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 an 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
- 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. 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 (`localhost:5000`). You'll see all the endpoints, descriptions and expected responses.

A Working Product

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

Getting Started

Clone the gitlab repository using the instructions below. It has a code, full documentation, and a working server you'll need for developing your frontend application.

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 building of the site. There is no need to implement any integration with the backend for this level.

Level 0

Login The site presents a login form and a user can log in with pre-defined hard coded credentials. You can use the provided users.json so you can create a internal non persistent list of users that you check against.

Once logged in, the user is presented with the home page which for now can be a blank page with a simple "Not Yet implemented" message.

Registration An option to register for "Instacram" is presented on the login page allowing the user to sign up to the service. This for now updates the internal state object described above.

Feed Interface

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 include:

1. who the post was made by
2. when it was posted
3. The image itself
4. How many likes it has (or none)
5. The post description text
6. How many comments the post has

Level 1

Level 1 focuses on fetching data from the API.

Login The site presents a login form and verifies the provided credentials with the backend (`POST /login`). Once logged in, the user can see the home page.

Registration An option to register for "Instacram" is presented allowing the user to sign up to the service. The user information is 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`)

Level 2

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

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

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

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

Post new content Users can upload and post new content from a modal or seperate page via (`POST /post`)

Pagination 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 Users can see their own profile information such as username, number of posts, sum of likes they received on all their posts, etc. You may choose to utilise the information from the api in more creative ways such as displaying their most liked post etc. 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 rigorously designed app to execute.

Infinite Scroll Instead of pagination, users can infinitely scroll through results. For infinite scroll to be properly implemented you need to progressively load posts as you scroll.

Comments Users can write comments on "posts" via (`POST post/comment`)

Live Update If a user likes a post or comments on a post, the posts likes 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
- Update name

User Pages Let a 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. The user should be able to see their own page as well.

This can be done as a modal or as a seperate page (URL fragmentation can be implemented if desired.)

Follow Let a user follow/unfollow another user to 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

To allow users to find other users implement some functionality to check if a username exists and allow the user to follow them. This may be as simple as a text box in the navbar which signals when a inputted username is wrong or if it's right gives you a option to follow the user. You do not need to implement anything more advanced such as searching etc.

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

Level 4

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.

Offline Access Users can access the "Instacram" at all times by using Web Workers to cache the page (and previous content) locally.

Fragment based URL routing Users can access different pages using URL fragments:

```
/#profile=me
/#feed
/#profile=janecitizen
```

Recording Assignment work on gitlab.cse.unsw.edu.au

It is a requirement of this assignment you 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.

It will also allow your tutor to check you are progressing on the assignment as they can access your gitlab repository

Adding Your SSH Key to Gitlab

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/

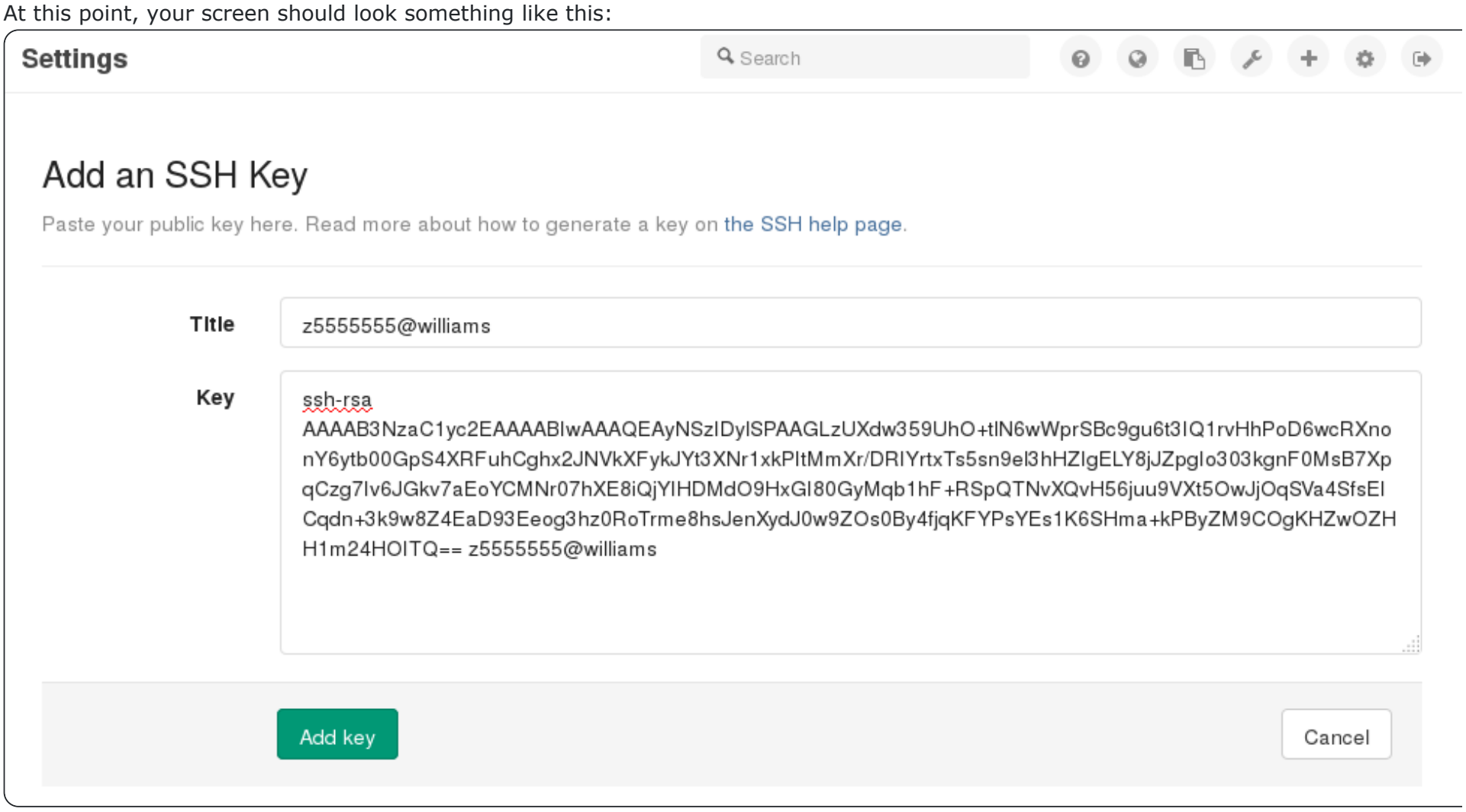
5. In the field labelled **UNSW Username** enter your zid (e.g. z5555555)

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

7. Click on Sign in

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!

Setting Up Your Own Git Repository

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

```
$ mkdir -m 700 -p ~/ass2
$ cd ~/ass2
$ git clone gitlab@gitlab.cse.unsw.EDU.AU:z5555555/18s2-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/z5555555/18s2-comp2041-ass2.git .
Username for 'https://gitlab.cse.unsw.edu.au': z5555555
Password for 'https://z5555555@gitlab.cse.unsw.edu.au':
```

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

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 code written by another person.

Do not use NPM to install libraries other than as indicated in the specifcation.

You can not, for example, use the popular Javascript framework such as Angular or React.

You are permitted to use CSS from external sources as long as it properly attributed.

For example, you are permitted to use the popular Bootstrap CSS framework.

Some Bootstrap functionality relies on accompanying Javascript. You are permitted to include this Javascript.

The Javascript accompanying Bootstrap requires the popular general purpose Javascrpt library jQuery. You are permitted to include **jQuery** so bootstrap can use it. However you are not permitted to use **jQuery** in the code you write for the assignment.

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[29]041.

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 files.tar --exclude node_modules frontend
$ give cs2041 ass2_instacram diary.txt files.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. Please do *not* include the folder `node_modules` in this tar as the folder contains package information and is usually quite large.
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

Originally assignments 1 & 2 were both intended to contribute 15 marks to your final mark.
Assignment 2 has been reduced in scope because it is late being released. Given this, and the large amount of work most students did on assignment 1, instead assignment 1 will contribute 20 marks to your final mark and assignment 2 will contribute 10 marks.

However to ensure students can not be disadvantaged by this late change, we will also calculate your final mark with the original assignment weighting (15 marks each) and if this result in a better overall mark for you we will use the original assignment weighting.

Assignment 2 marking occurs in peer assessment sessions. Details of the peer assessment sessions will be announced in week 13. 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 tentatively due Saturday 5 January 19:00
If your assignment is submitted after this date, each hour it is late reduces the maximum mark it can achieve by 0.5% per hour for 24 hours, then 2% per hour after that. For example if an assignment worth 84% was submitted 20 hours late, the late submission would have no effect. If the same assignment was submitted 30 hours late it would be awarded 76%, the maximum mark it can achieve at that time.