**TiaoTiaoXiong TV Project Report**

Dr. Hungwen Li

CMPE131 Spring 2021

Group 11

Chengkai Yang, Hongkai Huang

Student ID: 012503569 (Chengkai)

Student ID: 012117794 (Hongkai)

Date: 05/06/2021

**TABLE OF CONTENTS**

EXECUTIVE SUMMARY………………………………………………………………….2

1. INTRODUCTION…………………………………………………………………...3
2. BACKGROUND and OBJECTIVES……………………………………………….4
3. APPROACH/METHODOLOGY……………….…………………………………..5
4. RESULTS, FINDING AND ANALYSIS………………………………………….14
5. CONCLUSION and RECOMMENDATION……………………………………...19
6. SOURCE CODE…………………………………………………………………...20

**EXECUTIVE SUMMARY**

It’s believed that humankind is a species that needs frequent social communication, otherwise we might feel our life meaningless or undesired. However, imagine if one day, we all cannot meet with our friends or family in-person to chat, to play or to hang out but could only stay at home for more than one year with your phone/computer. Guess some people may think that’s a new lift style worth a try. But in early 2020, this ‘dream’ surprisingly came true and most people started to feel bored after being isolated at their homes for the first 2-3 weeks. People still need physical social interactions to stay mentally positive. Thus that’s how we had the idea of creating this web app for people to watch movies/videos and share their experience online with their friends and family, to possibly create a social scenario where people virtually meet in person. Here we wrote the report to introduce our web application--TTX TV which could help people overcome this lonely time.

Firstly our report will introduce you to the basics of our web app--TTX TV, which is a video sharing web app that can allow users to upload their movies/videos and share/watch with their friends and family online. Then the second part introduces you to the background of creating this web application -- in 2020 people started to stay at home to reduce COVID-19 virus spread. Also we will go through the objectives we want to achieve, including 2 big parts --- front-end and back-end which will connect each other with a Restful HTTP connection. Next, we will show you the approach and methodology we used to fulfill our goals and make our web app practicable. To make our web app convenient, user-friendly, privacy-conscious, and widely accessible, we conceived the functions and structure of the application and designed our low fidelity prototype in turn. A good example for user-friendly is our user-interface (UI) which is carefully designed to provide a clean user experience. In the fourth part of the report, we will share the results and findings after testing our web app with sample customers (randomly selected from our friends/classmates). Also we would provide the analysis of user experience survey results to better understand the pros and cons of our web app and what’s next for it. At last, it’s the conclusion of our report and some thoughts on the future development.

**I. INTRODUCTION**

The reason for designing Tiaotiaoxiong TV is because of a demand for movie watching with family and friends during the age of COVID-19 and lack of related software products/services in the market.In the COVID-19 age, people are isolated at their home and outdoor activities/entertainments like seeing movies in the theater are limited.

Tiaotiaoxiong TV is a web application that wants to provide people the scenario of watching movies together online. We want to help families and friends to still enjoy the entertainments when they are getting together, even if they are separated by distance.Tiaotiaoxiong TV will serve users with a fast, immersive and relaxing experience.

The web application is designed with front-end,back-end and database. The front end takes user input and receives feedback from the back-end. Then display the result on the screen. The Back-end proceeds the user’s input from the front-end and query the corresponding data from the database then send back feedback to the user. Database will contain 2 tables. First one stores each room's name, password and a unique ID. The

second one stores each media file uploaded with the ID of the room it belongs and its stored

path.

**II. BACKGROUND and OBJECTIVES**

The idea for developing our web app system was to help customers who wanted to share their movies/videos to their family or friends during COVID-19 age. Because from early 2020, the global breakout of COVID-19 made people isolated at their home alone, but people need some interpersonal interactions. Having some online entertainment was one of the good ways during this stay-home time. Then we had a thought of allowing people to watch movies physically at their homes but mentaly share watching experience together online, which is very close to what people can get for going to a cinema.

The very beginning idea is to make the web app generate a media room with a unique name and a password upon users’ request; Allow users be able to join a media room after they enter the correct name and password; Allow users be able to quit the media room while they are in one; Make web app be able to receive media files uploaded by users and stream it in the media room; Make web app be able to display the list of media files uploaded in the media room while in that room; Make web app be able to play or delete the media files uploaded in the media room depends on users’ request.

Then we started to draw the software's architecture and objectives we want to realize for our web application. The architecture will split into 2 big parts, front-end and back-end, they will connect each other with a Restful HTTP connection.

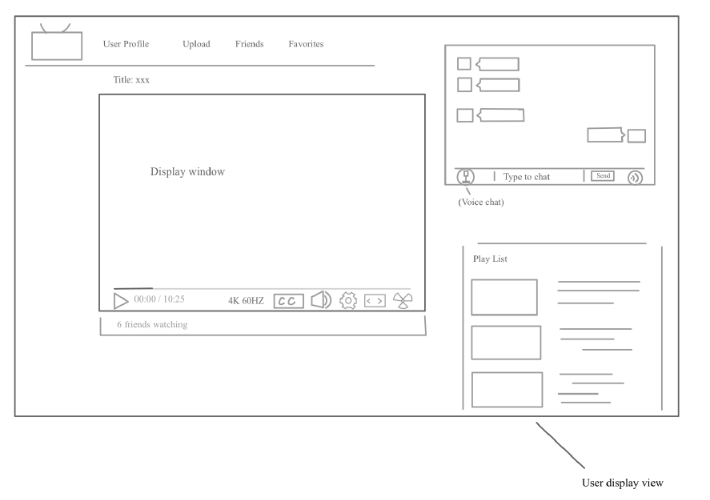
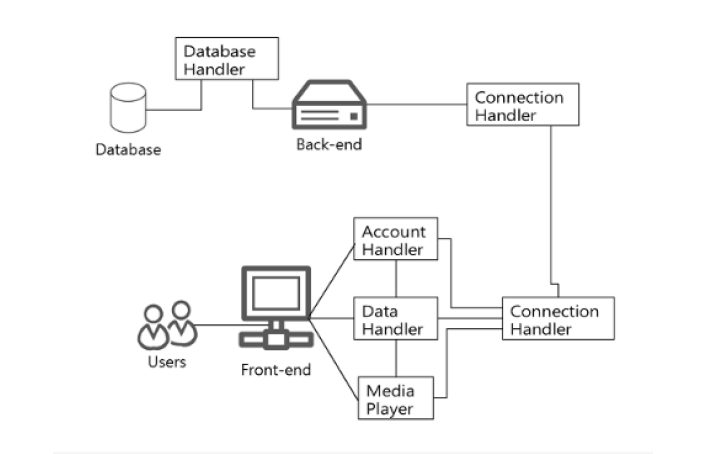
Front-end provides a user-friendly interface, grand user's input and return feedback. It will be divided into 4 main subsystems, connection handler, data handler, account handler, media player. In front architectural. Data handler will handle most of the data transfer between subsystems. And all of the requests to the backend are sent to the connection handler to process. Connection handler also sends the response from the backend to each subsystem to process.

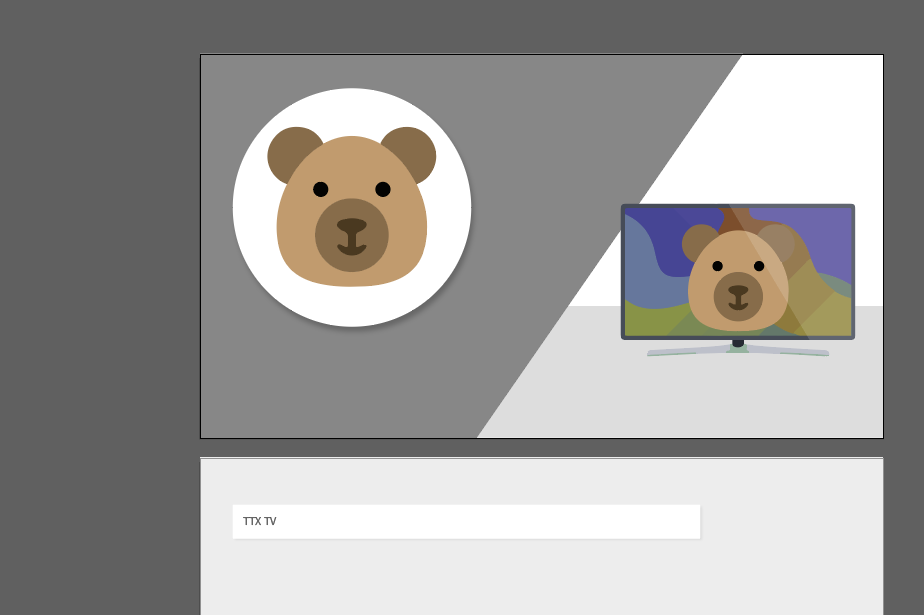
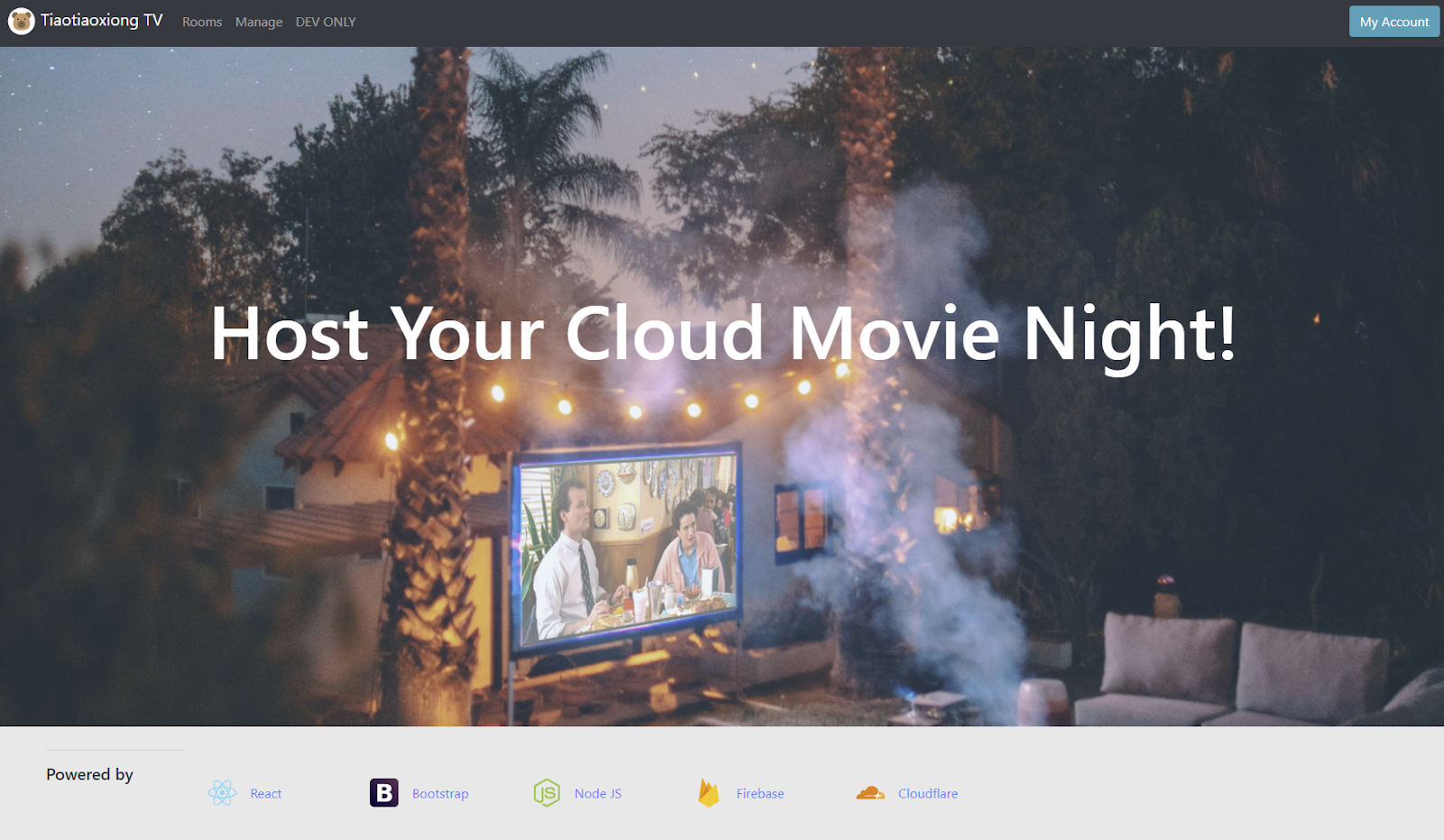
Backend provides a restful interface for frontend to access, it manages the logic of data flow in this application. It reads the requests from the front-end and sends back responses, it is also in charge of the database access. It acts as a middleware between the user interface and datas. It can be decomposed into connection handler and database handler

**III. APPROACH/METHODOLOGY**

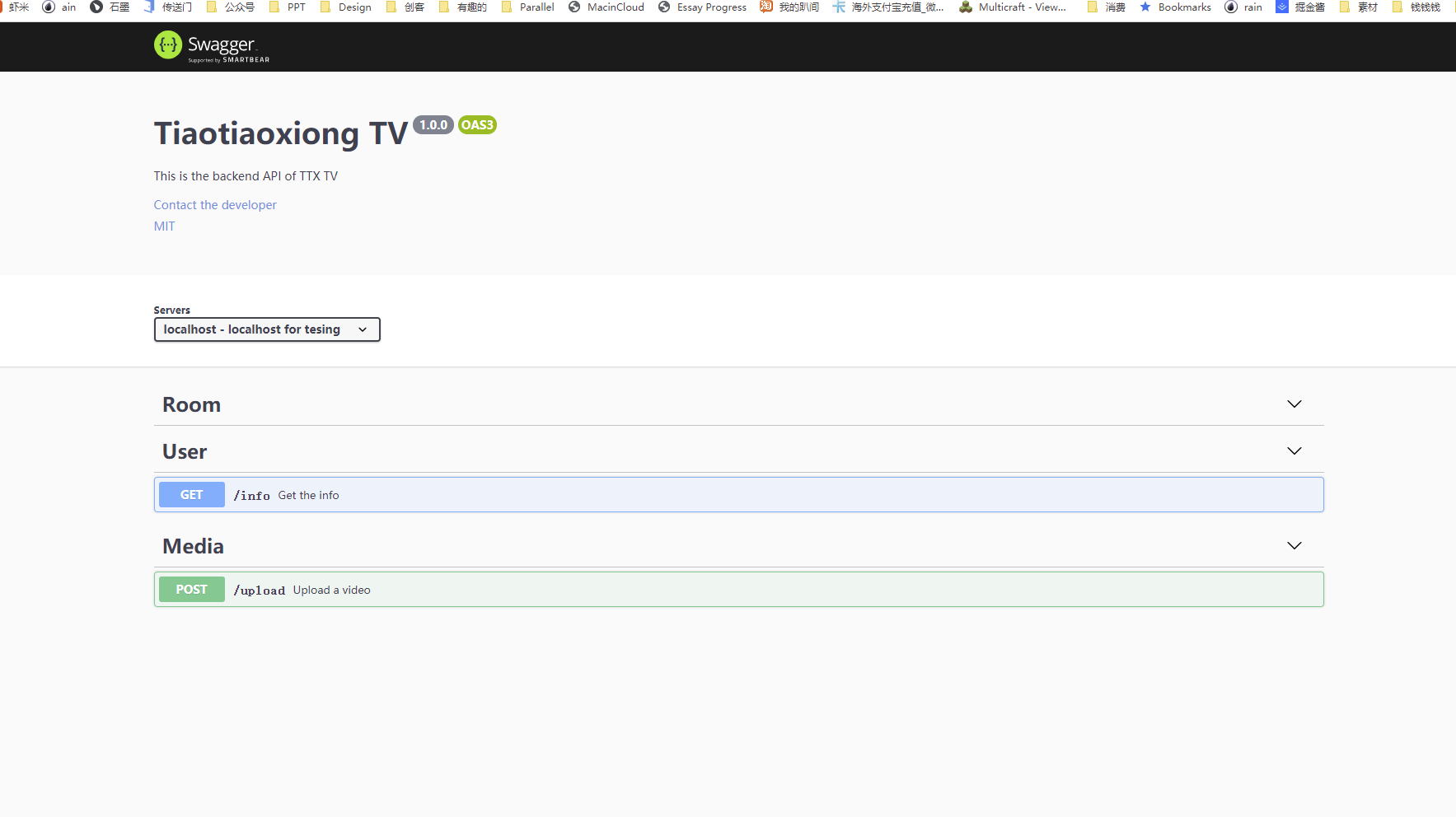
**1. Approach**

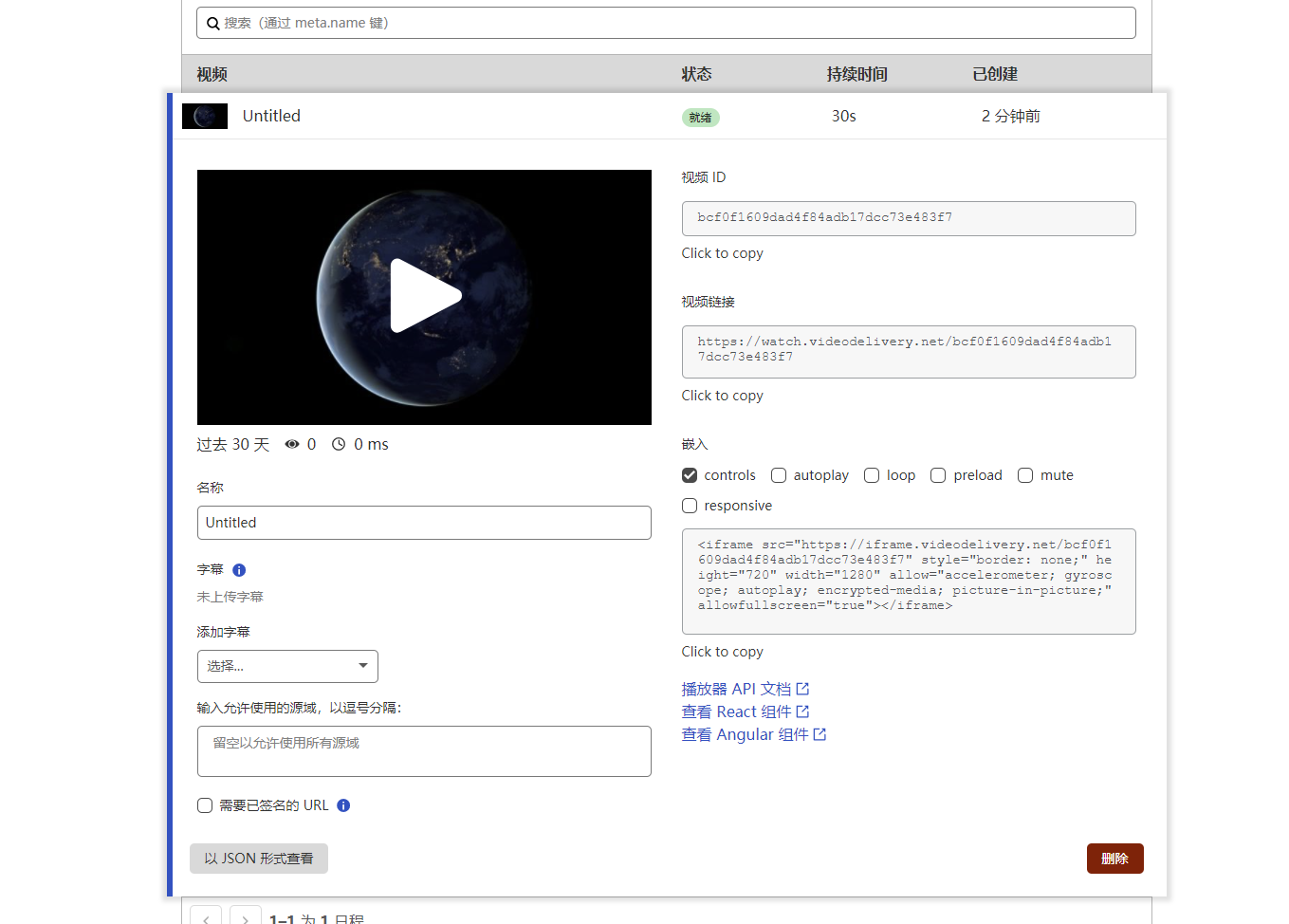
To achieve this goal, we started to design an application for uses to share content and watch together. We considered the application should be convenient, user-friendly, privacy-conscious, and widely accessible. Based on these beliefs, we conceived the functions and structure of the application and designed our low fidelity prototype in turn.

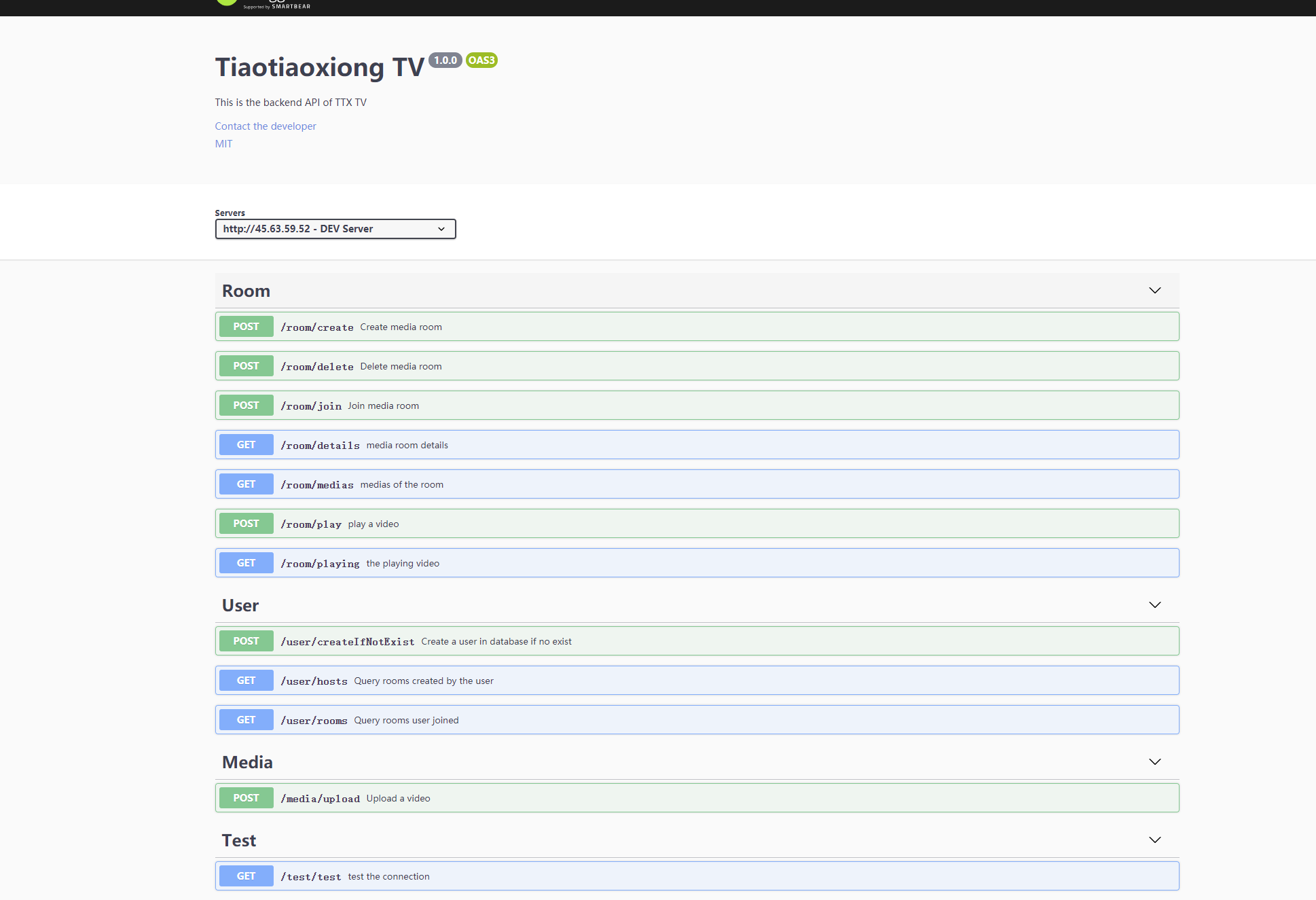
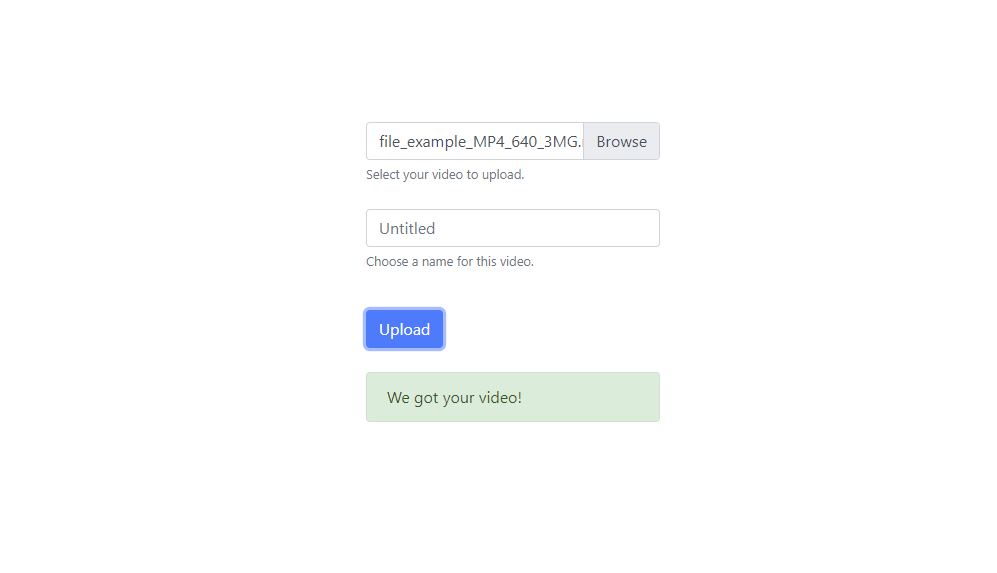
After considering the design at the technical level, we combed them into the Software Design Document and started the development. Firstly, we looked into the user interface's styling design since it is a very prioritized factor to deliver a good experience. We designed our brand and built a skeleton of the graphic interface in order to ensure the convenience of interaction and the delivery of user experience.



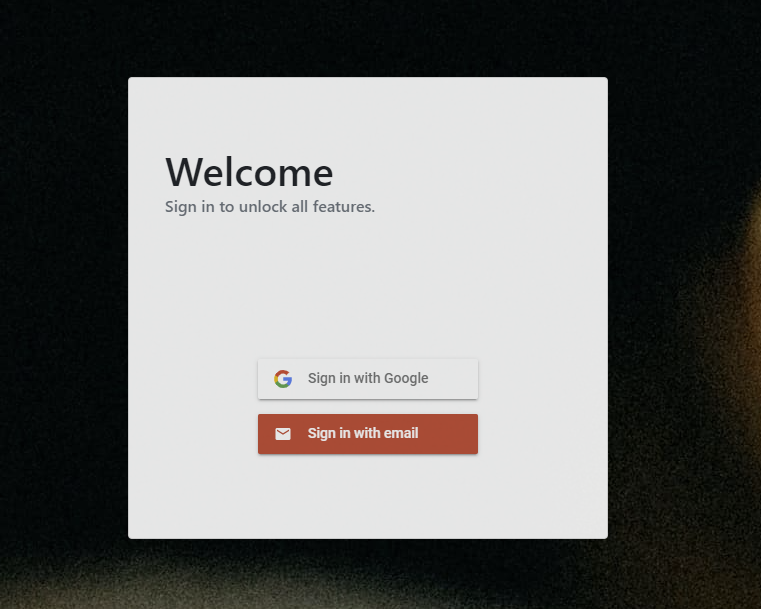
Then began the development on the technical level. We have decided to build TTX-TV as a web application since the accessibility of websites is capable of covering most of the user terminals nowadays. Thus, Node.js becomes the perfect option for us. Node.js is a common JS framework that is primarily used on modern web applications, and React JS and Express JS are frameworks based on Node JS. React JS uses virtual DOM technology to best ensure our front-end interface's speed and performance, and Express JS is a perfect choice for building a neat RESTful backend and rapid development. Moreover, build on these frameworks would be very in line with our front-back separation design.



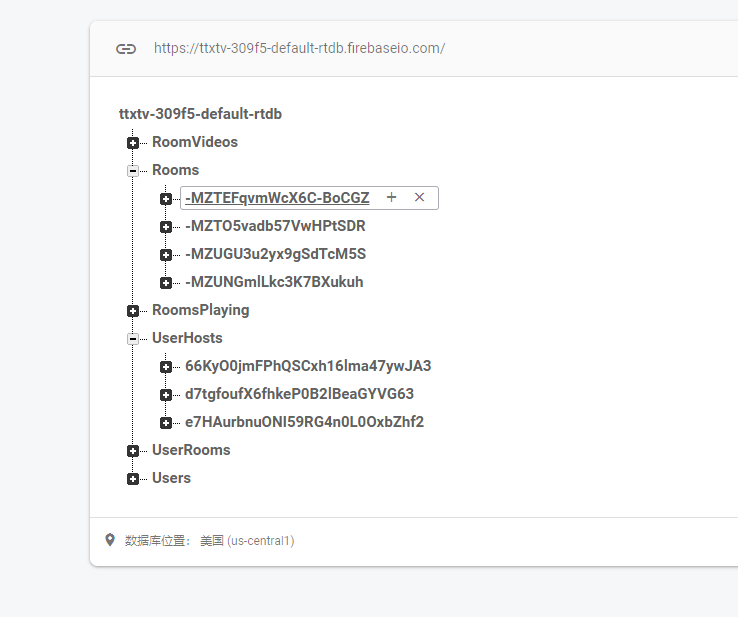
The media upload function and streaming player component were first developed and tested on the front-end side, considering we want to minimize the scope at the beginning. We choose Cloudflare Stream as our streaming service provider，by implementing its API, we finish the most function of video playing and completely demo it in the Prototype Review 1

With the success of Prototype Review 1, we continued to work on the remaining functions of our application. The next influential part of the application is the back-end API. It will handle most of the logic on functionality level, communicate between database, service and GUI. Using Express JS, we quickly practiced all the routes we have designed in Swagger. The back-end will mainly handle 3 parts of functions: user account interactions, media room interactions, and media uploads. 

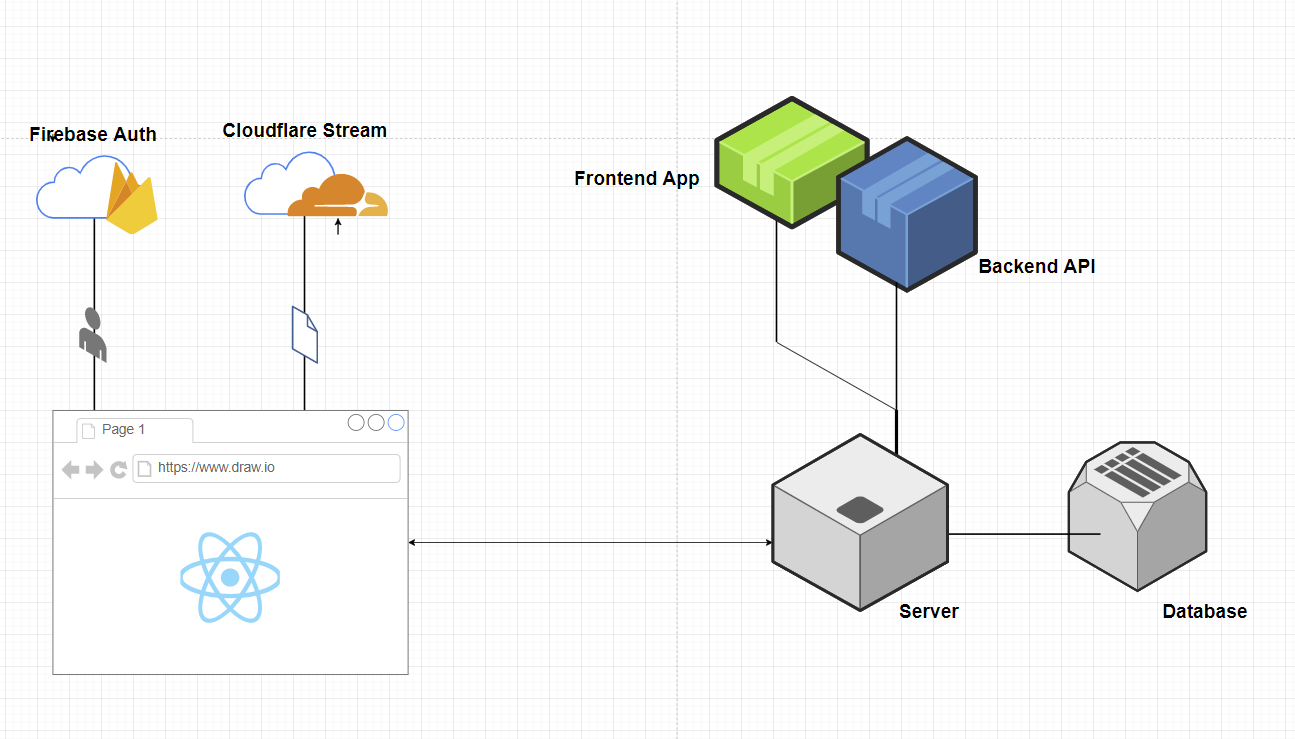
For the user account interactions, we congregated Google Firebase. Google Firebase is a powerful platform for applications, it provides various kinds of well-documented services. In our case, we have integrated Firebase Auth and Firebase Realtime Database. Firebase Auth provides us with simple, safe interfaces. In addition, it helps us fix the user accounts authentication in an easier way. With the user account system done, we can then distinguish users, serve them their data, and recognize their actions correspondingly.



We stored all of the data in the Firebase Realtime Database, which is a NoSQL real-time database. Difference from the traditional relational database, NoSQL has no relation restrictions. It is swift and convenient for rapid development in our case. Using Firebase's NoSQL database, we avoided direct CRUD interactions with the server, which significantly improved the security of data operations.



Following the completion of all minimum viable functions' development, we deployed the app and demoed it in Designed Review 2.



**2. Methodology**

In order to get fast service for our users, we use lots of methodology to improve our web app.

1. Front-Back separation

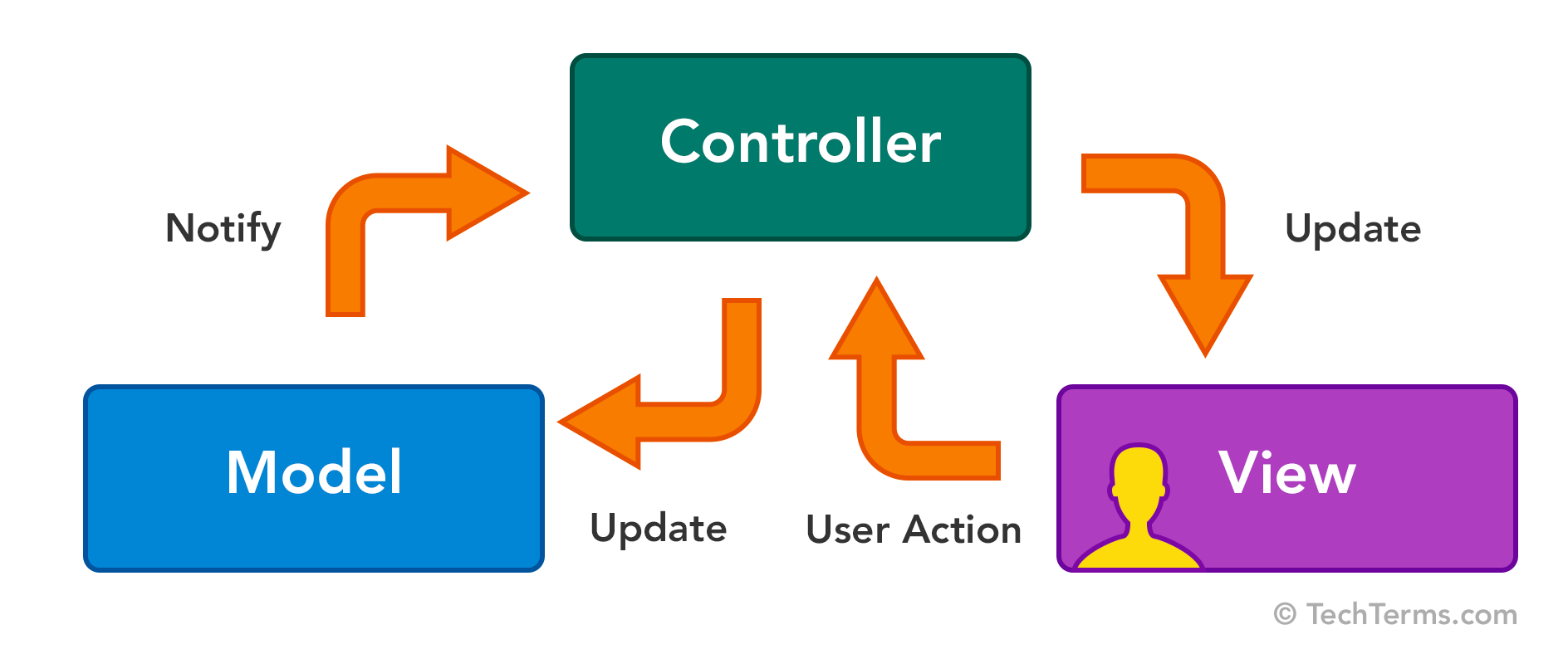
The separation of Front-end and Back End will cut the code into two parts. Since the code is divided into two parts, Front-end and Back End can be developed at the same time. The progress of the front end and the back end does not affect each other. In other words, it could increase resources for the front-end and back-end at a separate pace since the back-end would need to be ramped up at a relatively faster pace as it grows. In our design, we spend more time on front-end design because front-end work in our app is not dependent that much on back-end work. The code of the front-end can be tested and potentially deployed whenever they're done without waiting for the back-end, which means the front-end worker doesn't need to wait for the backend worker to test the code. In this way, we can have better time management for development of our web app.

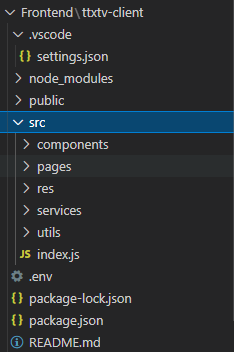
2. RESTFUL API

In our design, we are using RESTFUL api for all of our backend. Resful api is an application programming interface that conforms to the conformers of REST architectural style and allows for interaction with Restful web services. The RESTFUL API provides us with a great deal of flexibility. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats and even change structurally with the correct implementation of hypermedia.

3. MVC: Model-view-controller

MVC is a paradigm, it is a software design pattern which is normally used for developing user interfaces. It will divide the program logic into three interconnected elements. So it allows us to factor out components of a software application and update them in an easier way. We are using MVC design for the front-end of our web application, which makes it easier for us to update our code.



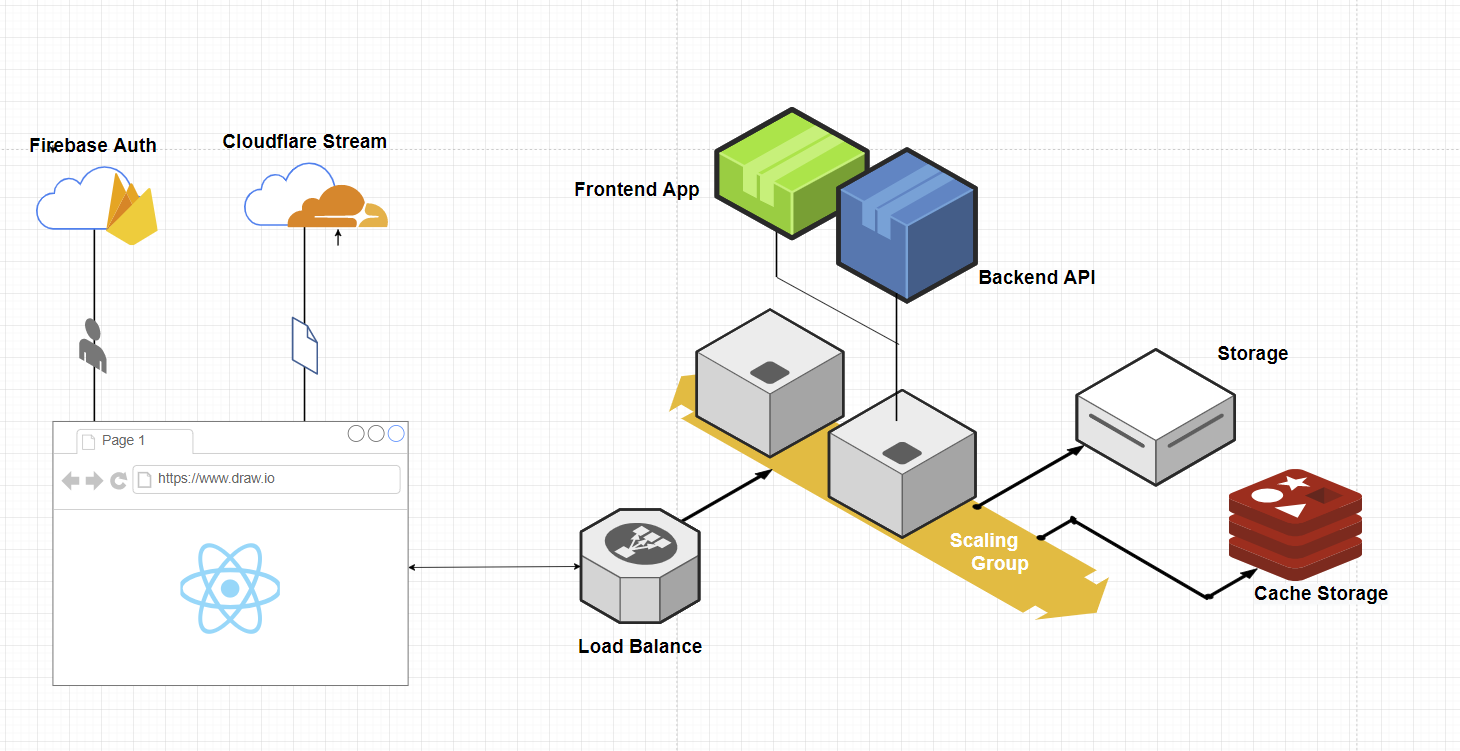
front-end table

4.NoSQL (firebase realtime database)

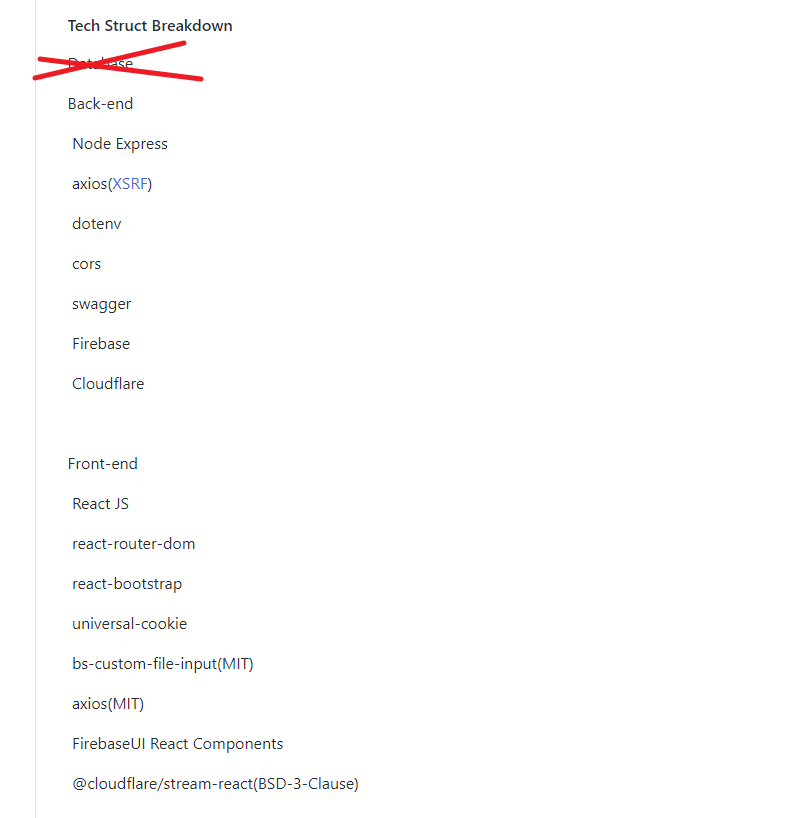
NoSQL database is referring to non-relational databases. It will provide a mechanism for storage of data. Unlike relational databases , they do not guarantee the ACID characteristics of relational data. NoSQL is a brand-new database revolutionary movement. NoSQL databases are often more scalable and provide superior performance. In addition, the flexibility and ease of use of their data models can speed development in comparison to the relational model, especially in the cloud computing environment. Its supporters advocate the use of non-relational data storage . In our web app, we are using firebase realtime database as our NoSQL database to store the user’s information and the uploaded video. We found that compared to SQL database, NoSQL database is faster, and it is way easier to handle. And there’s no cost for our web application by using NoSQL databases Service.

5.Promise based HTTP connection

An http client is very useful when we need to send or receive requests from other servers. As we all know, callback is an old concept. So we need an http client which is promise-based. In our web app, we are using Axio as our promised based http connection. Along with Node.js, Axios has the widest compatibility among browsers i.e. Firefox, Chrome, IE, Edge, Safari for sending requests, which is a big benefit. Moreover, it handles all types of request body in post request.



This is the ideal deployment with load balancing.



There is middleware on our front-end and back-end.

**IV. RESULTS, FINDING AND ANALYSIS**

1. **Results**

As the development of the web app was close to the end, we conducted 5 tests on different features/parts for our prototype following our Testing Plan. Here is the tests and results:

1. **Name: User Login**
2. Purpose:  
   □ Login user’s account and enable user’s further operations.
3. Preconditions  
   □ User has registered an account and entered the login page.
4. Input parameters or manual operations  
   □ User will have its account and password prepared and input in the text input form.  
   □ [Optional] User can check the “remember me” checkbox.  
   □ Users will click the Login button to attempt the login.
5. Results

□ Client received an HTTP response from backend API in a promise function.

* □ Status code of the response was 200.
* □ The HTTP response included a JSON format return value from the API (result{}, success, errors[], messages[]).
* □ Login succeeded, and a token was stored.

1. Met criteria

□ **Given** account name and password were inputted in the text entry box.

□ **When** I clicked on the login button. **Then the client** attempted to log in. The client stored the token and the login status then redirected to the home page.

2. **Name: Room Creation**

1. Purpose:  
   □ Communicate with the backend API, create a room for users to share their contents.
2. Preconditions  
   □ User has a login and enters the room management page.
3. Input parameters or manual operations  
   □ Users will provide a room name and an invite code in the text entry form.

□ Users will click the create button to create a room.

1. Results

□ Client received an HTTP response from the backend API in a promise function.

□ Status code of the response was 200.

□ Creation succeeded (Status code 200), the room was added to the list and displayed, an invitation link was sent to the user's clipboard.

1. Met criteria

□ **Given** room name and invite code entered.

□ **When** I clicked on the create room button. **Then** an HTTP request was sent to the backend, the backend attempted to make a room. The room was added to the client and an invitation link was sent to the user's clipboard.

3.  **Name: Media Upload**

1. Purpose:

□ Communicate with Cloudflare Stream API, store user’s upload media.

1. Preconditions

□ User has a login and enters a room, upload function unlocked.

1. Input parameters or manual operations

□ Users will select a media file from the browser’s file selection dialogue (with format filter of ‘mp4’).

□ [Optional] User can input a name in a text form.

□ Users will click the upload button to start uploading.

1. Results

* □ Client received an HTTP response from the API in a promise function.
* □ Status code of the response was 200.
* □ The HTTP response included a JSON format return value from the API (result{}, success, errors[], messages[]).

1. Met criteria

* □ **Given** there is a media file prepared and selected.
* □ **When** I clicked on the upload button. **Then** the file attempted to be uploaded and returned JSON data from API after the upload process.

4. **Name: Room Joining**

1. Purpose:

□ Communicate with the backend API, join the user to the specific room so that it can view shared contents in the room.

1. Preconditions

□ User has logged in and entered the rooms page, also an invitation link has been received from another user.

1. Input parameters or manual operations

□ Users will click on the join new room button.

□ Users will input the room id and invite code from the invitation link.

□ Users will click the join button to try joining.

1. Results

* □ Client received an HTTP response from the backend API in a promise function.
* □ Status code of the response was 200.
* □ Joining succeeded (Status code 200), client received JSON data of the room and contents were displayed on the screen.

1. Met criteria

* □ **Given** room id and invite code entered.
* □ **When** I clicked on the join room button. **Then** an HTTP request was sent to the backend, the backend attempted to join the user to the room. The user was joined and room data was returned and displayed on the client.

5. **Name: Media Streaming**

1. Purpose:

□ Stream media from Cloudflare Stream, let users view media stream together in the same room.

1. Preconditions

□ User has logged in and entered a room.

1. Input parameters or manual operations

□ Users will click on a media content in the content list of the room.

□ Users will click on the play button to start receiving the stream.

1. Results

□ Client received a media stream from Cloudflare Stream.

□ The player component parsed the stream and displayed the media.

1. Met criteria

* □ **Given** that I entered a room.
* □ **When** I selected a media in the content list of the room.
* □ **Then** The media was streamed to my client and ready to play.

**2.** **ANALYSIS**

The five tests above were conducted successfully and the results were as expected with little error. Our web application can fulfill the software requirements and provide users with good customer experience. By far the web app can function as expected but more tests for other parts/features of the web app shall be conducted in the future.

**V. CONCLUSION and RECOMMENDATION**

Overall, our project was successful as most of the requirements from SRS were fulfilled and most of the features of our web application from SDD were able to be implemented with little error. In conclusion, we were satisfied with the results after successfully conducting our planned tests, and were glad that our TTX TV can actually run and provide users with the functions we set in our goals of this project. We believed that TTX TV can finally help people have some more in-person-like social interactions and have fun with their friends closely in this very special time in human history. As for the future, we look forward to being able to integrate the IoT into our web app which was believed to be the next big ongoing evolution on technology and lifestyle. If IoT function was developed, our app can provide a seamless watching/sharing experience for their family and friends in different locations/scenarios.

During the progress of developing our web application, we found teamwork is an important factor that helps us succeed. The word ‘teamwork’ means working together as a team to achieve a particular goal. As we know, humans are social animals. People tend to stick together and work along with mates in their life. No matter in home or office, people need to communicate with others. During the communication, we could improve our skills of being a good member in a team and make us a part of the team we are in. We are appreciated for the teamwork we have done during developing the TiaoTiaoXiong TV project. Each team member did a good job to communicate with each other. We come close to others and bond, and we learned bunch of useful things from teamworks.

**VI. Source Code**

Because there’s too many files. Source code will be uploaded as a zip with build instructions in the canvas.