Canvas Team 30
2/13/2020
CS 492 Winter Term
Final Project Proposal

# GetSumGame Proposal

Members:

Tristan Hilbert, Huahua Wang, Yuhang Chen, Dianxiong Zhang

# Introduction and High Level Description

This document covers the design of the app *GetSumGame* prototypes for CS 492. This app works to help users comfortably transition into new video games. It does this through providing Twitch Streams for demos and user ratings for satisfaction. The developers imagine a user which has a free weekend, considering buying a game. The user will use the app to see popular Twitch streams, video game wikis, and critic rating scores. The app's end objective is to provide enticement to buy video games from platforms. The app will use information from provided sources to best entice the user to purchase. While this may seem like adware, for the contexts of a school project this is a worthwhile effort.

# Third Party APIs

The *GetSumGame* application will use two free APIs.

- The Twitch API: https://dev.twitch.tv/docs/api
- The ChickenCoop API: https://rapidapi.com/valkiki/api/chicken-coop/details

## The Twitch API

The Twitch API will query the most popular 10 video game streams when reaching the MainActivity. The results will include the game name and an image link. These will be shown to the user verbatim. The user may select one of these streams to launch the StreamDetails activity. The StreamDetails will use an additional query to provide a stream showing the game. It will also show details based on other API calls and a WebView. The user may then launch the Twitch Application if installed. The Twitch API does allow generalized queries with very little parameters to acquire the most prevalent streams alongside details about those streams like the streamer name and viewer count.

### API Calls

- GET https://api.twitch.tv/helix/games/top?first=10
    - Will provide the Top 10 Games based on viewer count
    - Provided in JSON format
    - Game_id will be used for the next call

- GET https://api.twitch.tv/helix/streams?game_id=29307&first=10
    - Will provide information on streams playing this game
    - Provided in JSON format
    - First decides how many streams you want (Max 20)
    - Additional calls may be made with pagination parameters to get more streams
    - Game_id is the game id from the previous call

**NOTE:** The generated Twitch “Client ID” must be attached to the request header of the request for all API calls. This is separate from the query.

## The ChickenCoop API

This API will provide possible user ratings. This will be shown alongside the Stream Details for a specific game. It could also be used elsewhere. The ChickenCoop API provides Metacritic (0-100) rating scores based on a searched game if it exists. It also provides a review (short blurb) should the group choose to use it. This does not require an API key and is 100% free to use.

### API Calls

- https://chicken-coop.fr/rest/games/Hollow%20Knight
    - Hollow Knight is the example Title queried in this call
    - This will return a JSON
    - Not all present games exist

# App Organization

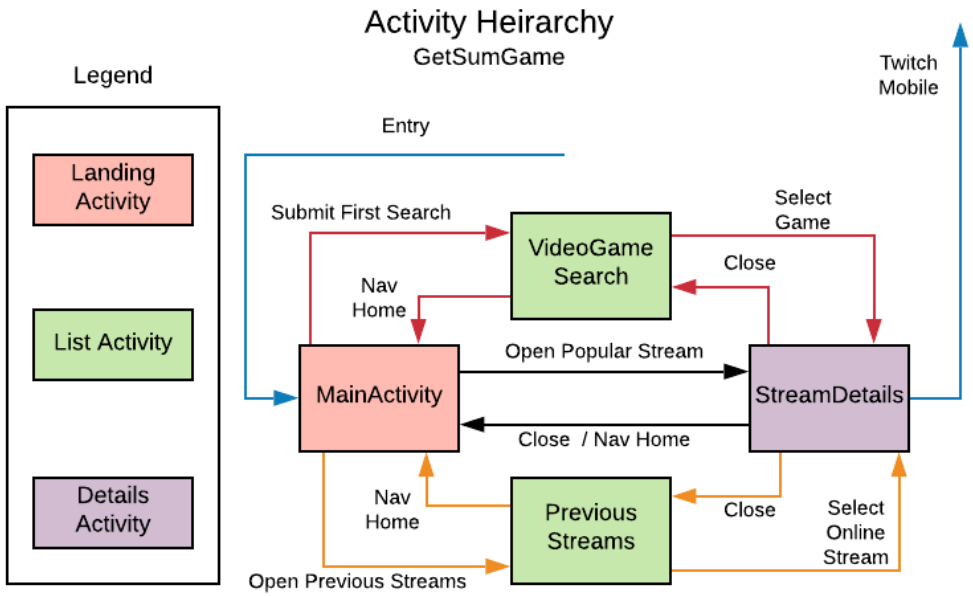
The app will consist of 4 activities.

1. The first activity is a landing page. It will be the *MainActivity* of the application. This should show the popular twitch streams, a search bar, and a button to access previous streams. Pressing on one of the popular twitch streams will bring you to a details page providing information on a suggested stream and some details about the game. The search bar will bring you to a search results page. The button will bring you to an activity which produces a list of previously looked at streams.
2. This details page represents the leaf node of our activities. The details page represents all the details carried over from the previous activity plus some data for the provided stream.
    - It will also carry a link to start a stream in the Twitch Mobile App.
    - There will be a streamer name and a viewer count
    - There will be an image of the game being played
    - There will be a webview with a wikipedia page should one exist.
    - There will be a metacritic review score

3. The PreviousStreams activity will be a list that appears for the user. The previous streams will navigate to a specific details page. The previous streams will need to query for the streamers in the background such that the details page can use the information once ready. The streamers might be streaming a separate game. Note, this will not be the same details page as the streamer might be offline or playing a different game. This will be done through a recycler view and many text views. This will also be the demonstration of the device storage, saving streams after destruction.

4. The VideoGameSearch activity will work like a search results page. The user will navigate here upon the first search and stay here upon multiple searches. Each result will be a possible game to view details on similarly to the MainActivity. The VideoGameSearch activity will not be required to show the images of the games however.

Here is a small diagram depicting the transitions. Notice that any activity outside of the main activity may “Nav Home” by clicking on an icon on the top. There will also be up navigation for closing to reach the previous activity.




It should be noted that an up navigation could either bring a user to the previous activity or the main activity. This should result in a “Home” Icon button and a “Close” button at the top.

# Technical Descriptions of Features

## WebView with Wikipedia

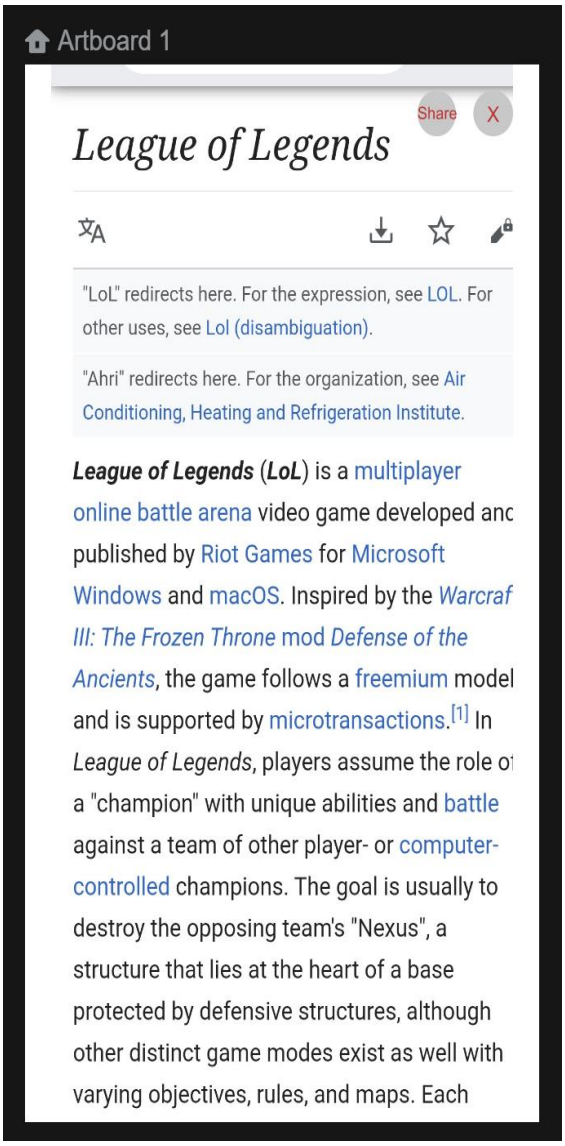
The application must contain an additional feature not shown in class. There are ideas for Android Workers for advanced notifications or a camera to set a user profile. To remain simplistic, the group wanted to instead add on to the StreamDetails Activity. Within the StreamDetails activity the group desires to implement a WebView from the Androidx WebKit package. This WebView will be hideable, but will pull from the web upon entering the StreamDetails activity. The WebView will be a wikipedia article based on the following URL: (https://en.wikipedia.org/wiki/[GameName]). If the app gets a page like this: https://en.wikipedia.org/wiki/NotAGame, then it will show the wikipedia page is not available. It should render for games liked Hearthstone: https://en.wikipedia.org/wiki/Hearthstone. This WebView will be in the activity. It is not an intent like the Twitch Mobile link.

# Denied Features (What the app will not do)

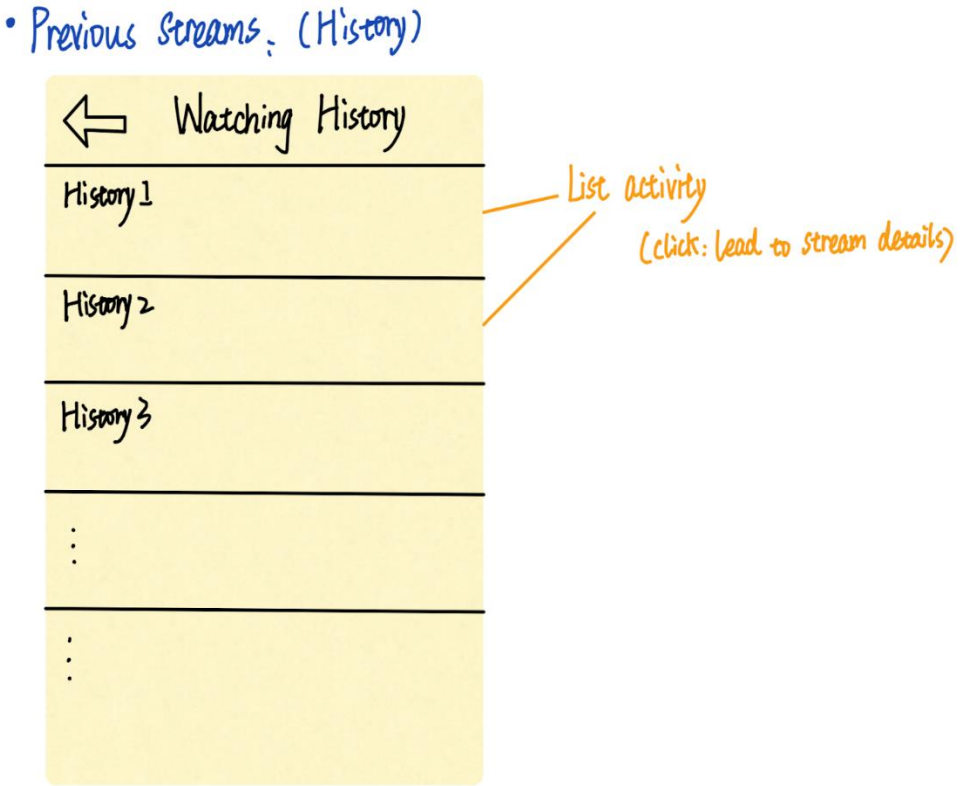
The application will not search all games in existence. The app will be constrained to online collected Twitch Streams. If a Twitch Streamer is not playing the game, then the app will not show that game. The application will also not show all the information for every live Twitch Stream. It will handle missing data. For example, as of writing this document, *Escape from Tarkov* does not have reviews. The app would show the screen for the stream without a user review score. The point of this application is to succeed in CS 492, not to be a fully complete and industry standard application.

# Mock-ups

## Mock-up of WebView




Mock-up of PreviousStreams



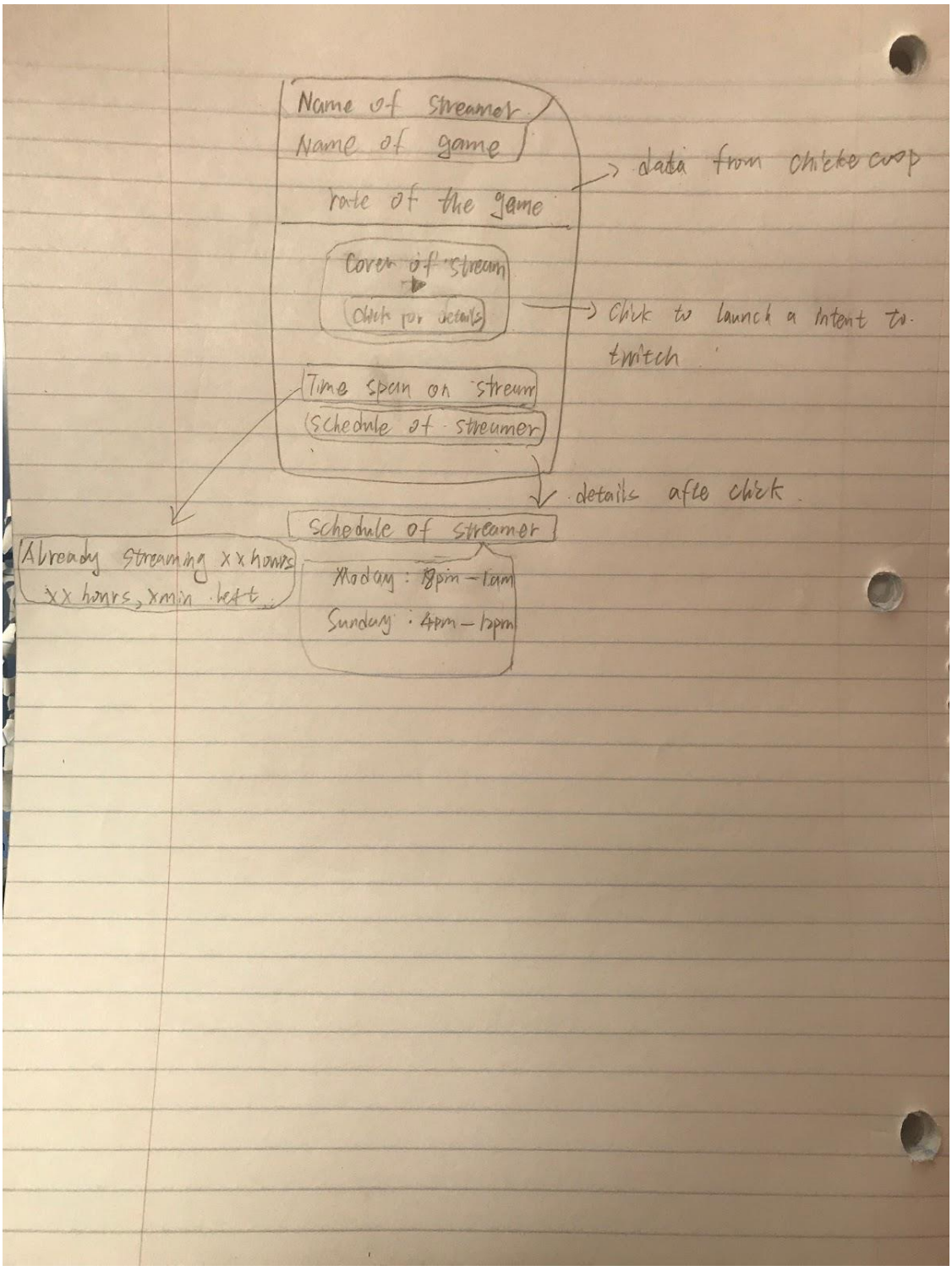

Mock-up of MainActivity

# MainActivity Mock-up

Tristan Hilbert | February 11, 2020


## Mock-up of StreamDetails:

Note: You must scroll down (slide up) for the Wikipedia WebView.


Name of streamer
Name of game
rate of the game
data from chicke coop
Coven of stream
click for details
Click to launch a intent to twitch
Time span on stream
Schedule of streumer
details afte click
Schedule of streamer
Already streaming xx hours
xx hours, xmin left
Monday: 8pm - 1am
Sunday: 4pm - 12pm