WeTube

Team 1

Colin Woodard, Egan Bower, Michael Capelotti, Hunter Hults

Software Design Specification Document

Version: (2)

Date:(05/09/2019)

Table of Contents

1 Introduction

WeTube is an Android application that will enable users to watch YouTube videos together, syncing the feed and providing live chat functionality.

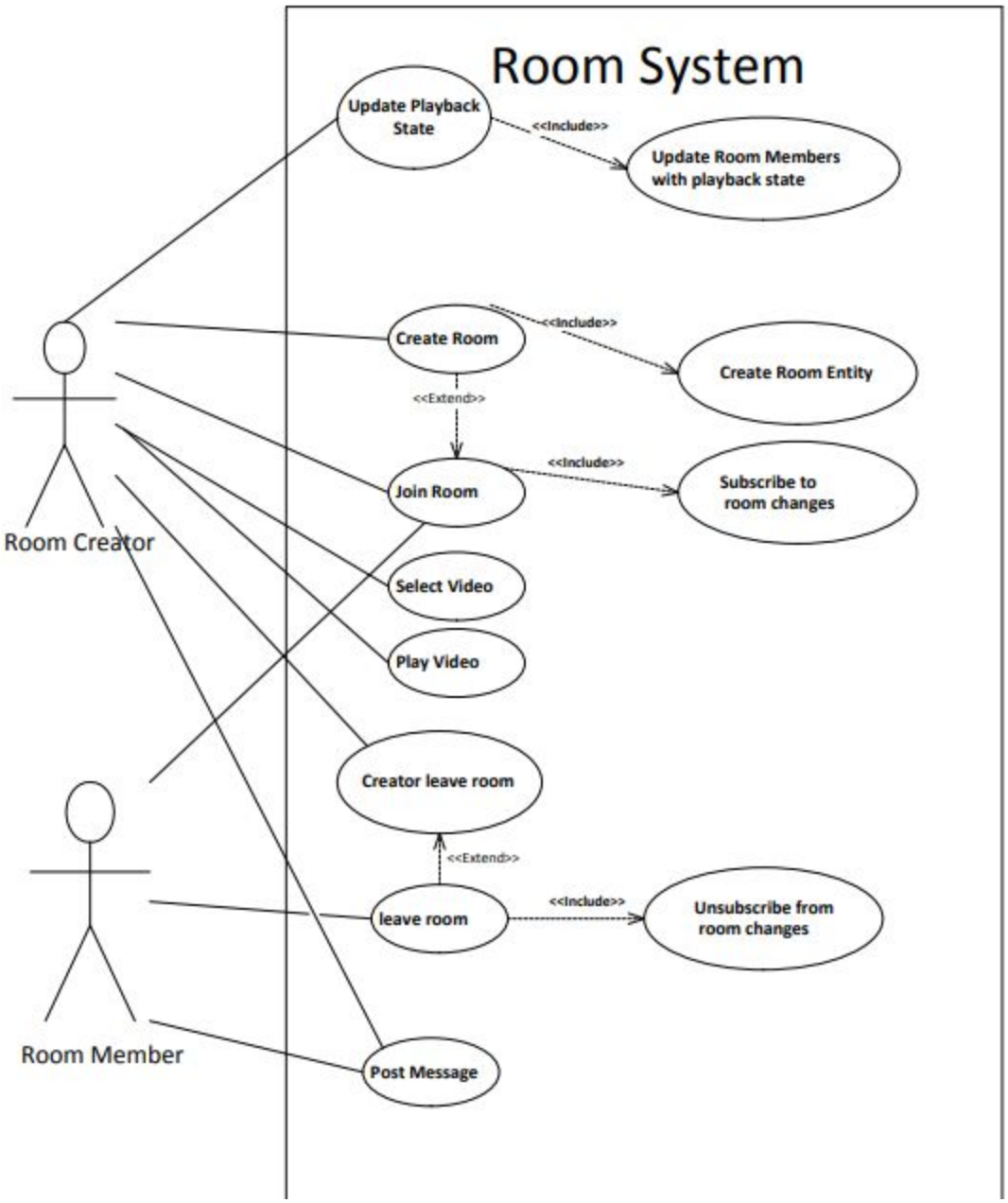
This document describes all data, architectural, interface and component-level design for the software.

1.1 Goals and objectives

Our goal is to create a user friendly Android application that allows users to set up YouTube video viewing rooms, and join other's rooms where they can all watch videos and chat together.

1.2 Statement of system scope

The software will use a FireBase Realtime Database to store room information, manage chat functionality, and provide state updates for synchronization, so that the app can make sure video times are synced.



1.3 Reference Material

N/A

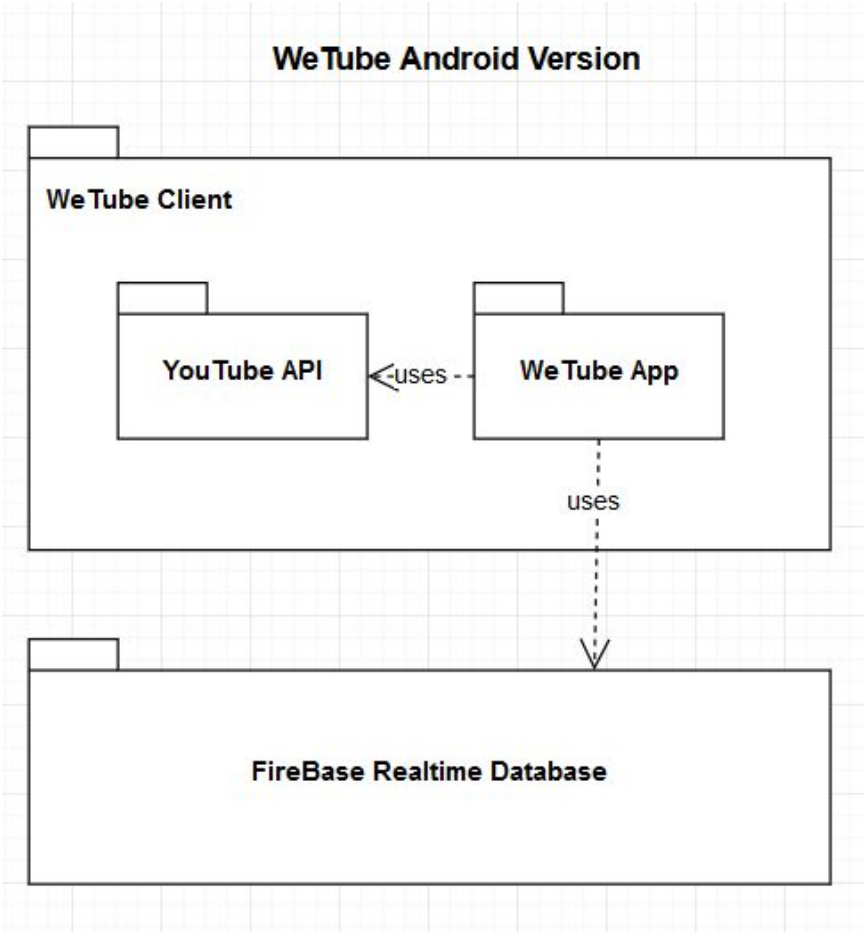
1.4 Definitions and Acronyms

Room - A lobby that has a YouTube video player and chatbox that multiple users can join at the same time.

2 Architectural design

2.1 System Architecture

WeTube uses a Client-Server architecture. The client will use the YouTube API to retrieve video and handle video controls, and communicate with a FireBase Realtime Database that handles synchronization, chatting, and storing room information/video state



2.2 Design Rational

We chose to use a Client-Server architecture as it felt like the simplest way to achieve our desired goals. We can have a single server serving multiple clients, which all need to communicate with each other, making Client-Server the best pick.

3 Key Functionality design

3.1 Video Synchronization

3.1.1 Video Synchronization Use Cases

The Room Owner updates the playback state (either pausing, playing, or selecting a specific time) and the app will synchronize the owner's state with all other room members.

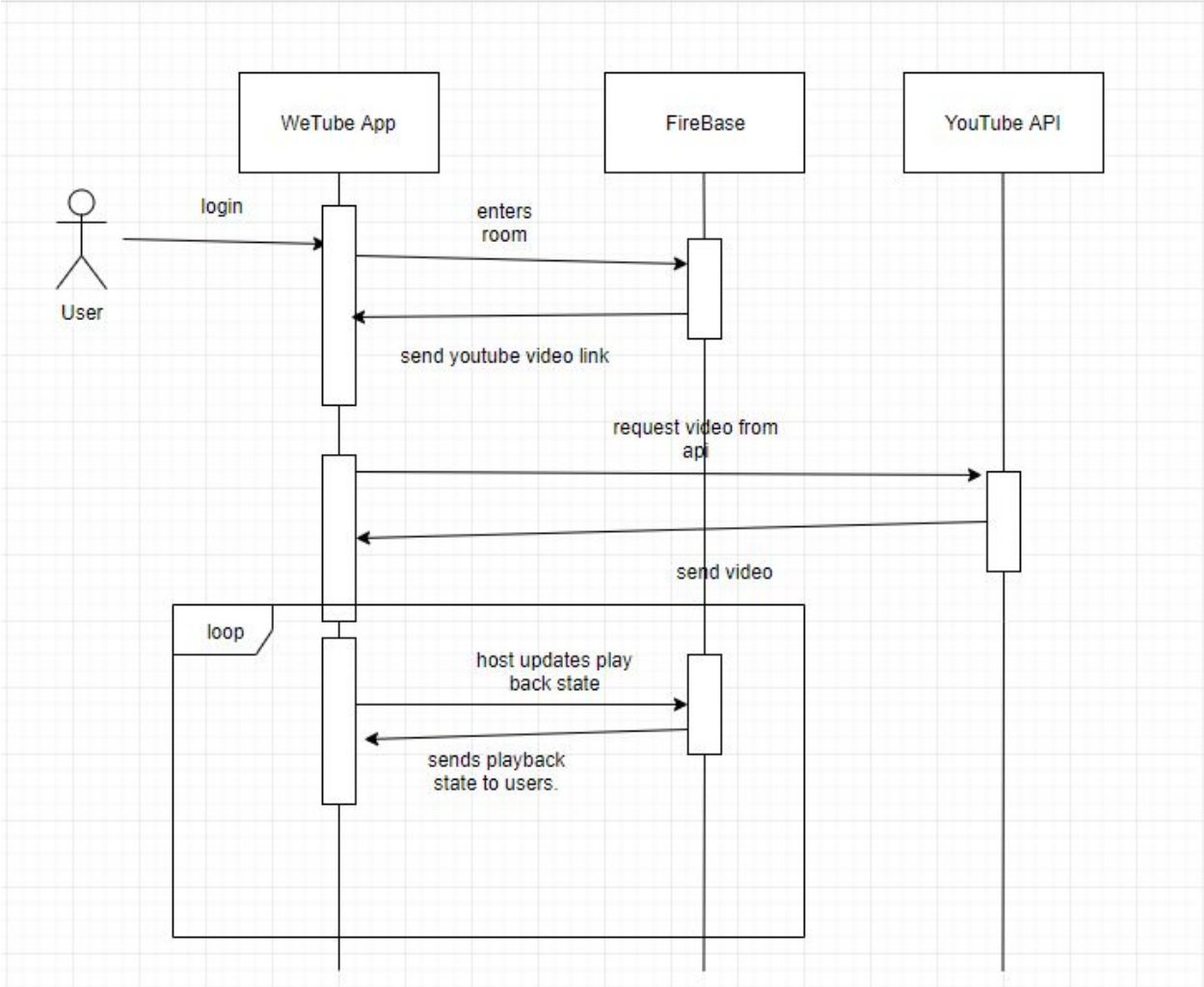
Room Owner updates playback state
Goal in context: Room Owner wishes to change the playback state of the video (pause/play/select time)
Scope: primary system
Level: primary
Primary Actor: Room Owner
Preconditions:

1. A room is created
2. The user is the owner of that room
3. There is a video playing
4. The user has permission to update the playback state

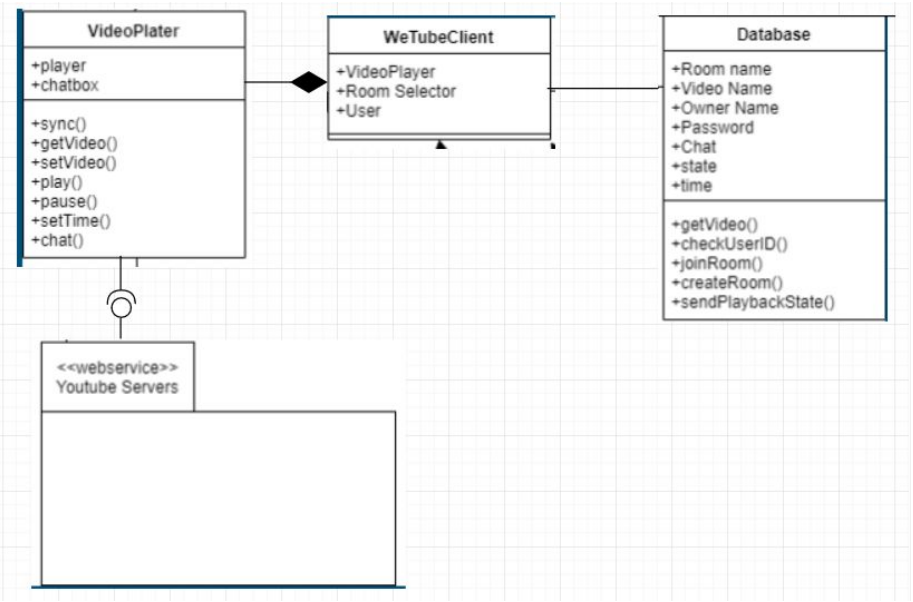
Minimal Guarantee: options to update the state appear
Success Guarantee: the user updates the playback state
Trigger: selecting the update playback options
Success Scenario:

1. Selects the update playback options
2. The video playback state updates
3. The client sends state information to Firebase
4. Firebase sends state information to all users in the room
5. All users in the room have their states updated
6. The video playback state can be updated again

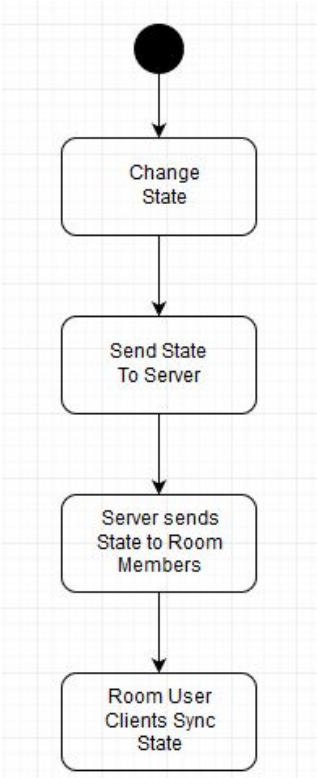
3.1.2 Processing sequence for Video Synchronization



3.1.3 Structural Design for Video Synchronization



3.1.4 Key Activities



3.1.5 Software Interface to other components

This component sends state information to other clients by updating the FireBase Realtime Database, and each client will communicate with the YouTube api in order to sync their playback state with the state information retrieved from the other clients.

3.2 Room Access

3.2.1 Room Access Use Cases

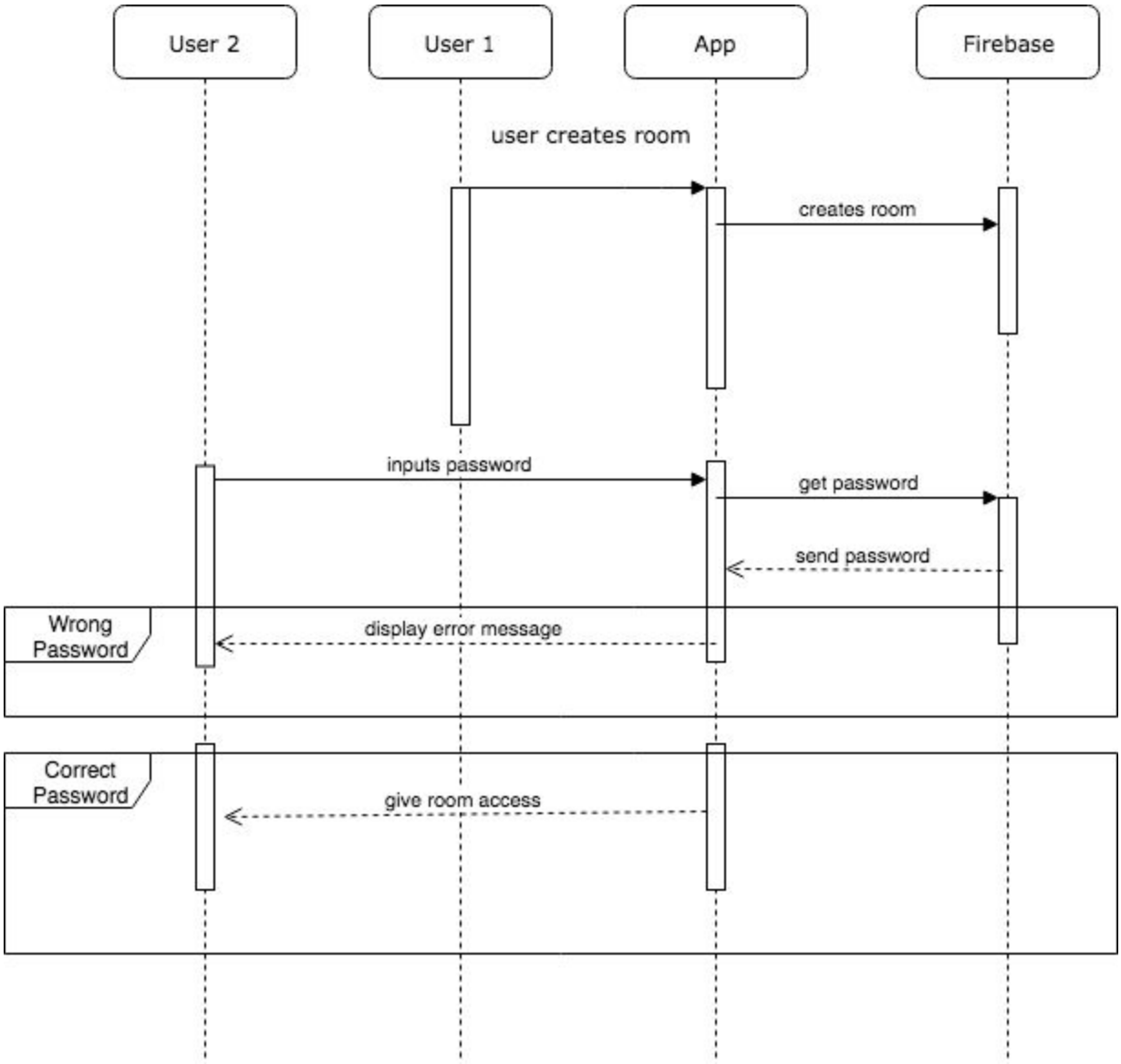
User enters room
Goal in context:User wishes to join video room
Scope: primary system
Level: primary
Primary Actor: Anyone
Preconditions:

1. A room is Available
2. The user has password for room (if required by room)
3. There is a video playing

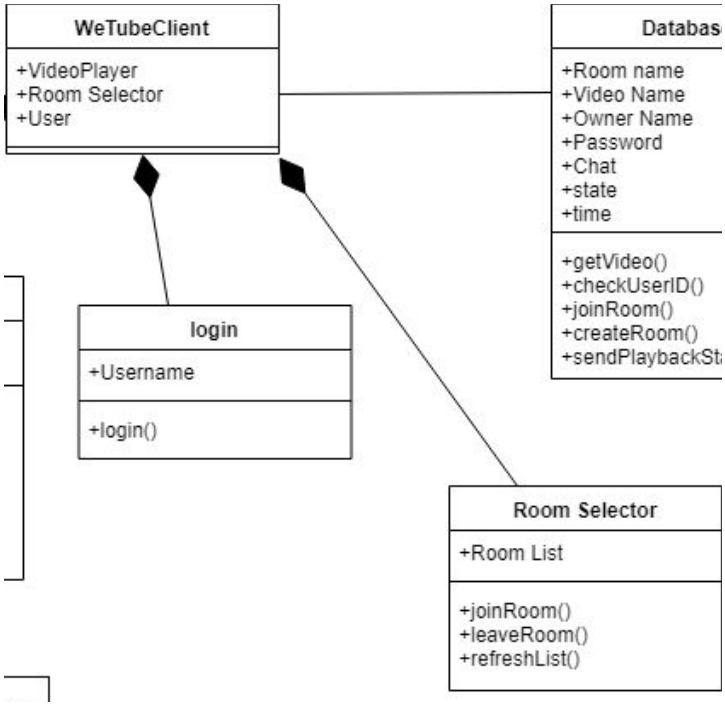
Minimal Guarantee: option to join room
Success Guarantee: the user joins room
Trigger: selecting the room to join
Success Scenario:

1. Selects the room to join
2. Enters password
2. Enters room
3. The video is set to the playback state of host
4. The video updates when host updates video playback state

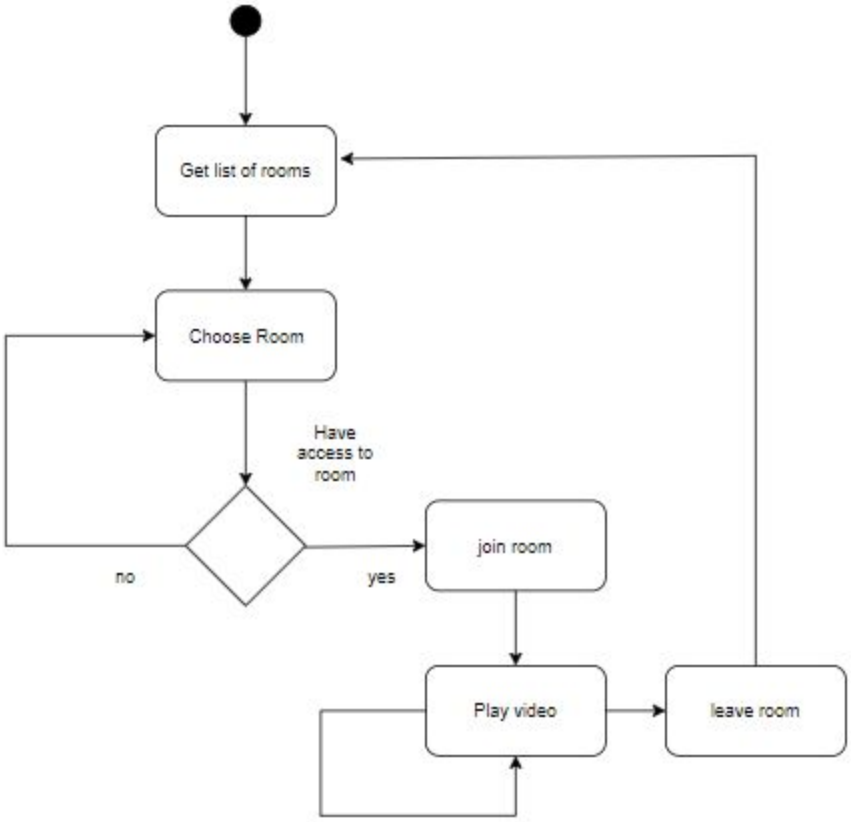
3.2.2 Processing sequence for Room Access



3.2.3 Structural Design for Room Access



3.2.4 Key Activities



3.2.5 Software Interface to other components

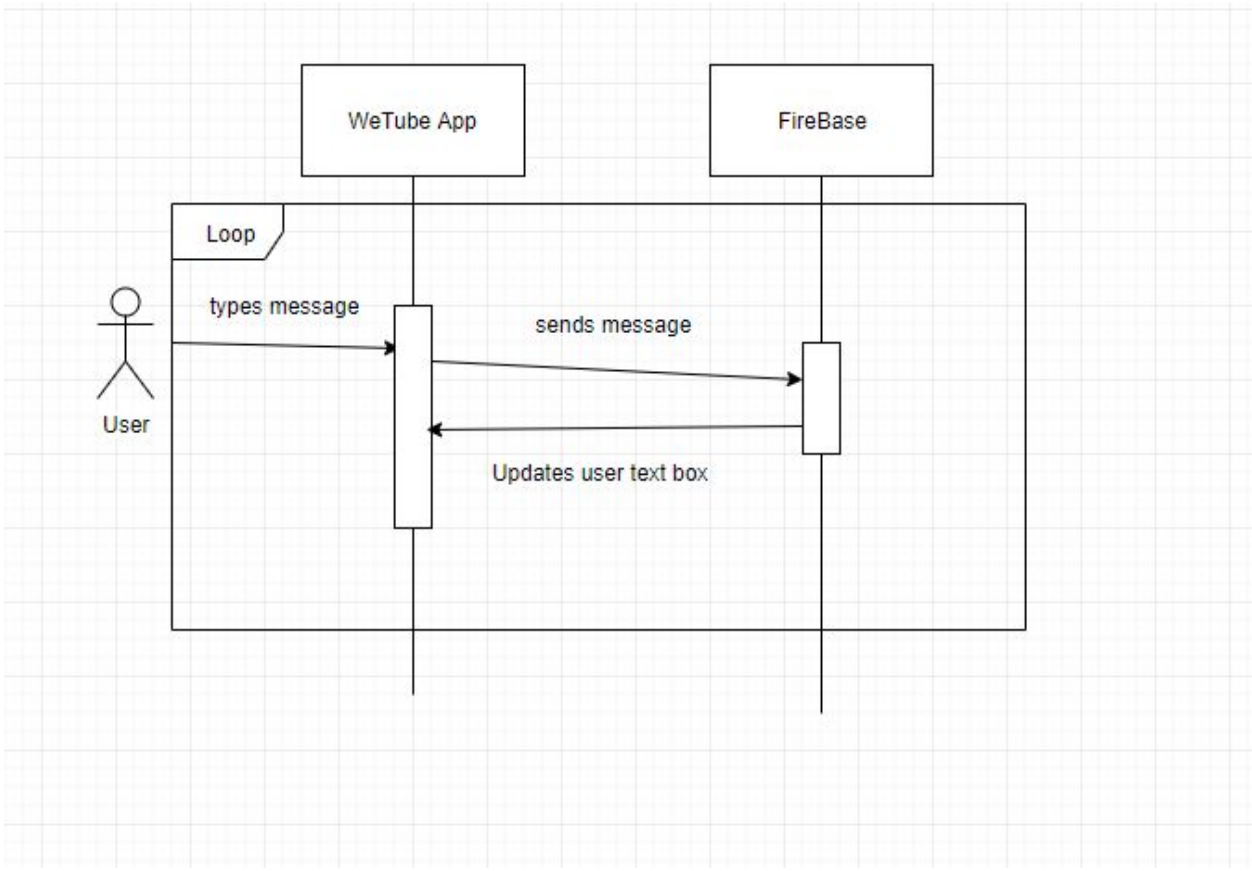
This component communicates directly with the FireBase Realtime Database to retrieve room information.

3.3 Chat

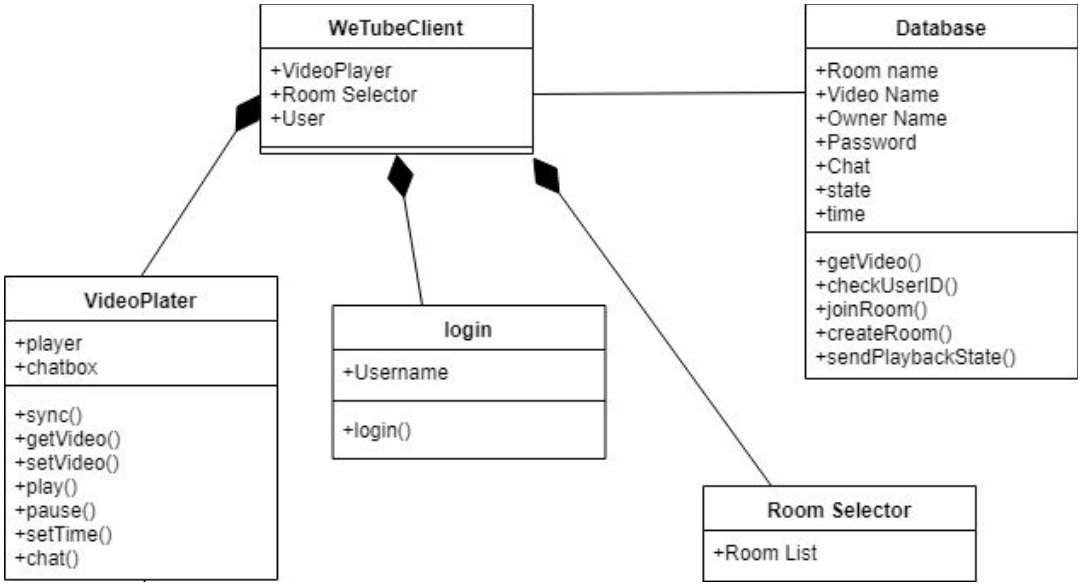
3.3.1 Chat Use Cases

The user will be able to access a chat function between the members of the room they are currently in. This chat is only open to users that are within the room the chat is centered. Each room will possess their own chat.

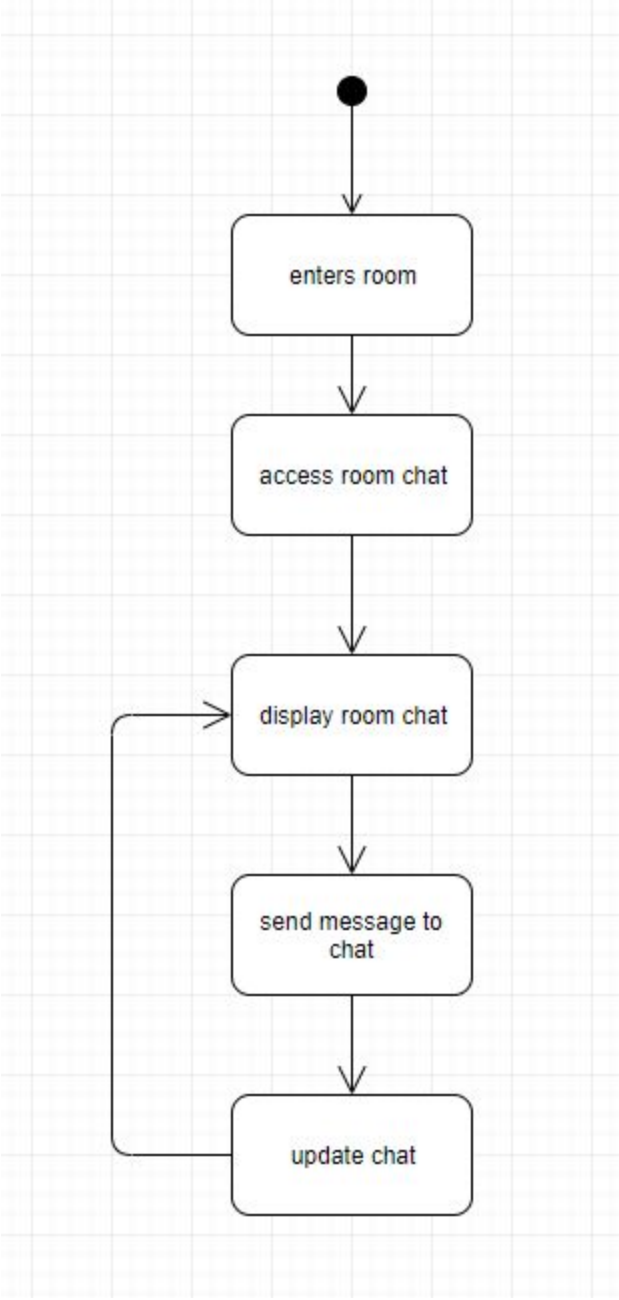
3.3.2 Processing sequence for Chat



3.3.3 Structural Design for Chat



3.3.4 Key Activities



3.3.5 Software Interface to other components

This component communicates with the FireBase Realtime Database to stores/retrieve Chat Information.

4 User interface design

4.1 Interface design rules

The only standards that we are currently following are to conform to the general color scheme of the YouTube website www.youtube.com

4.2 Description of the user interface

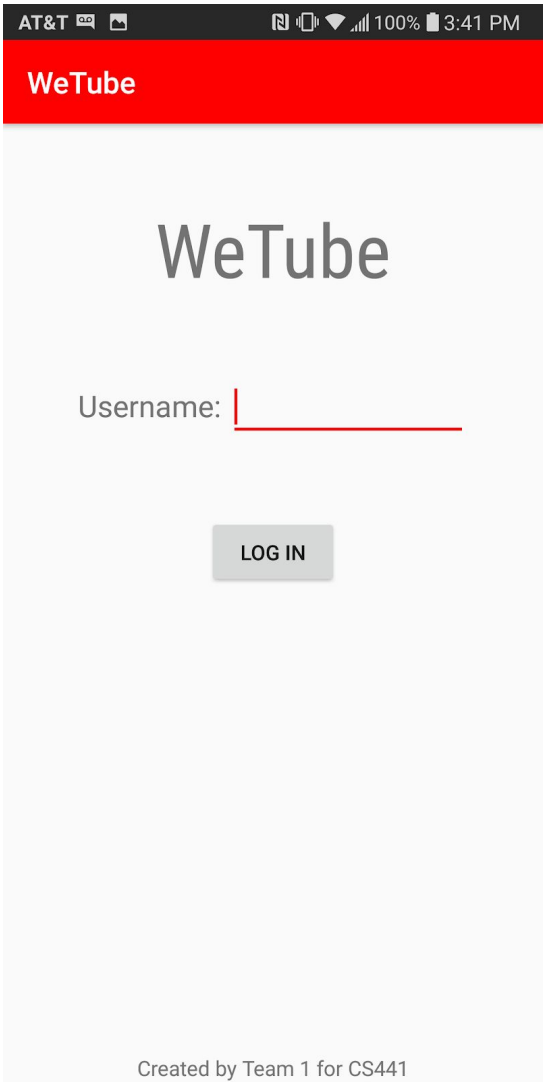
WeTube is an Android application with three activity screens: Menu screen, Room Select screen, and Video Viewer screen.

4.2.1 Main Menu Page

The main menu page allows a user to pick a username for their session, and move to the Room Select screen.

4.2.1.1 Screen Images

Shown here is the release version image for the Main Menu screen



4.2.1.2 Objects and Actions

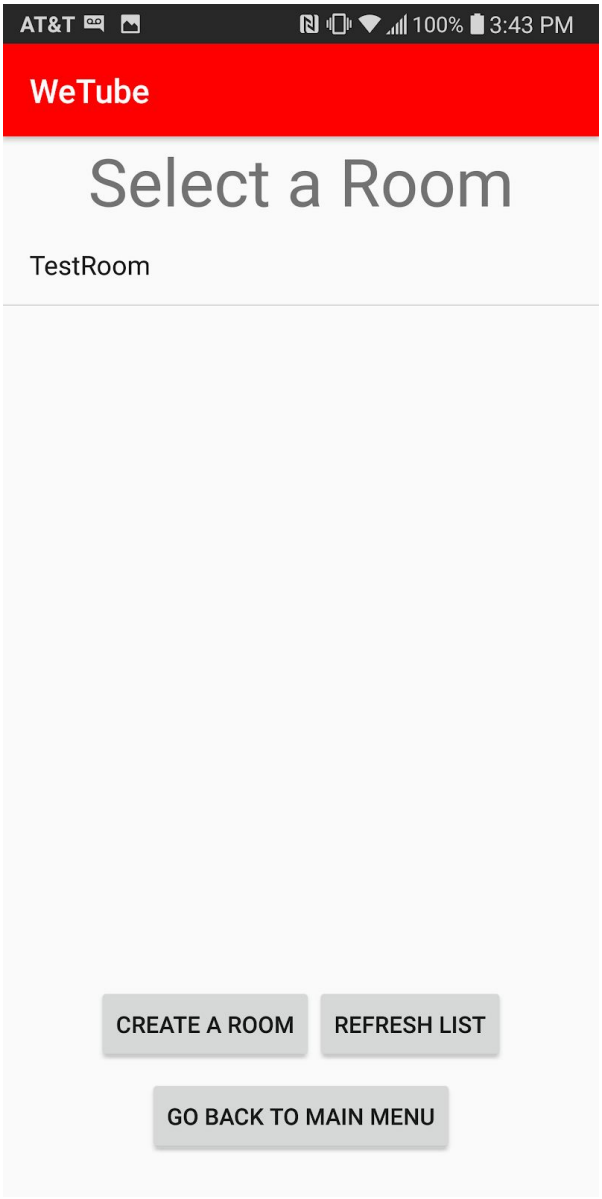
The main objects on this screen are the Username entry box, and the Log In button. Users can enter a custom username before creating or entering a room to be used in the chat system, if no name is written then a guest id is automatically given. The Log In button will transition to the Room Select screen where the user can either select or create a room (seen in 4.2.2).

4.2.2 Room Select Screen

The Room Select screen displays a list of available rooms to join, alongside a button to create a new room, refresh the room list, or go back to main menu. Selecting an existing room, or creating a new one will then move you to the Video Viewer screen

4.2.2.1 Screen Images

Shown here is the release version image for the Room Select screen



4.2.2.2 Objects and Actions

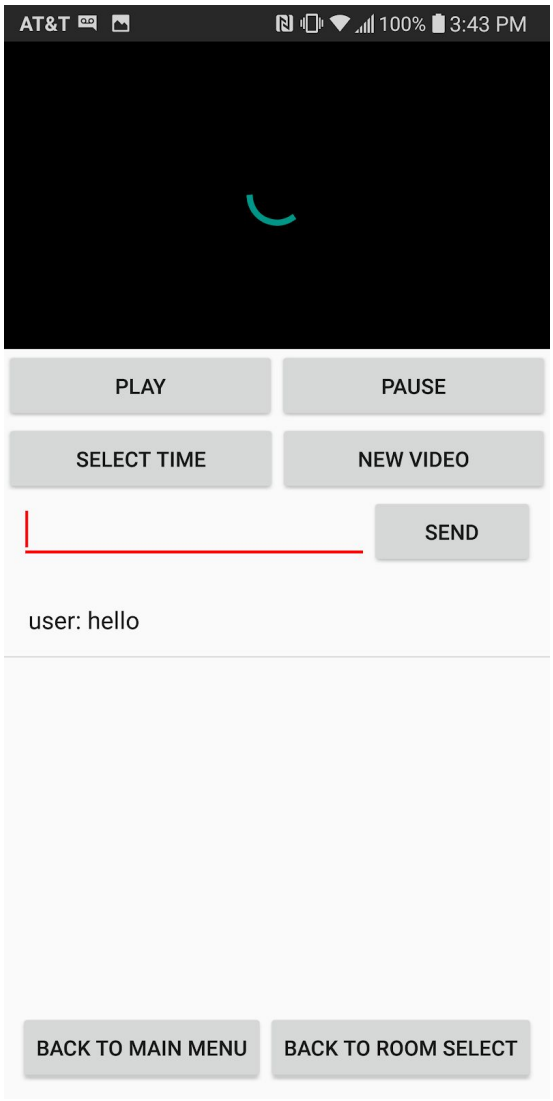
The main objects on this screen are a list of currently existing rooms, each being its own button that will ask for a password before moving the user to the Video Viewing screen connected to that room. There is also a Create Room button where the user can start a new room, a Refresh List button to refresh the list of existing rooms. and a Back to Menu button that will return the user to the Main Menu screen.

4.2.3 Video Viewer Screen

The Video Viewer screen includes a video player, video control option buttons, and a chatroom for users connected to a room to chat with each other. There are buttons to leave the room and move back to the Room Select screen or the Main Menu screen.

4.2.3.1 Screen Images

Shown here is the release version image for the Video Viewer screen



4.2.3.2 Objects and Actions

The main objects on the Video Viewer screen are: the Video Player, playback option buttons, chat room, leave room button, and main menu button. The Video Player communicates with the YouTube api to display video. The playback option buttons are used by the room owner to update playback state of the video and sync their state with the other room members. The

Chatroom allows users to view and leave messages for each other. The Leave Room button exits the room and returns to the Room Select screen, and the Main Menu button exits the room and returns to the Main Menu screen.

5 Restrictions, limitations, and constraints

- Only play YouTube videos
- Tool must operate on android phone.
- Only the creator can play and pause the video
- The information store will be a FireBase Realtime Database
- Users get a new guest name every time they login into the app

6 Testing Issues

Test strategy and preliminary test case specification are presented in this section. Additional test cases are located in the Appendix section in spreadsheet format.

6.1 Types of tests

● Performance Test – ensure that videos are synced within 1 second during playback, ensure that state updates are timely.

● Accuracy Test – determine if the information from the FireBase database is correct.

● User Interface Test – buttons clearly say what they do and UI works on different device sizes.

● Security test – Correct password is needed to enter room, and only room host can update video and playback state

● Repeatability Test – works the same on different devices.

6.2 List of Test Cases

You should document each test case in the following format:

Test Type	**Security Test**
Testing range	Room Selection

Testing Input	Wrong password
Testing procedure	wrong password Click enter
Expected Test Result	Prompt “Incorrect password”
Testers	Colin Woodard
Test result	Passed

Test Type	**Security Test**
Testing range	Video Room
Testing Input	Playback buttons
Testing procedure	Enter room as non-creator Press playback button
Expected Test Result	Prompt “Only the room owner can edit playback state”
Testers	Colin Woodard
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Update playstate
Testing Input	120
Testing procedure	Enter 120 Click select time
Expected Test Result	Video goes to time inputted
Testers	Michael Capelotti
Test result	Passed

Test Type	**Performance Test**
Testing range	Video player in room
Testing Input	pause
Testing procedure	Pausing video and seeing if it pauses globally
Expected Test Result	Video will pause at the same point for every member of the room
Testers	Egan Bower
Test result	Passed

Test Type	**Performance Test**
Testing range	Video player in room
Testing Input	play
Testing procedure	Playing video and seeing if it maintains a 1 second sync between devices
Expected Test Result	Videos on different devices will remain within 1 second of each other
Testers	Colin Woodard
Test result	Passed

Test Type	**Repeatability Test**
Testing range	Different devices accessing the same room
Testing Input	Different devices (Galaxy s8, LG G6, LG G7)
Testing procedure	Enter a room and sync with its content
Expected Test Result	Video is synced with room owner

Testers	Hunter Hults
Test result	Passed

Test Type	**User Interface Test**
Testing range	Different Devices accessing the app
Testing Input	An android mobile device
Testing procedure	Accessing the app
Expected Test Result	The app looks the same on each access
Testers	Hunter Hults
Test result	Passed

Note: you may have more than one test cases for each type of tests. If the tester and test results information is not available, you may add it in the final submission.

7 Appendices

7.1 Packaging and installation issues

The system uses a Firebase Realtime Database in order to sync information between client devices. We chose a very simple database schema, where each room entity is a single node with some child fields. These fields are the following: room name, room owner, room password, video watch code, video time, video playback state, most recent chat message. When a user creates a room they initialize all of these fields, and any subsequent changes they perform update the database. The only data the owner listens for changes in is the most recent chat message, which when changed, the owner's device will add that message to a listview acting as a chatbox. Anyone who joins an already created room sets up listeners on the video watch code, video time, video playback state fields, and most recent chat message. When data in these fields are changed by the owner, any non-owner will react to the changes (if video watch code changes, load new video, if playback state changes, update client playback state, if time is too out of sync, resync, if the most recent chat message changes, add it to the chatbox). This database is obviously not run locally, so as long as the project remains active on Colin's google

account, the app will be functional. Here is a picture of the database with one room entry inside of it.

The app itself requires an android device running OS version 4.4 (KitKat) or higher. To install it one would download the apk and run it on their device.

7.2 User Manual

1. Download the app
2. Enter a username
3. Create or join a room
4. If creating a room set a room password, if joining a room enter the room's name and password
5. The room's creator inserts a YouTube video's watch code.
6. Watch the video with room creator controlling the video's state
7. Leave the room

7.3 Open Issues

- User Accounts/Authentication - We chose to use a guest account system in order to slim down the database and reduce complexity there. We had enough issues incorporating the Firebase Database into the app, and didn't want to add user authentication into the mix.
- Video Searching - We wanted to included an in app search feature in order to find youtube videos, however we did not have the time to develop that, and instead opted for a simple but crude method where a user inputs a video watch code that they find manually. This would be our number one priority if we had more development time
- Room owner migration - Currently, when the owner of a room leaves (by going back to room selection/main menu or closing the app) the room is destroyed and any other

members currently in the room are moved back to the room selection screen. If possible we want to add host migration so that a new owner can be chosen, however that was not possible with our database design.

7.4 Lessons Learned

Third Party APIs/SDKs don't always have great instructions for their services so you have to brute force your way through to figure it out.

The development process for any program is a rocky slope that is full of potholes and roadblocks.

Different considerations have to be made for mobile programming.

7.4.1 Design Patterns

We used a Client-Server pattern between the app and the FireBase. We used this pattern because it was the most efficient and cost effective for our project.

7.4.3 Team Communications

Team communication was rough at first due to miscommunication with personal assignments. By the end we were able to use a system based on communication through online chats and online mutual editing tools such as google documents to work together on reports. We would change how long it took to assign the different teammates their roles so the creation process could have happened earlier.

7.4.4 Task Allocations

All members worked on documentation
Hunter Hults: FireBase creation and integration
Colin Woodard: Application design and implementation
Michael Capelotti: planning, organization, and management
Egan Bower: Testing and proofing

7.4.5 Desirable Changes

Colin:
I would like to develop the video selection/searching feature. Currently the app requires that users find a YouTube video outside of the app, and then copy the video ID into WeTube app. If we had time I would create a pop up box that accessed the YouTube API to retrieve search results, letting users tap on video thumbnails to load videos into the room.

Hunter: I would like to create a system in which when the room creator leaves the room the room is not dissolved but instead the room's owner is shifted to another user.

Egan: I would like to include a video playlist so that multiple videos can be played in sequence rather than just putting each one in after the previous video was finished.

Michael: i would add the choice for public and private rooms and a symbol to indicate them as such. This would allow people who want to allow anyone to join do so without having to give out a password. Also the symbol identifying it as public allows people to know that they can join.

7.4.6 Challenges Faced

Colin: I feel like the hardest task was system design. It takes some thinking to figure out what you want a piece of software to do (specification), but designing the system to conform to those requirements requires you to find third party libraries, apis, etc., and then figure out how to fit all the pieces together inside of your implementation.

Hunter Hults: I think the hardest task was the system design. The amount of third party information and interactions was a hurdle and a major challenge. The largest reason is because we had to redesign our system multiple times

Egan: I would say the hardest would be system design. There was so many issues to connect all the parts together and reworking them to talk to each other properly. The server and database design changed about 4 different time before it finally worked.

Michael: trying to make servers and database with instructions that were out of date or did not work with our system. We tried many different types of server/database options and not all worked with the systems we had and had little instruction on making it work. Eventually working with FireBase be managed to get it to work with the app.