SENG2021 Deliverable 3

Group 1
Members

Name: Kiran Gupta
Zid: z5164964
Email: k.gupta@student.unsw.edu.au

Name: Abanob Tawfik
Zid: z5075490
Email: z5075490@ad.unsw.edu.au

Name: Riley Sutton
Zid: z5164867
Email: z5164867@student.unsw.edu.au

Name: Aidan Farrell
Zid: z5162069
Email: z5162069@unsw.edu.au

Name: Michael Yoo
Zid: z5165635
Email: michael.yoo@student.unsw.edu.au

Table of Contents

Requirements Analysis

Problems Addressed By our System

Currently there is no system which allows users to browse for twitch clips based on categories and streamers. The Twitch website itself only has a few clips hand-picked by the streamer themselves to display, however all clips made by users tend to end up lost, unless the link for the clip is kept. Some problems we attempted to address such as downloading the clips for the user with the button, were unable to be implemented due to changes in twitch API.

The below problem statements, represents the final problems which were solved by the project.

- There is no large navigable and easily searchable collection of Twitch Clips.
- There is no way to access Twitch Clips without a direct link to them or the streamer's profile. In other words, there is no way to browse Twitch Clips among the entirety of Twitch.
- There is currently no specialised homepage with current Twitch clip compilation channels and videos. This will make browsing and searching for these Clips easier.
- There is no way to search for Twitch clips based on categories and the tops clips for each category.

Different Features of the System

Our developed system has many features that address the problem statement. We present to users a central hub in which viewers can easily browse for twitch clips and watch some of their favourite moments from Twitch. The features in our system include:

- A centralised homepage
    - Recommends users clips in different categories
- A categorised search system to browse via specific streamer or game
- A search bar to search for specific clips/terms
- A clip page containing
    - The embedded clip from twitch so that users can watch the clip
    - Data from the user that uploaded it
    - A rating system (like and dislike button)
    - A chat system that displays the live chat from the Twitch clip
    - A list of recommended clips similar to the one being viewed
- A login system that includes/allows
    - A signup page
    - Storage of liked and watched videos
    - An account profile
    - The ability to upload clips to the site
- An upload page that allows:
    - Users to take clip links and upload them to ClipBucket
    - Users to customise their upload by adding a description and tags
    - Users to upload an image to set their thumbnail of the clip
- A database that stores all the data within the system uploaded by the users
- A history page that keeps a list of all clips viewed by a user on the website
- A liked page that keeps a list of all the clips liked by a user on the website

updated user stories

Some of the user stories for some of the features were not included as they were not implemented in the final project. This was due to a variety of reasons, the main one being that the API had changed for twitch. And lack of understanding of how to implement certain features.

Interested parties of use cases (functional categories):

- Viewer/normal user
- Uploader

Requirement
Users require a medium to navigate through twitch clips

Feature: Search bar to search for queries

As a: Clip Viewer
So that I can: Look for specific clips based on tags
I want to: A search bar where I can search for search queries such as “Dota 2” and be shown popular and trending Twitch Clips

GIVEN: I am on any page of the website
WHEN: I click the search bar
THEN: I enter my search queries such as “Dota 2”
AND: I click the search button or press enter
THEN: After 2-3 seconds, I am redirected to a page which contains a grid view or list of the most relevant results based on how relevant it is to the query.

Below in Figure 1, is the representation of the search bar

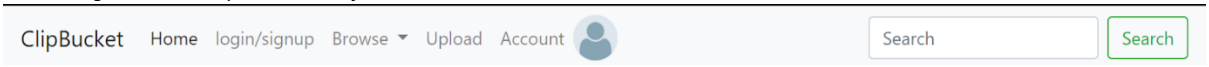


Figure 1 The menu bar with the search bar on it

Feature: Home page menu which shows relevant clips

As a: Clip Viewer
So that I can: Look for interesting and fun clips
I want to: have a home page menu which shows me relevant clips based off my view history

GIVEN: I am on any page of the website
WHEN: I click the home button
THEN: I am redirected to the home page menu
WHEN: I click on a specific clip on the home page
THEN: I am redirected to the clip page

Below in Figure 2, is the representation of the home page

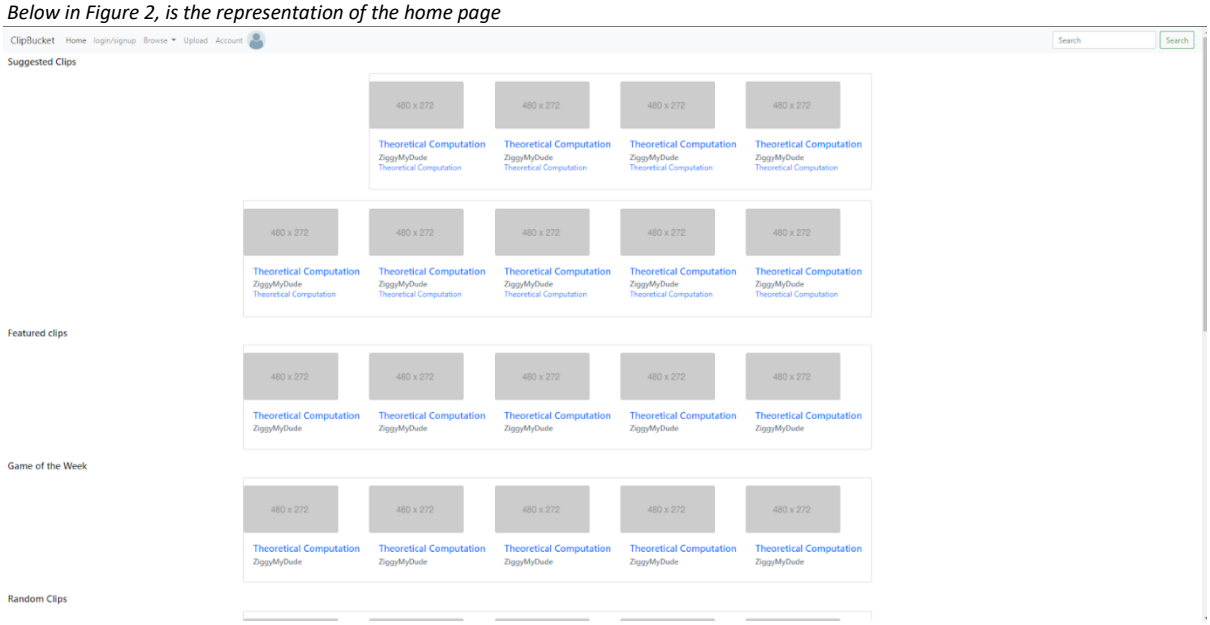


Figure 2 representation of the home page

Feature: viewing a clip
As a: Clip Viewer
So that I can: view a clip
I want to: Be shown a page with a video player that will play the Twitch Clip

GIVEN: I am on the home page, or the clip page with recommended videos
WHEN: I click on a clip I am interested in
THEN: I am redirected to a page with a video player that will play the twitch clip

Below in Figure 3, is the representation of the clip page

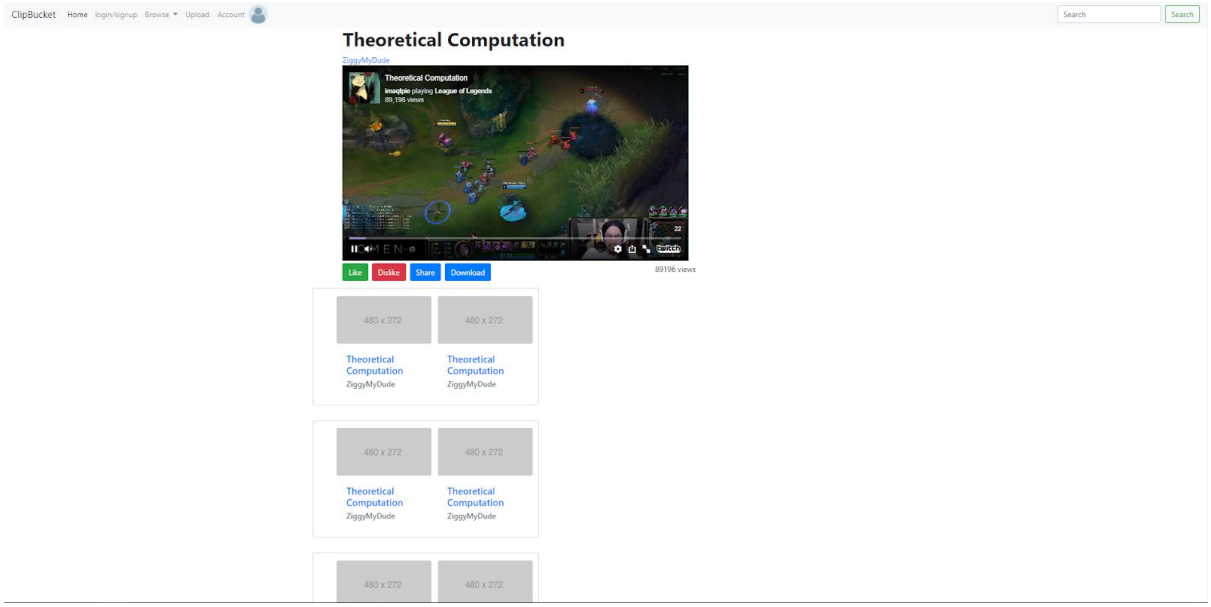


Figure 3 a representation of the clip page

Feature: Sidebar

As a: Clip Viewer
So that I can: view different clip categories
I want to: Be shown a sidebar that contains different clip categories

GIVEN: I am on any page of the website
WHEN: I click on the menu button
THEN: a side-bar pops up with multiple categories
WHEN: I click on a category
THEN: I am redirected to a page which contains all relevant clips of that category

Below in Figure 4, is the representation of the drop-down menu (side bar)

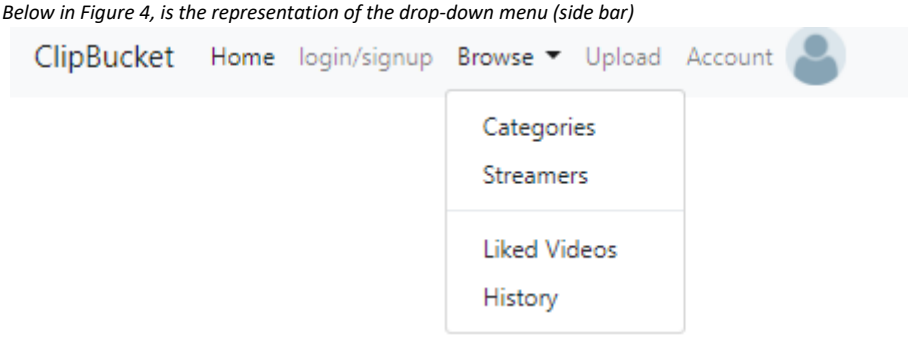


Figure 4 The representation of our side bar

Requirement:
Users require a method to upload twitch clips onto a website that will allow the clip to be described, tagged, categorised and easily accessed by other users.

Feature: Upload Page

As a: Clip Uploader
So that I can: share clips onto the website
I want to: upload the link to my twitch clip onto the website

GIVEN: I am on any page of the website
WHEN: I click on the upload button
THEN: I am redirected to the upload page
WHEN: I click on the upload link box
THEN: I enter the link for my twitch clip onto the upload link box ***AND:***
I press the upload clip button.
THEN: A progress bar will appear showing upload percentage

Below in Figure 5, is the representation of the upload page

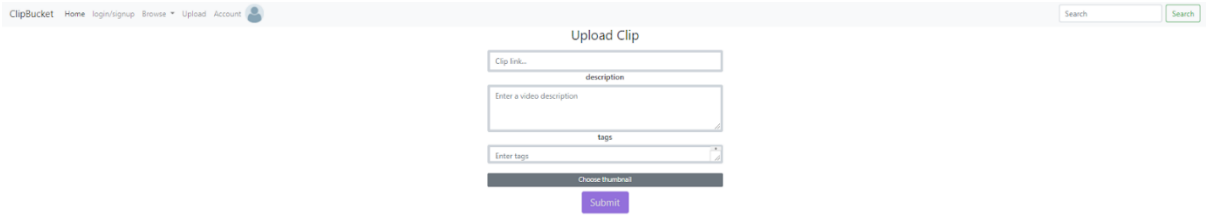


Figure 5 the representation of the uploag page

Requirement:
Twitch clips requires a method of rating to give feedback on the clip.

Feature: Like Button

As a: Clip viewer
So that I can: rate the clip showing that I enjoyed it
I want to: be able to leave quick feedback on the video

GIVEN: I am viewing any clip on the website
WHEN: I click on the like button
THEN: the amount of likes on the video is incremented by 1 (or removed if it exists) instantly
AND: the video is added to the liked videos playlist (or removed if it exists) instantly

Below in Figure 6, is the representation of the like button on any clip

Figure 6 a visual representation of the like button

Feature: Comment section

As a: Clip viewer
So that I can: let the clip submitter know more specific feedback on the video
I want to: be able to leave comments on the video

GIVEN: I am viewing any clip on the website
WHEN: I click on the comment box
THEN: I am able to give a comment describing thoughts/feedback on the clip
WHEN: I click on post
THEN: the comment will be posted onto the clip

Below in Figure 7, is the representation of the Comment Section

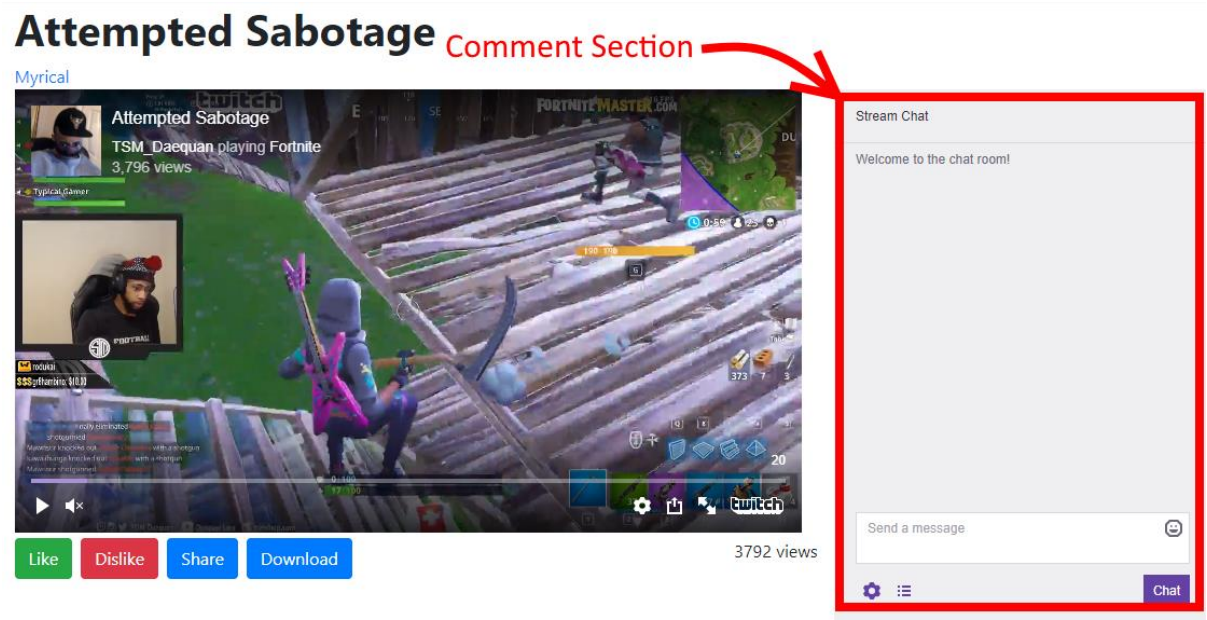


Figure 7 A visual representation of the comment section

Requirement:
Users require a method of maintaining the history of their website activity and being able to keep playlists of clips they enjoyed watching.

Feature: History Page

As a: clip viewer
So that I can: see my viewing history
I want to: be given a page that contains all watched clips

GIVEN: I have the sidebar open
WHEN: I click on the "Clip History" button
THEN: I am immediately redirected to my viewing history page containing a list or grid of watched clips in chronological order

Below in Figure 8, is the representation of the history page

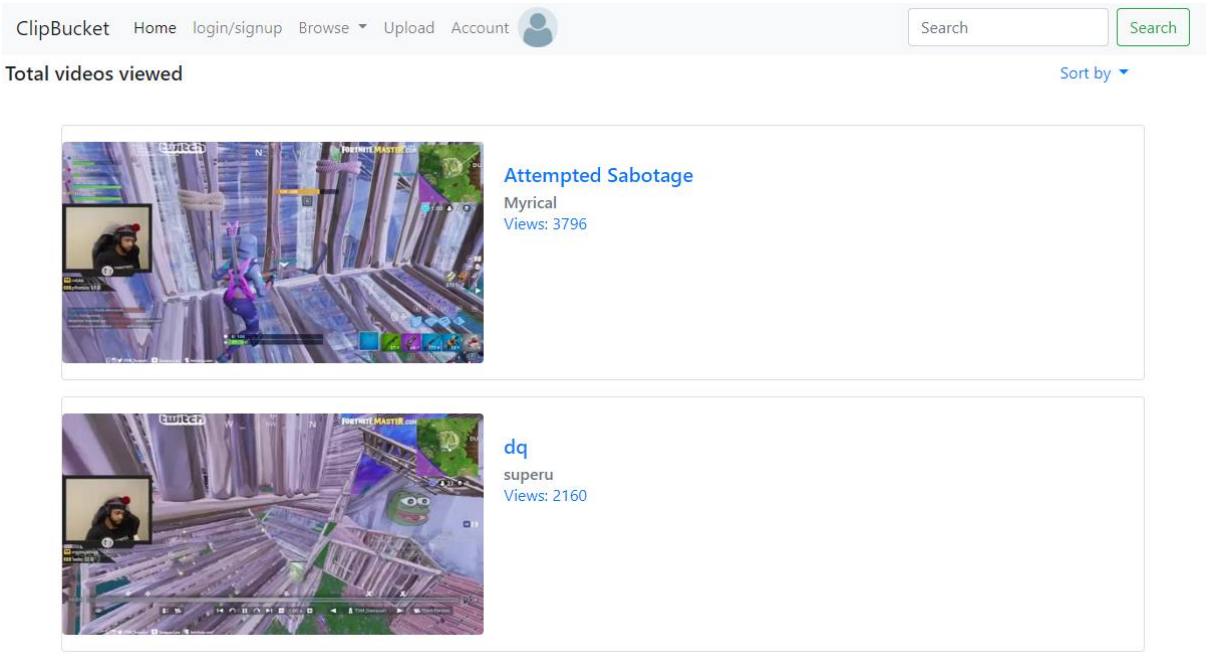


Figure 8 A visual representation of the history page

Feature: Liked Videos Page

As a: clip viewer
So that I can: see my liked clips history
I want to: be given a page that contains all my liked clips

GIVEN: I have the sidebar open
WHEN: I click on the “Liked Clips History” button
THEN: I am immediately redirected to my liked clips page containing a list or grid of clips I have liked in chronological order

Below in Figure 9, is the representation of the Liked clips page

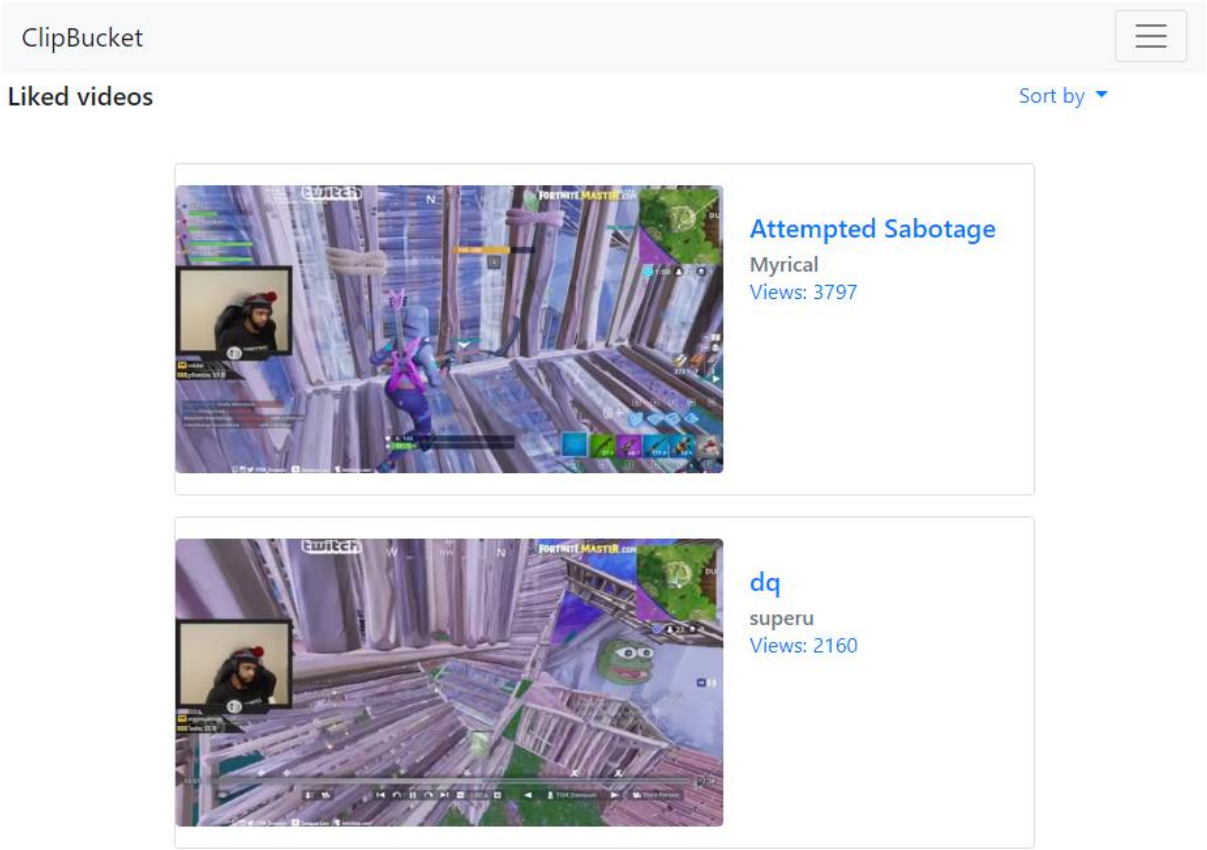


Figure 9 A visual representation of the liked clip page

Design

Final Software Architecture

Bootstrap
Flask
SQLite
Any browser with a viable connection
Figure 10 displays the application stack used in the project design.

Web Application Stack
Web Browsers Layer

Client-side Web Framework Layer
 
Application Layer (Our Application)

Server-side Web Framework Layer

Database Layer

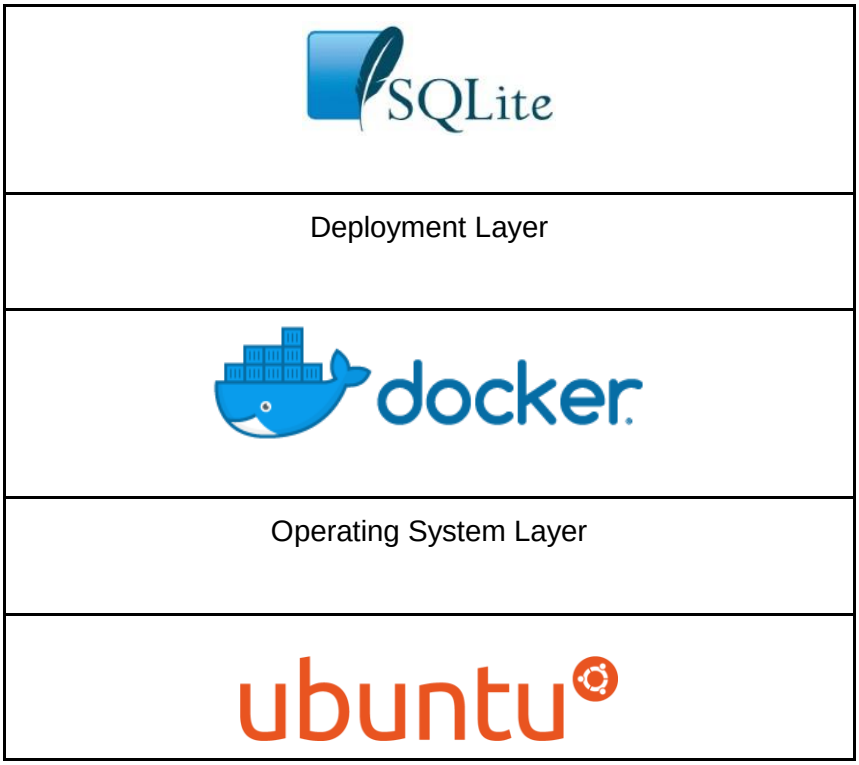
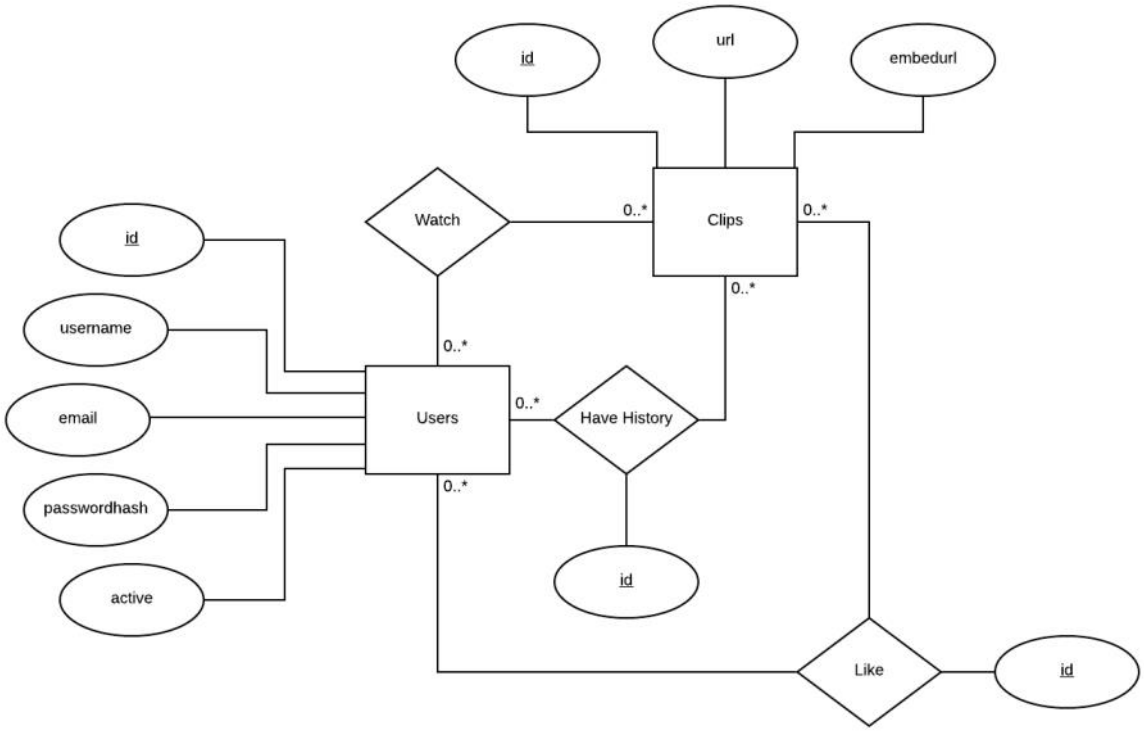


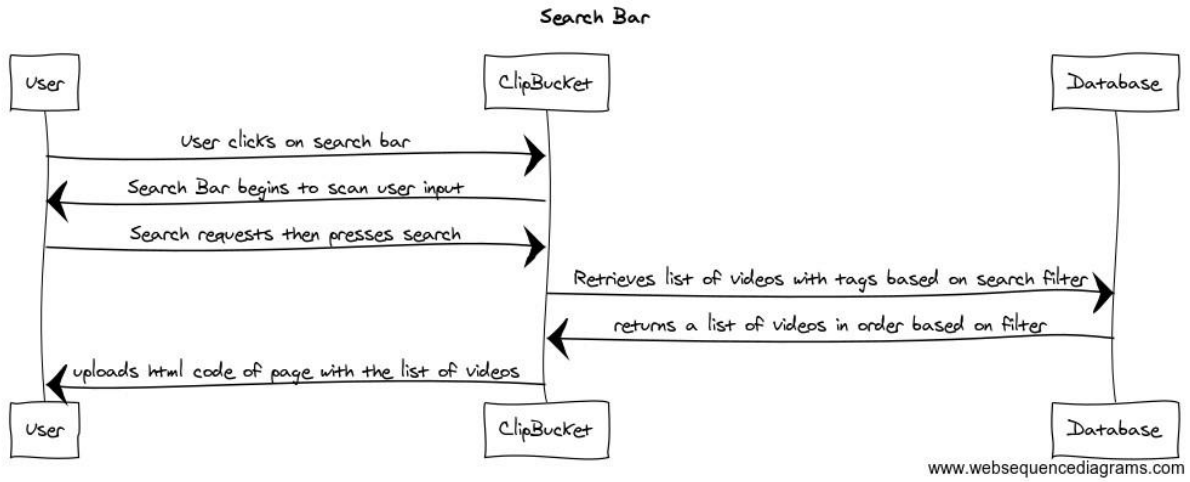
Figure 10 A layered view of our application stack.

Entity Relationship Diagram

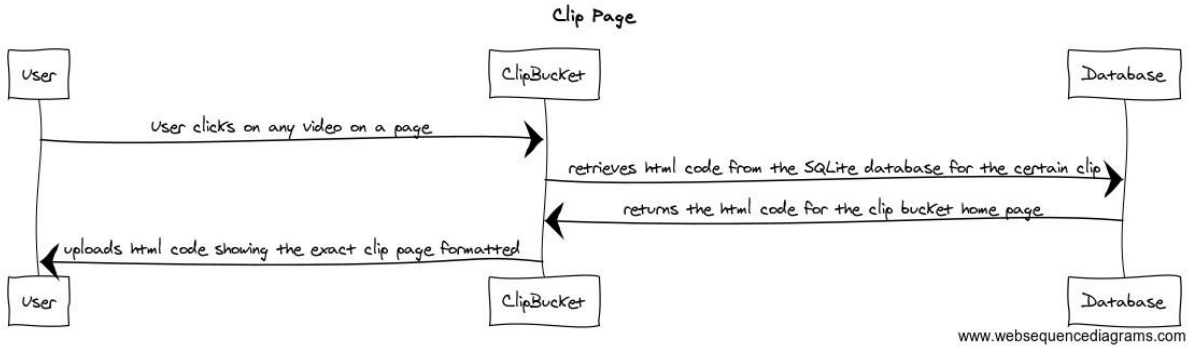


Updated UML Sequence Diagrams

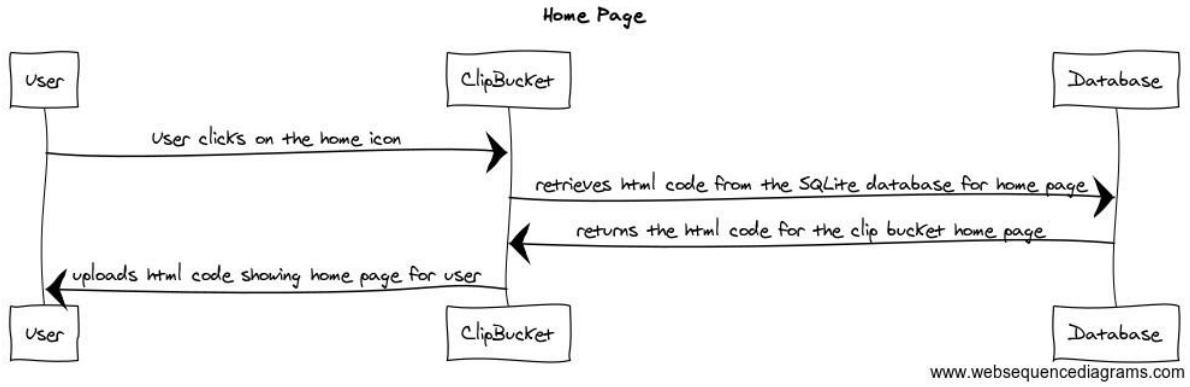
Search Bar



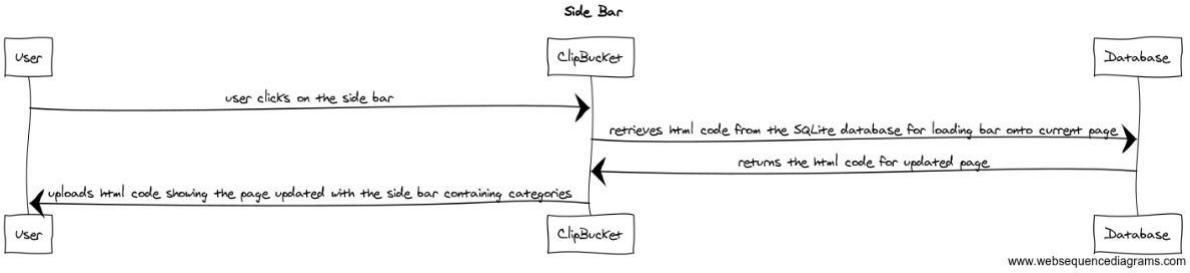
Clip Page



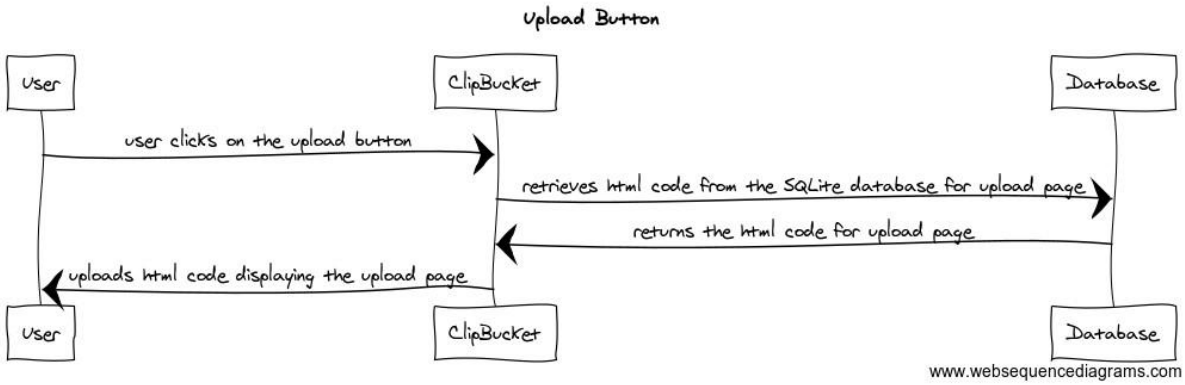
Home Page



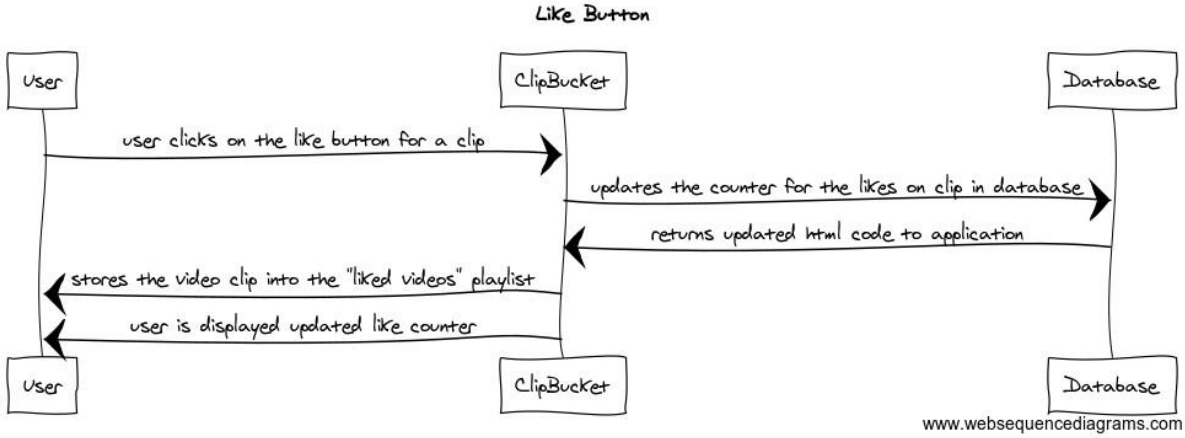
Side Bar



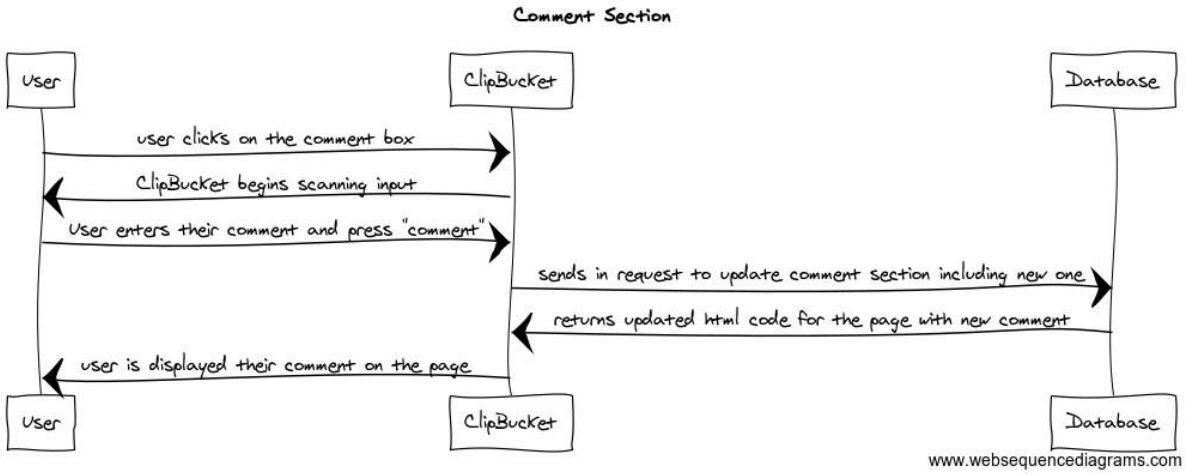
Upload Button



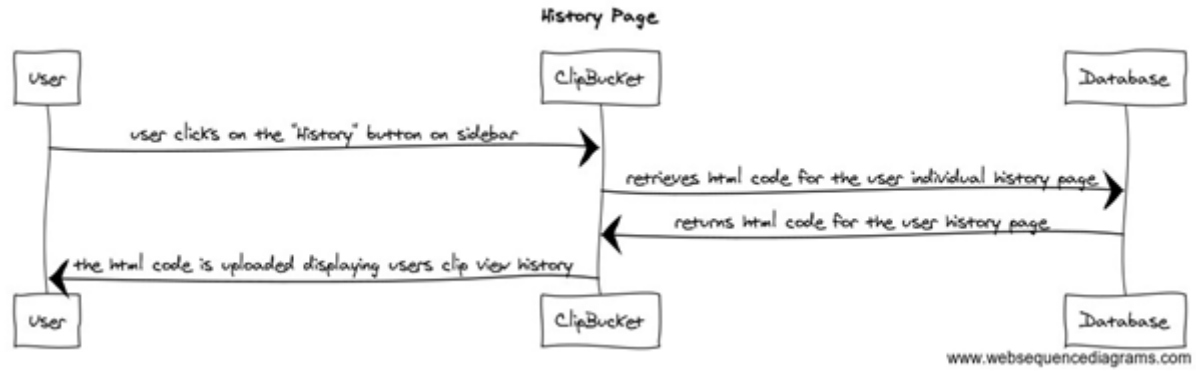
Like Button



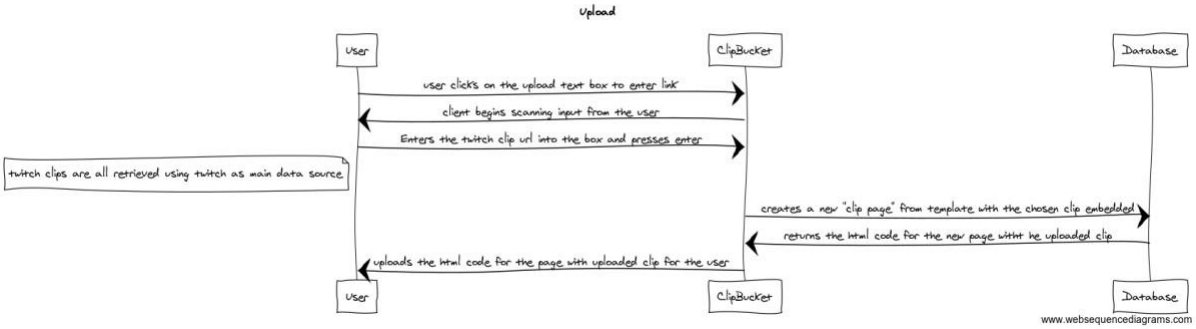
Comment Section



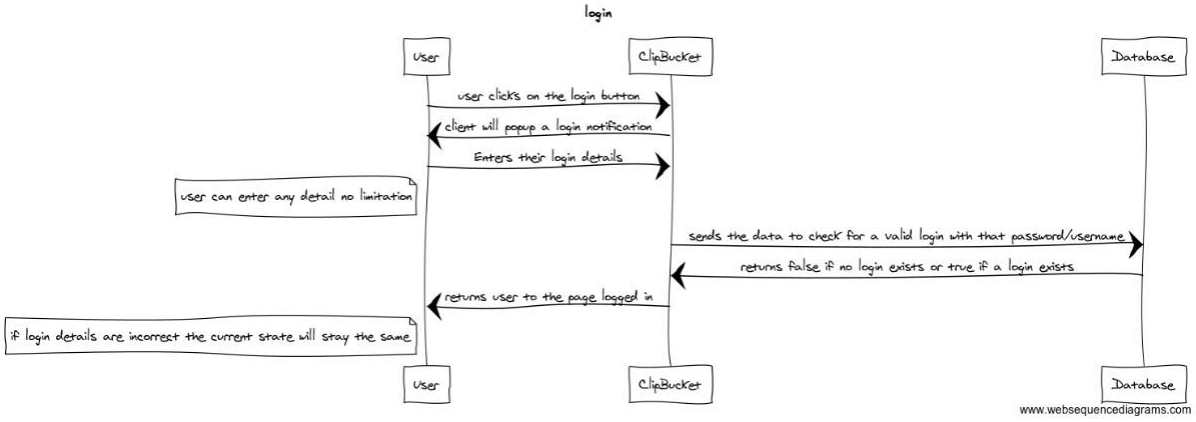
History Page



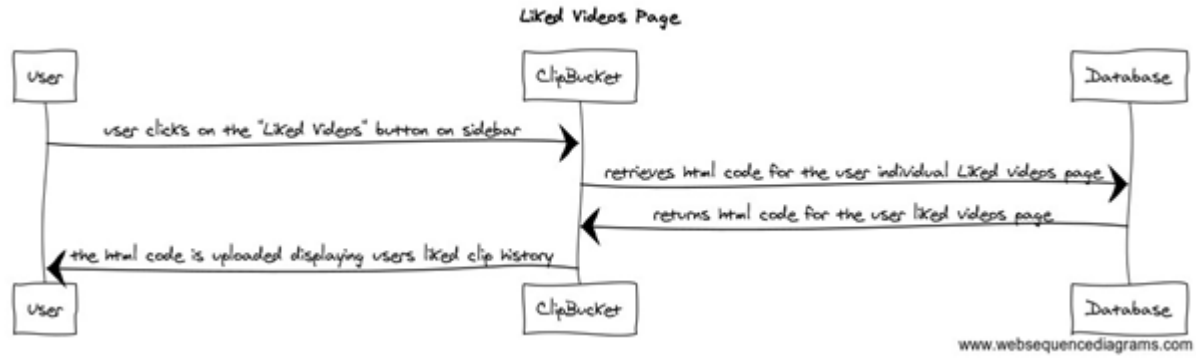
Uploading



Login



Liked videos



Key technologies used in design

Bootstrap was used in order to implement all of the front-end design, with a small amount of JavaScript to add some functionality, for example, changing the name on upload button to be the file name, or allowing reselection for upload.

Flask was used in order to implement all of the back-end work, including retrieving data from twitch API, displaying the data for the user, getting the correct details from a clip, implementing the search bar, creating the pages and other similar processes.

SQLAlchemy was used for database management. It was used to keep track of user data, clips on the website, storing the videos under the correct tags, return clips from database based on user search, and overall connecting the front end to the back end.

Summary of Key Benefits/Achievements of Design/Implementation

There were many key benefits to the choices made by the group. The following table below displays the design choice or technology, followed with the list of benefits achieved with the choice. This is displayed in the table 1

Table 1.

Key benefits and achievements of the technology choices

Technology	Benefits and Achievements
Bootstrap	• Was simple to use, allowed us to complete the front end within the framework, react would have been too difficult to use and learn. Bootstrap allowed for simplistic styling, and all pages on the project were styled using bootstrap • Simple integration with other libraries such as jQuery, allowed for a more customisable website
Python (Flask)	• The group has had experience with flask previously whilst none of the group has experience with node.js • Allowed for the minimum requirements to be implemented • Flask allowed for easy integration between all other aspects of the project
SQLite (SQLAlchemy)	• Simple database implementation, didn't require us to install and connect the server to the website • We didn't need to store multiple user data and user interaction for the scope of the project, which are features more widespread in other databases such as MYSQL. • The most simplistic deployment method as was required for the project specifications

Certain design choices came with many benefits.

- The choice to have the menu bar as a sticky menu bar on every page allows users to navigate through the website and be able to access all aspects of the project regardless of which page the user is on.
- The choice to use the twitch chat itself instead of a custom one, was one which makes the experience closer to the twitch experience.
- The choice to layout the project similar to YouTube's style was one that is able to create an easier experience for users, as most users will have familiarity with YouTube, allowing a more intuitive design.
- The choice to embed the twitch clip rather than create our own player, was one that both allowed users to watch the clip on the current website and allowed for a very simple implementation.
- The choice to use clip links for upload rather than a system where the user had to download then upload the clip, was one that allowed for not only simple integration, but also allows for a much simpler user experience and a more intuitive user experience.

All these design choices helped us achieve our minimal viable product, and even more in some aspects of the website. The overall design choices alongside the stack choices allowed for compatibility, and implementation of the project in a presentable state.

Team Organisation and Conclusion/Appraisal of Work

Responsibilities/Organisation of the Team

For project development our group split up into 3 teams. Riley and Abanob worked on the front end, making sure that the html, bootstrap and JavaScript worked with the routes of the program. Michael worked on the login system, making sure that flask-login was working correctly and securely. Aiden and Kiran worked on the database, making sure that the data inputted by users was stored correctly into the SQLite database and that there were functions available to retrieve the data from the back end to the front end. Of course, there were cases where people overlapped into other members responsibilities, but this is how the team was primarily organised.

How the Project Went including Issues/Problems Encountered

The project went decently with our team able to successfully build a solution to our problem statement. We believe that we were able to create a system that successfully allowed users to browse for twitch clips based on categories and streamers. However, there were definitely some problems that we ran into along the development process.

One of the problems that we encountered was the inability to code certain features of the project. One such feature was the download button. Initially, when we defined our user stories, the Twitch API allowed for us to create a download button, allowing potential users of our system to download the Twitch clip onto their system. However, since the most recent Twitch API change, this is no longer feasible. Additionally, the dislike feature was also affected by the change to API change and hence both of those features within the system were not able to be implemented into the system successfully.

Another issue that we had in regard to our project was the UI of our homepage. Unfortunately for us because of the number twitch clips that needed to be presented on the main homepage, the formatting of the homepage went horribly wrong. The main issue was that the text lengths of the title of a Twitch clip was shifting the alignment of the Twitch clip in the homepage. This resulted in a major misalignment in the main homepage of our website. In addition to these problems, we also couldn't get the image loading in correctly for the thumbnail. The final product was a homepage that

had no thumbnails and was misaligned. Given a larger development timeframe we probably would have figured out a solution to this issue, however we weren't able to fix this problem in the time given.

Overall, our project went well. Although we encountered a few problems and issues along the way, we were able to successfully produce a product that fulfilled our problem statements. The system created did allow users to browse for uploaded twitch clips based on categories and streamers. If the project was furthered to its full potential, we believe that it could be a viable product on the market, allowing users to successfully browse for Twitch clips with a more customisable experience.

What we would do differently

If we were to redo this project, there are definitely some things that we would change in order to make the development of project a lot smoother and allow for more features to be implemented in the time possible.

One major thing that could have improved the development process and overall quality of the final product is our time management. A lot of the development of our project happened over the final two weeks of the time allotted and if given more time, we would have had a much more functional and better-looking system. The UI of the homepage is a direct product of this, and as aforementioned, if given more time to work on the homepage UI we would have come up with a better-looking homepage with working thumbnails and aligned clips.

Additionally, work could have been better delegated and we definitely did not estimate how much work was involved in each part correctly. For example, we had two people working on the SQLite database and two people working on the front end with the html, JavaScript and Python routes. Realistically we should have had three people working on the front end and one on the back end because the front-end work load was much larger than the back end. This was a misjudgement on our part and if given the chance to redo the project, we would take that approach.

Another thing that we could have done differently is that we could have tried to have more API integration throughout our project. All of the API integration within our project was to do with Twitch API and nothing else. For example, if we were able to integrate the Google login API, that would have alleviated some of the workload off our own developers and allowed users to login with their own existing Google accounts, instead of having to create a new login for a new website. It would have led to an overall better product.

Given more time we would have definitely loved to implement more features. There were a lot of features that we defined in our initial user stories that just didn't come to fruition, mainly because we just bit off more than we could chew. We needed to definitely make our scope of our project much more simplistic in order to match the time frame and because of that, a lot of our features weren't to the highest quality. Given another chance at the project, we would definitely reassess the time frame given and make sure that we could implement the features that we wanted to the best of our abilities and to the highest quality.

There's always going to be stuff that we would want to do differently, and that's the point of having these sort of projects, to make mistakes and rectify them in future project. Overall, we're happy with the project produced and have become better product developers from it.