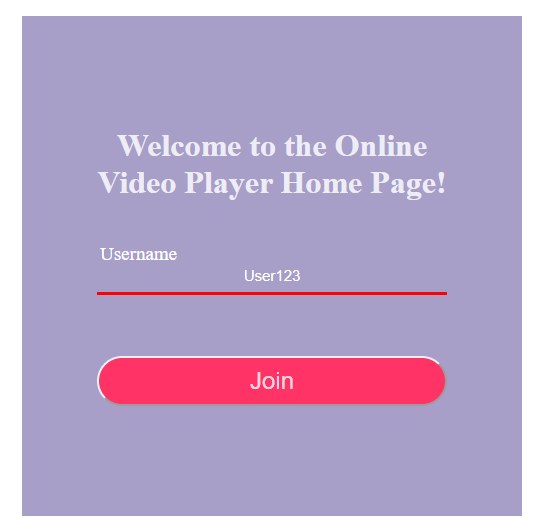
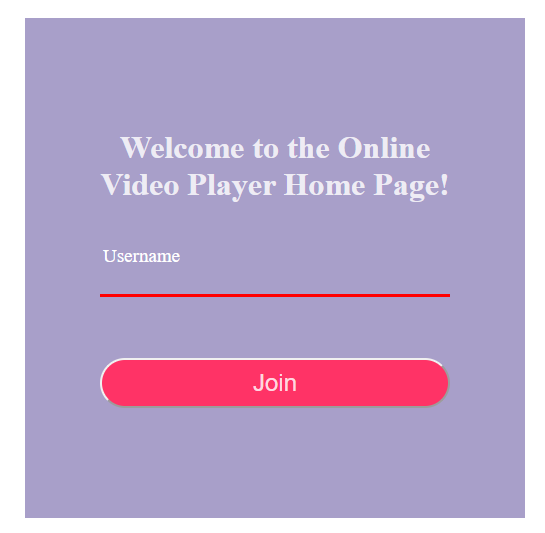
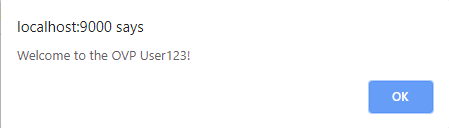
**SOFT355**

**Student ID: 10578755**

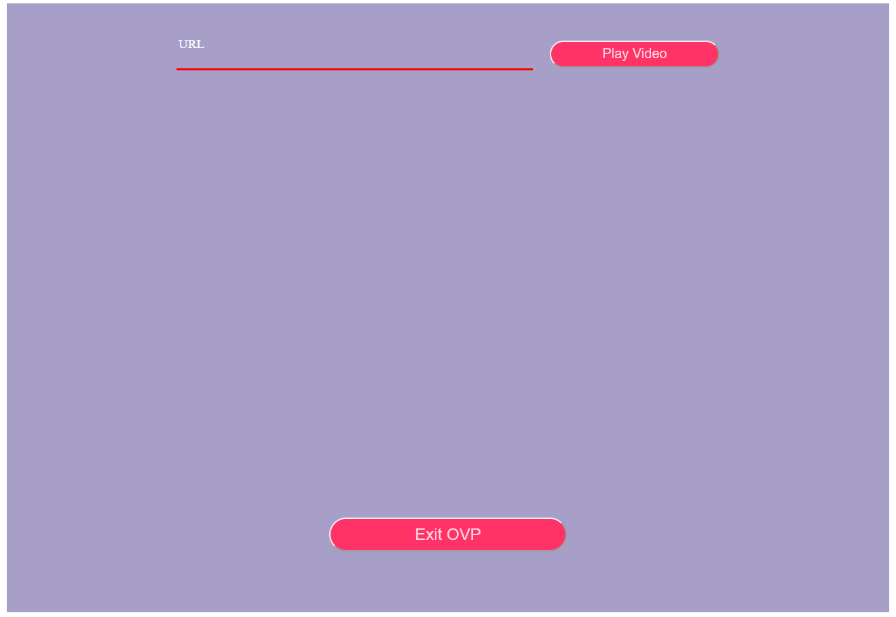
**Functionality**

The final deliverable includes two .html files the Home Page of the application where the user has to enter their username to access the Online Video Player page where they get to paste a YouTube URL video into an input field and with the click of the play button every user who joined the OVP gets to see the video a user wishes. That works more as a public group chat for videos used for example by a group of friends instead of sending videos to each other they all together join the OVP and whoever has something to show to them they just paste it and it gets to play for every one. The system uses Express/Node.js as the server and WebSocket to communicate with the clients and display the video to the video played by a user to the rest inside the OVP. Moreover, the application includes jQuery, CSS, and basic JavaScript for the functionality of the program. The application uses MongoDB as the Database where we have 2 tables called Users and URLs. Users stores the username of each user joining the OVP and the URLs stores the video links the users paste in the OVP.

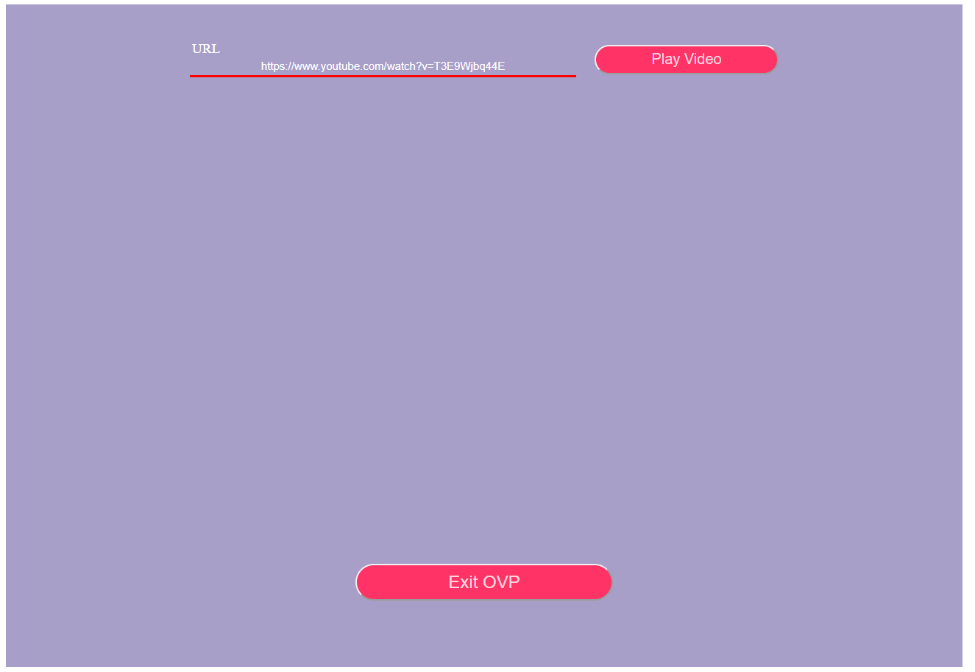
User interactions start at the home page where they get to type their Username to access the OVP by clicking the button Join.

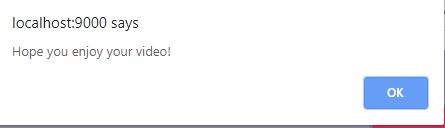


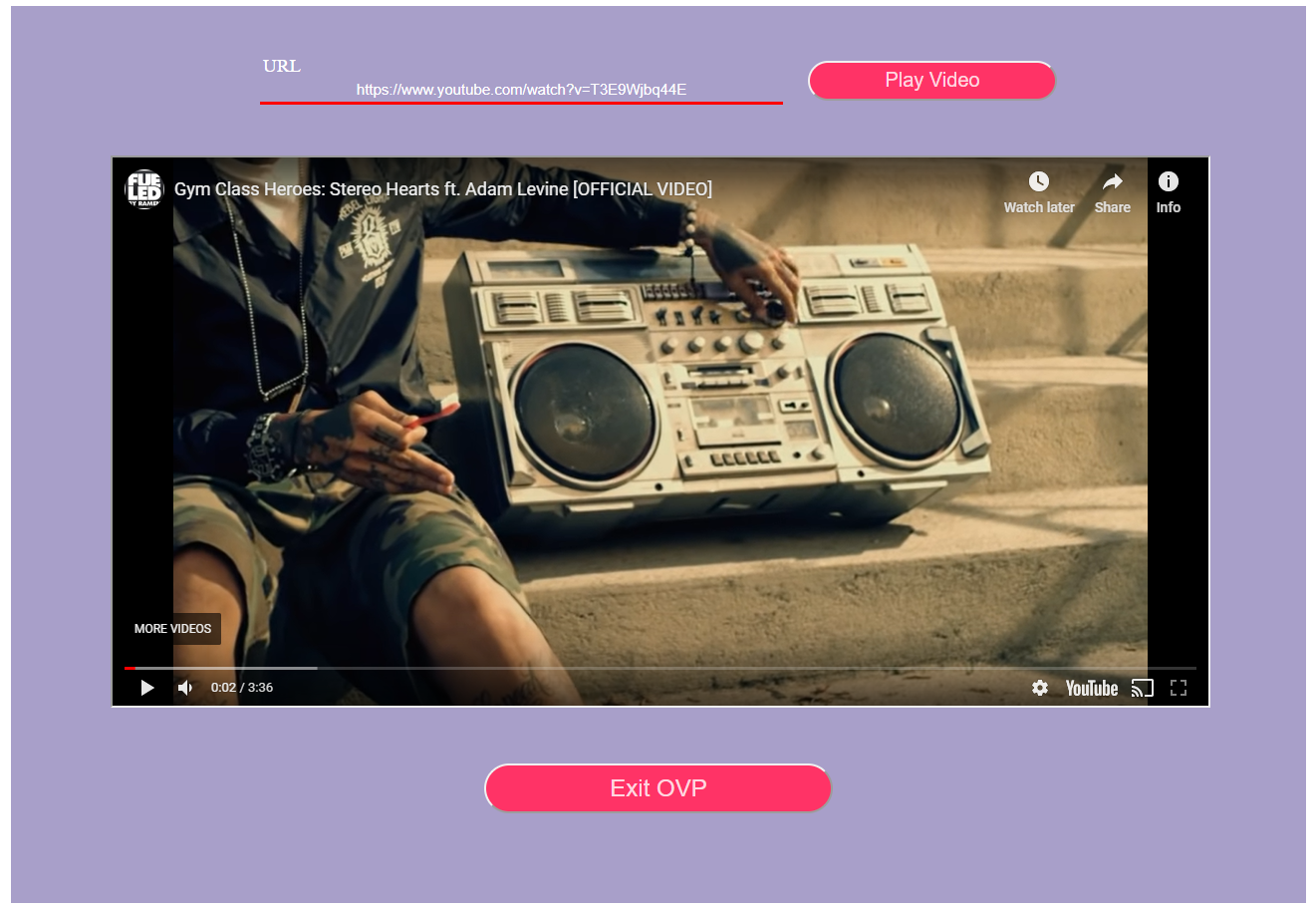
After joining the OVP they will view the OVP Page where they can play their videos.



So the User who desires to view or display a video to the rest who joined the OVP copies a YouTube Video URL into the URL input and by clicking the Play Video button the input is successfully displayed to them and the rest of the Users inside the OVP.







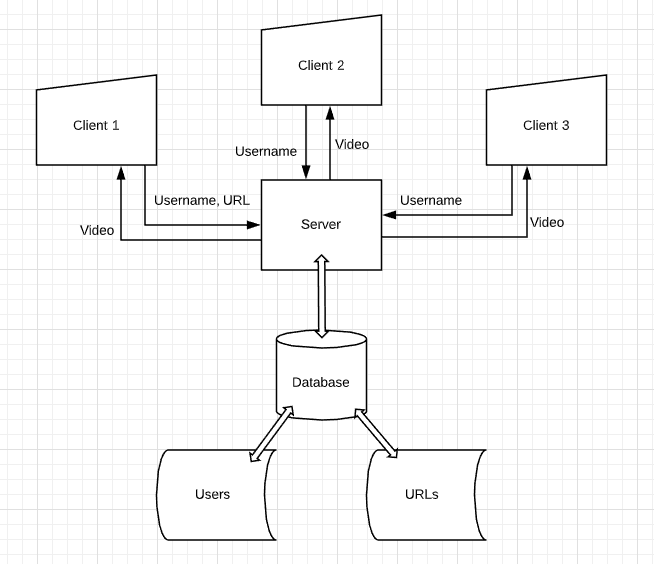
If the user wants to exit the OVP they just press the Exit OVP button where they exit the Player.

**Requirements**

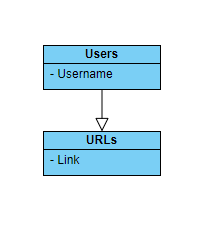
The application is aimed at all the group of friends out there who instead of having to send the link of the video over to their friends to see it they can now all join the OVP together and play it for them with just a click. The features included were a WebSocket that gets the desired input from the User’s client playing the video sending the video to the server then the server sends the video back to the rest of the clients and plays the video.

**Design**

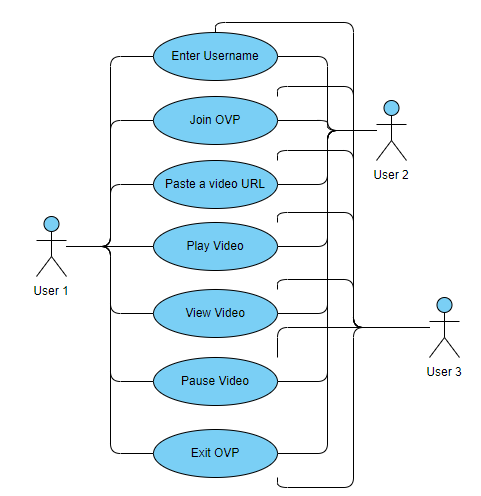
Basically, let’s say we have three Users who want to join this OVP chat. Each User enters their username respectively with a post functions implemented in Server which receives, connects and stores them into the table Users. With some JavaScript we transfer the Users into the OVP on button click. After they join the OVP a User wants to play a video. User A pastes a YouTube Video URL and clicks the play button. The video gets transformed into an Embed mode with some JavaScript and with the help of another post function implemented in Server which gets the URL and sends it into the Database and gets stored in the table URLs. Play button also has a function which will get the User 1 input and using WebSocket will display the video to User 2 and User 3 so everyone inside the OVP can see the video inserted. Same happens for each User who wants to play a video for the rest of the Users. If User 2 wants to play a video then he simply enters the URL and then clicks the play button and the same process is repeated so User 1 and User 3 get to see User’s 2 video.



User Class:



Use Case:



A get function fetching the data from the database to display the users who are in the player was implemented as well managing to get the list with the users but wasn’t displaying appropriately what I wanted it to display it got commented out.

**Testing**

QUnit was used in terms of Unit Testing performed in this application where is divided into 2 parts. Firstly, checking if the User didn’t enter a username to access the OVP and second if the Username matches a specific Username set in the files where both tests passed. Same goes for the URL. QUnit was used for that as well where it checks if the URL is null or if it’s a specific YouTube Video URL where both passed the criteria.

The Usability Tests used were some simple instructions to perform on the completion of the application. Users were given the tasks to

1. Enter an appropriate nickname.
2. Enter the OVP
3. Copy a YouTube Video URL in input
4. Click Play
5. Watch the video

Simple and basic tasks where every user managed to complete.

The reason I believe my strategy for the testing performed is appropriate is based on giving some simple instructions to the Users and by the design of the Application it was easy for the Users to navigate and complete the tasks. If Users then manages to complete those tasks without any questions and problems means that there won’t be any trouble for any user interacting with the page.

**DevOps Pipeline**

Firstly, I started by creating the server so I can establish a connection with the server before doing anything else. Afterwards, I made the connection between the server and the database and tried to pass json data inside to test if it actually worked. Then I implemented the client and the button functions that would allow the manipulation of the page by the user and doing what it had to do. After doing that I created the 2 post functions in server to get the user input and store it in the database. After implementing a function in JavaScript to get the video and transform it into an Embed video to play working in one client then came the addition of the WebSocket where it would send the video to every client connected to the video player and make it an actual online video player for groups of friends to join and play the videos they want for their friends.

GitHub repo: <https://github.com/Theohr/OVP>

Every move and task after implementation were immediately build and tested to check if it was working then moving into the next task. The result conveyed that all the functions and code implemented work fine and the application does what is meant to do after all with no errors occurring.

**Personal Reflection**

In conclusion, the application that has been submitted is sufficient overall doing the work I was expecting it to do. I manage to get the user’s input not only the username but the URL, send them to the Server then from there store each one accordingly in their tables. Then by using WebSocket to display the video desired by a User to the rest of the Users connected to the system. Displaying a list of the Users who were currently in the chat was implemented fetching the data from the Users table from the database but due to not a good usage of Angular I cant manage to display only the Usernames, IDs are being displayed as well.

Researching and getting to know better each feature and library used in this application would be the first step if I had to take something with me for the next project such as this as I wasn’t well aware of some things before starting the project but got into a better understanding while doing it. This was a new experience and many things were taught by this that will be applied onto the next projects to get the best results possible.