:localhost:8080 weber.cse.unsw.edu.au
```

This connects port 2041 on your localmachine to port 8080 on the CSE login server.
You should now be able to access your frontend server at the URL **http://localhost:2041** on your local machine.

Alternatively you could after starting the backend server run a command like this to connect port 5000 on your local machine to port 5000 on the CSE server, and then run the frontend server on your local machine. a

```
$ ssh -L 5000:localhost:5000 weber.cse.unsw.edu.au
```

SSH Tunnelling on Windows

This guide to creating ssh tunnels on Windows written by a COMP2041 tutor should help you create a SSH tunnel from Windows, if you have Putty installed.

Assumptions/Clarifications

It is a requirement of the assignment that when you work on the assignment for more than a few minutes you push the work to `gitlab.cse.unsw.edu` (see above).
You must implement this assignment in Javascript.

You may use small amounts (< 10 lines) of general purpose code (not specific to the assignment) obtained from a site such as Stack Overflow or other publically available resources. You should attribute clearly the source of this code in a comment with it.

You can not otherwise use Javascript, CSS or HTML written by another person.

Do not use NPM to install packages.

You can not, for example, use Javascript frameworks such as Angular or React.

You are not permitted to use CSS from external sources, e.g. you are not permitted to use Bootstrap. This is an individual assignment. The work you submit must be your own work and only your work apart from the exception above. Joint work is not permitted.

You should follow discussion about the assignment in the course forum. All questions about the assignment should be posted there unless they concern your private circumstances. This allows all students to see all answers to questions.

Diary

You must keep a record of your work on this assignment as you did for assignment in an ASCII file The entries should include date, starting & finishing time, and a brief (one or two line) description of the work carried out. For example:

Date	Start	Stop	Activity	Comments
15/10	16:00	17:30	coding	implemented creation of accounts
20/10	20:00	10:30	debugging	found bug in handling of posts

Include these notes in the files you submit as an ASCII file named diary.txt.
Some students choose to store this information in git commit messages and use a script to generate `diary.txt` from `git log` before they submit. You are welcome to do this.

Attribution of Work

This is an individual assignment. The work you submit must be your own work and only your work apart from any exceptions explicitly included in the assignment specification above.
Joint work is not permitted.

You are only permitted to request help with the assignment in the course forum, help sessions or from course lecturers or tutors.

Do not provide or show your assignment work to any other person (including by posting it on the forum) apart from the teaching staff of COMP(2041|9044).

The work you submit must otherwise be entirely your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined both automatically and manually for such submissions.

We are required to inform scholarship authorities if students holding scholarships are involved in an incident of plagiarism or other misconduct, and this may result in a loss of the scholarship.

Plagiarism or other misconduct can also result in loss of student visas.

If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted you may be penalized, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.

Note, you will not be penalized if your work is taken without your consent or knowledge.

Submission of Work

give doesn't allow submission of a directory hierarchy, so you'll need to submit your work as a tar file. You can do that like this:

```
$ tar -Jcf frontend.tar frontend
$ give cs2041 ass2_seddit diary.txt frontend.tar
```

You can't submit the backend because you are not permitted to change the backend. Your frontend will be run with the backend you have been given.
Unlike the last assignment you don't need to run give everytime you work on the assignment, you do need to run **git push** very time your work on the assignment, so your repo on **gitlab.cse.unsw.edu.au** is updated.

You may still run give multiple times, only the final submitted version of your assignment will be marked.

Assessment

Assignment 2 will contribute 15 marks to your final mark.
Assignment 2 marking occurs in peer assessment sessions. Details of the peer assessment sessions will be announced in week 10. Your must attend one peer assessment sessions.

80% of the marks for assignment 2 will come from your code against a specified set of operations and assessed by fellow students. You will be present to assist in determining what features are and are not working, you will also be able to indicate any extra features you have implemented.

20% of the marks for assignment 2 will be awarded on the basis of clarity, commenting, elegance and style of your code. In other words, your fellow students will assess how easy it is for a human to read and understand your program.

Here is an indicative marking scheme .

100%	Elegant Javascript with an excellent implementation of levels 0-4) features
HD++%	Very well written Javascript which implements levels 0-3 successfully
HD (85+)	Well written Javascript which implements most of levels 0-3 successfully

DN (75+)	Readable Javascript which implements of levels 0-2 successfully
CR (65+)	Reasonably readable code which implements level 0-1 successfully
PS (55+)	Reasonably readable code which implements level 0 successfully
45%	Major progress on the assignment with some things working/almost working
-70%	Knowingly supplying work to any other person which is subsequently submitted by another student.
0 FL for COMP2041/9041	Submitting any other person's work with their consent. This includes joint work.
academic misconduct	Submitting another person's work without their consent.

The lecturer may vary the assessment scheme after inspecting the assignment submissions but its likely to be broadly similar to the above.

Due Date

This assignment is due Tuesday August 13 21:59
If your assignment is submitted after this date, each hour it is late reduces the maximum mark it can achieve by 4% per hour for 25 hours For example if an assignment worth 84% was submitted 3 hours late, the late submission would have no effect. If the same assignment was submitted 6 hours late it would be awarded 76%, the maximum mark it can achieve at that time.