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

Software Architecture

External Data Sources

For the clipbucket project we have decided to use twitch's own database where all twitch clips are stored as the main data source for the project.

Software components

the selected Web stack showing major software components that comprise your solution. These will include both components that need to be developed and third-party components (e.g. web browser).

- Flask
- SQLite
- Bootstrap
- Docker
- Web Browsers (We'll be targeting Mozilla Firefox)

Figure 2A: A layered view of our application stack.

Web Application Stack
Web Browsers Layer

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

Server-side Web Framework Layer

Database Layer

Deployment Layer

Operating System Layer


Relating choices to components

- Sql for handling http requests/clicking on links/searches and also storing the project data
- Bootstrap for the front-end component to display a page for the user depending on which page they request
- Apache for web hosting
- Flask for back end component

choice of an implementation/technology or framework

- Python 3 (Flask)
- JavaScript (Bootstrap and jQuery),
- HTML and CSS (Bootstrap),
- SQL (SQLite and SQLAlchemy)

choice of platform

The platforms which the project can be supported on will include any computer with a web browser. This covers all operating systems such as Linux/Windows/Macintosh

key benefits/achievements of your architectural choices

Advantages	Disadvantages
Flask	
• Simple request-response design allows quick prototyping. • Simple templating system allows easy integration with the rest of our stack (i.e. Bootstrap) • Our team has experience working with Flask. • Does not require strict adherence to a design principle, reducing waste of time on developing boilerplate and learning its way of doing things. • Easy to learn in contrast to a more difficult framework such as Node. • Flask is API-stable for the long-term with its 1.0 version.	• Flask is not as full-featured as some other frameworks (i.e. Django). It does not include essential features such as a built-in login system. • Is difficult to deploy in production. Requires set-up of a WSGI server to deploy in production. In comparison, PHP easily integrates with the Apache web server. • Loose design principles require us to come up with our own software architecture compared to (i.e. Django or Laravel).
Bootstrap	
• Simple HTML/CSS oriented design, with minimal javascript. • All in one framework for HTML/CSS and javascript • Does not restrict our ability to integrate with other Javascript libraries. • Built-in cross-browser support • Customizable and lightweight	• Does not apply well on applications with lots of mutable data that need to stay consistent with the user's viewpoint. In comparison, React is made for this purpose. • Slower to do full updates because it doesn't have an in-memory DOM. • Doesn't allow single-page-applications like Angular or React.
SQLite	

• Simpler deployment. It doesn't require installation, and then a connection to the project itself in comparison to mySQL, • Suits the project as we only need to store web data for the project itself, we don't need multiple user data features that are in mySQL	• Unlike server-based database solutions, SQLite works as a file on the filesystem. It does not support deployment on multiple servers. If the project begins growing, SQLite may not scale to our needs and will require refactoring it to a different solution. • Doesn't have the same security level as mySQL • No option for user management unlike MySQL
Docker	
• Allows for contained deployment which is consistent across all production and development environments. • Eases development conflict by maintaining a consistent environment and library versions across all developers. • Eases deployment to production	• None

Achievements of our choices

- Flask will allow us to handle all the http routing on the web page
- Bootstrap will allow us to do all the web interfacing required for the product, create links, formatting of the web page etc.
- Docker will allow us all to code in a consistent environment and allows simple deployment
- SQLite will allow us to handle all the data of the web page and managing our data

Technology choice justification

Flask

- we chose flask because the majority of the group has experience with flask
- Flask allows for very easy and flexible prototyping
- The template of flask allows easier integration with the rest of the web stack
- In contrast to Django or node.js or even javascript to code the back end of the project, those languages are much more difficult to learn

Bootstrap

- We chose bootstrap because it's an all in one framework which allows for simple integration of HTML/CSS with very little javascript required
- It allows for integration with the other javascript library in case we decide to use some react for a feature or angular for a different feature
- In contrast to react or angular which are very difficult languages to learn, bootstrap is simple to learn and can develop the front end of the project in the time required

SQLite

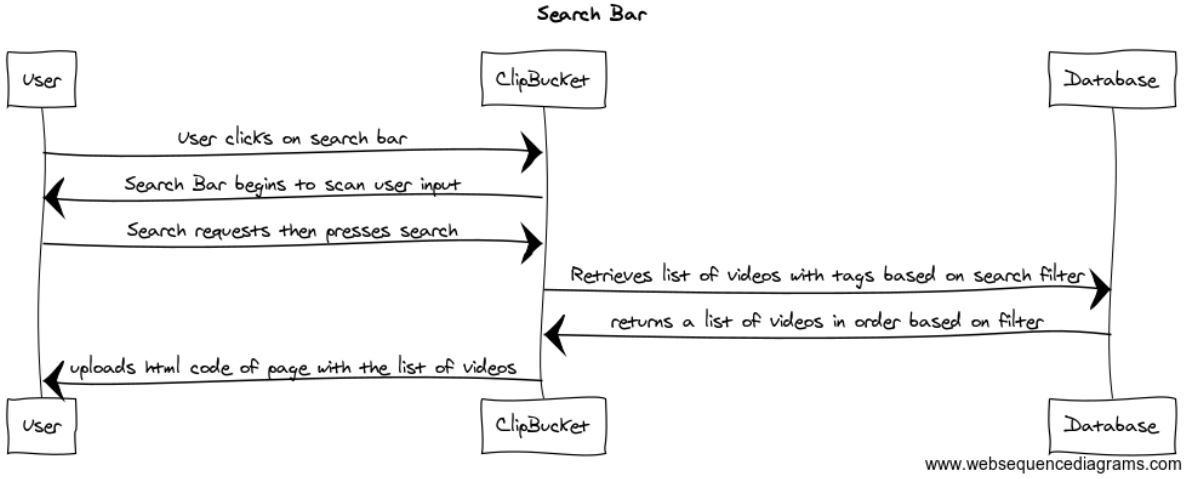
- We chose SQLite because it allows simple deployment from within the application itself
- We also chose it because this prototype does not require to account for scaling
- In contrast to MySQL which has more safety features/scalability and allows users to manage data, all these extra features are not required and mySQL doesn't allow as simple of a deployment technique (requires installation and connection of device to server)

Docker

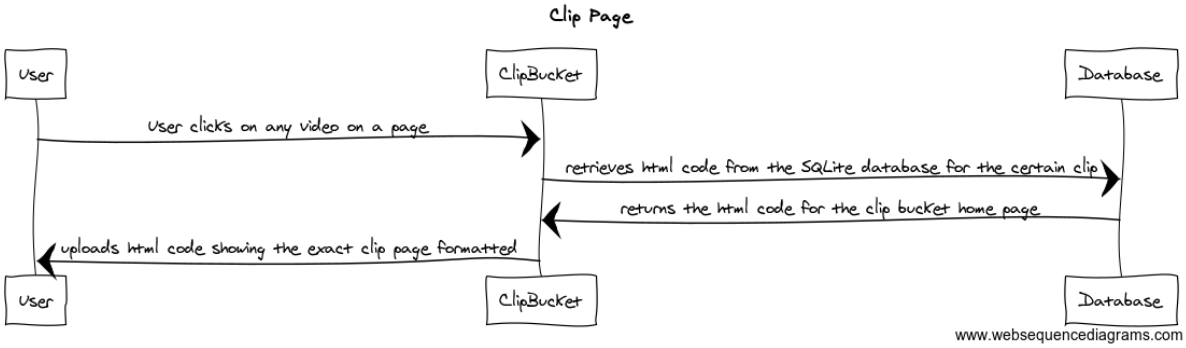
- We chose Docker because it for contained deployment so all code is consistent and regardless of environment or machine
- Docker also makes sure there is no conflicts on different systems
- There were very few choices to compare with as the only other alternative was bash scripts, but even then docker also allows for shell scripts

Initial Software Design

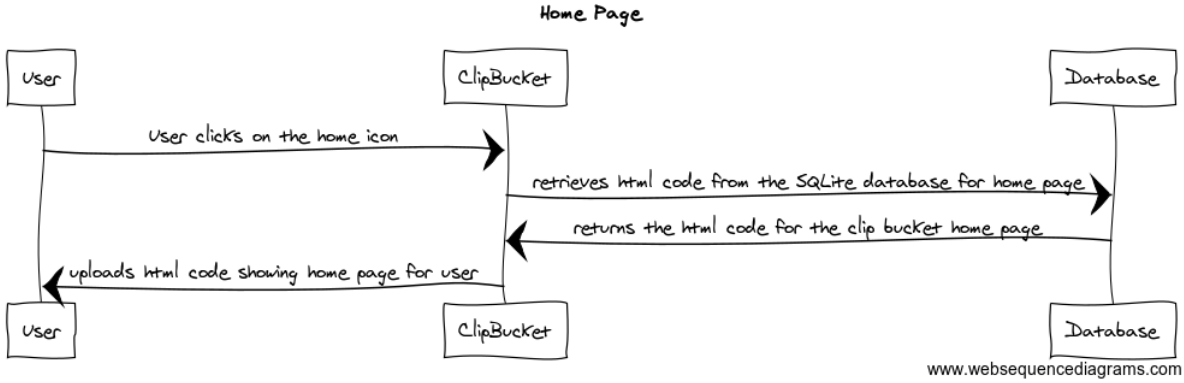
Search Bar Sequence Diagram



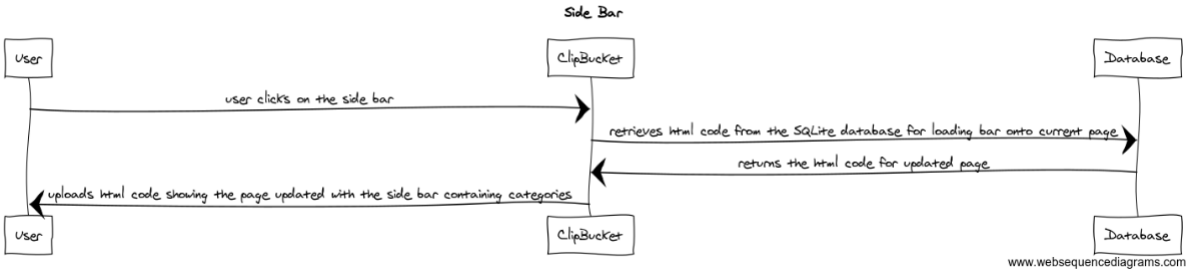
Home Page Sequence Diagram



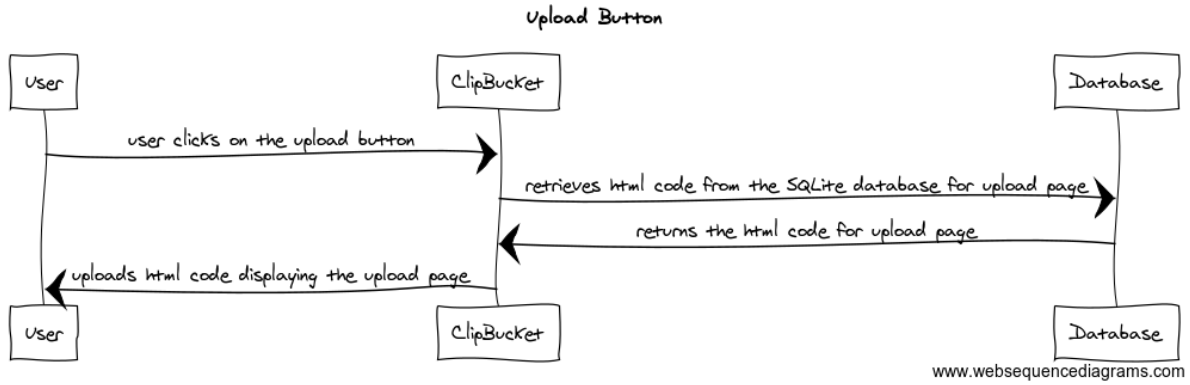
Clip Page Sequence Diagram



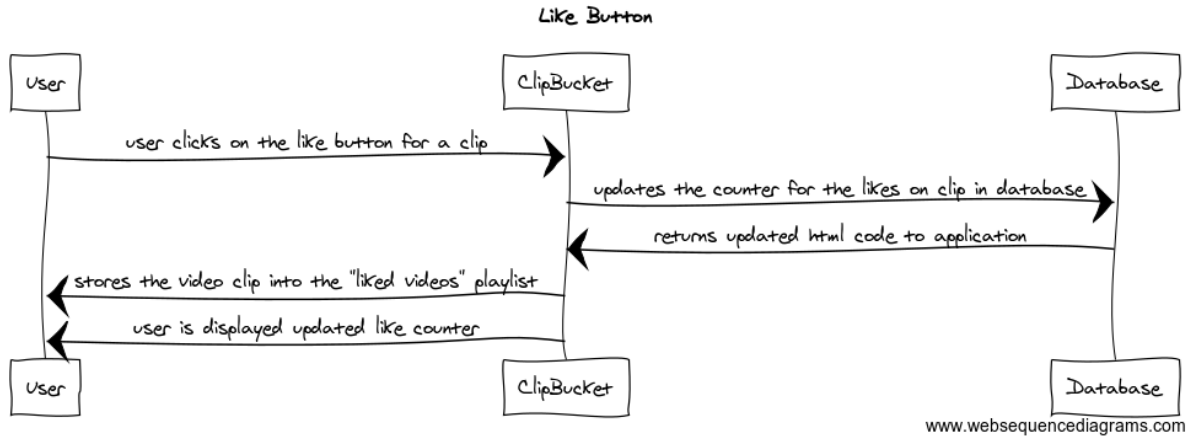
Sidebar Sequence Diagram



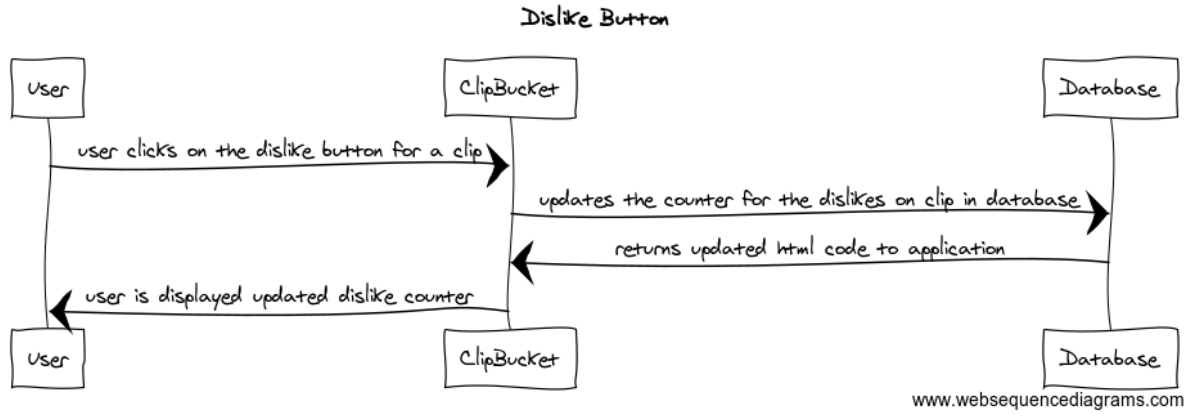
Upload Button Sequence Diagram



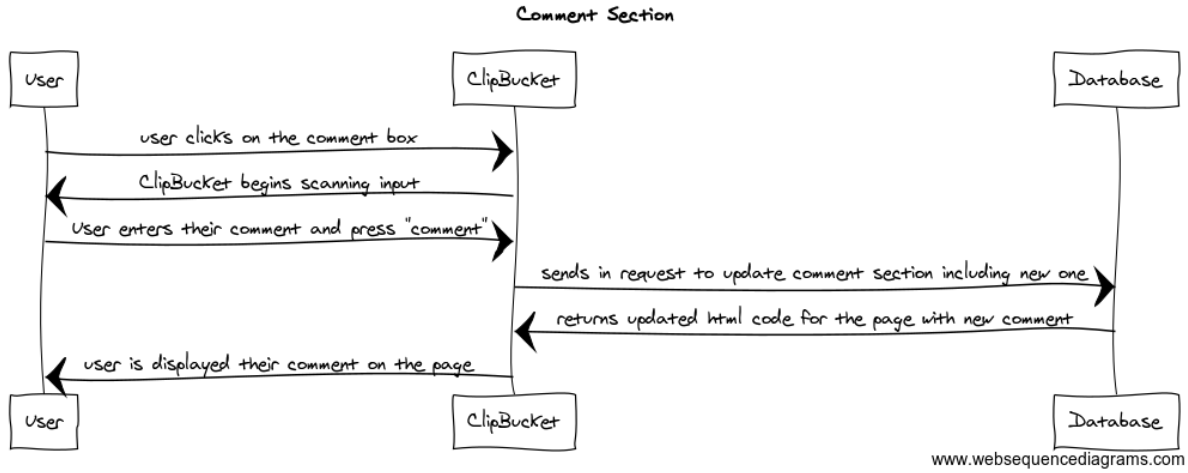
Like Button Sequence Diagram



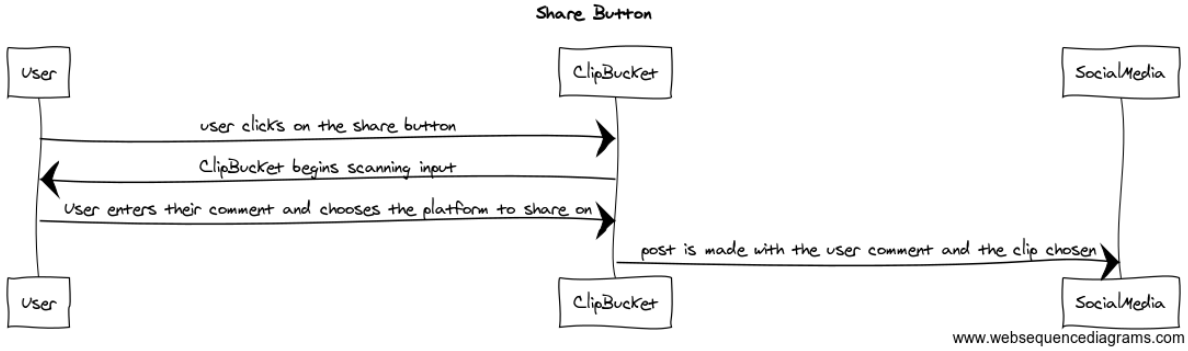
Dislike Button Sequence Diagram



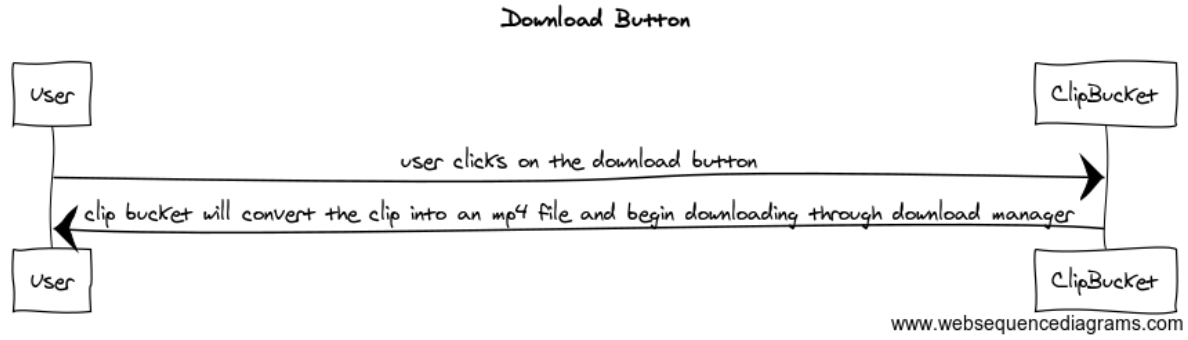
Comment Sequence Diagram



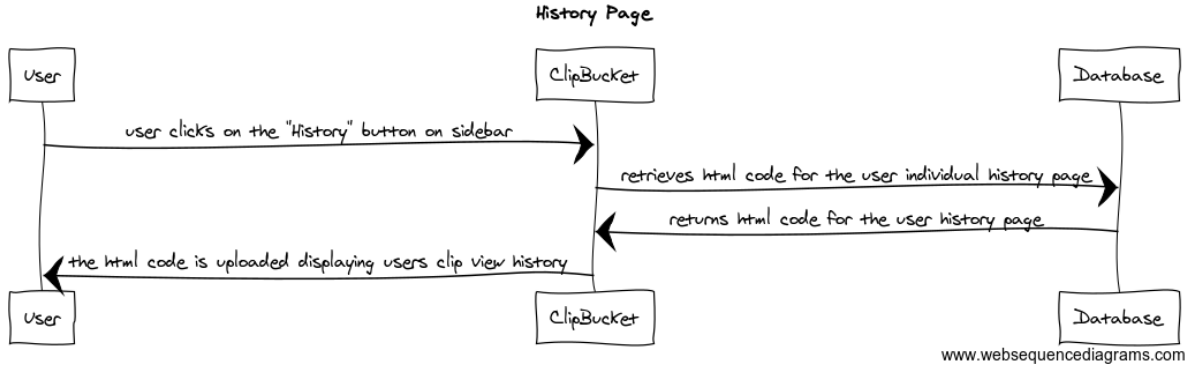
Share Button Sequence Diagram



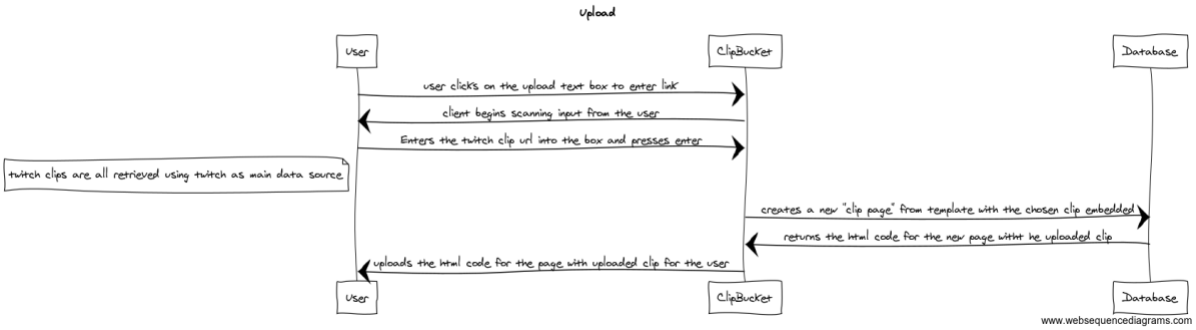
Download Button Sequence Diagram



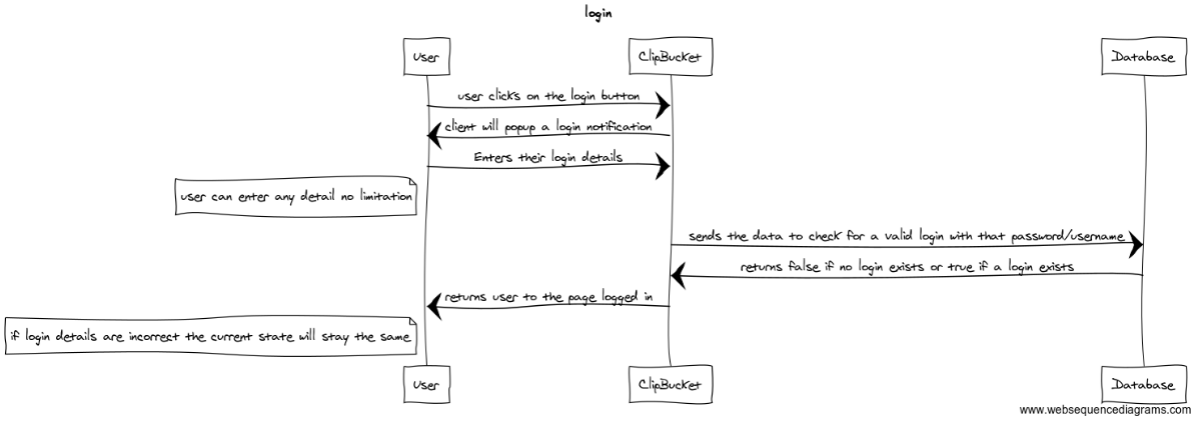
History Page Sequence Diagram



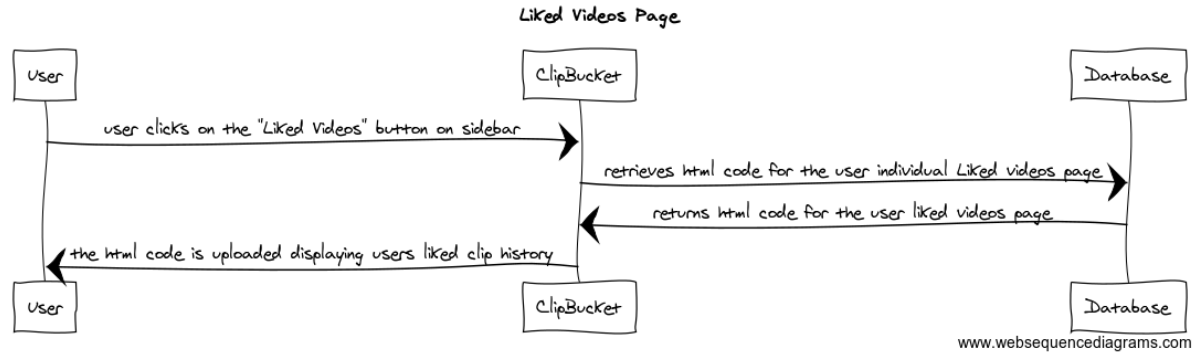
Upload Sequence Diagram



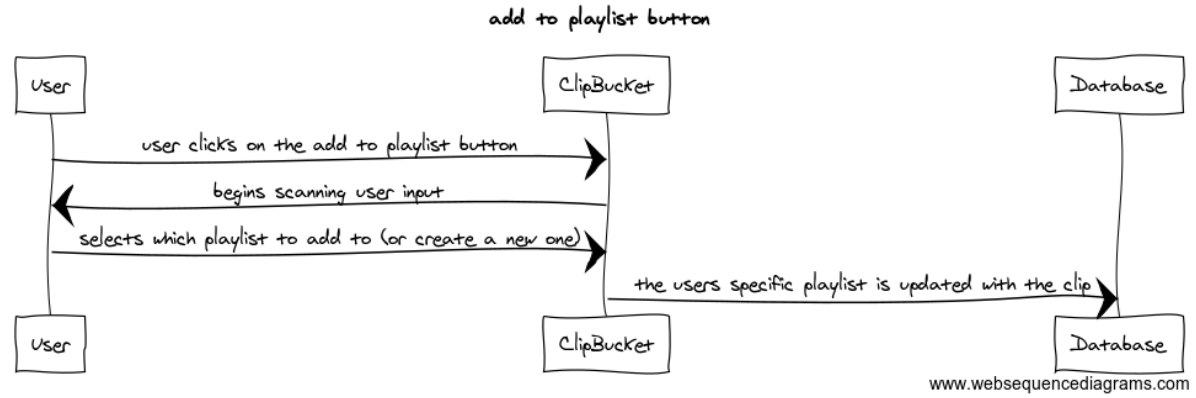
Login Sequence Diagram



Liked Video Sequence Diagram



Add to Playlist Sequence Diagram



Playlist Page Sequence Diagram