General Idea:

Want to scrape the Roblox website to identify X most popular game and find the most popular youtube videos on how to play the games

Need to find a way to transform data

1. Can extract and add quantitative data on the quality of the Roblox game from the

Want to extract data from the Youtube API on keyword searches

Need to find the best way to scrape the

High level idea of API search -> query the Youtube API to find “%Roblox game% how to” or “%Roblox% tutorial”

Visual

# Project Proposal

Business Opportunity: Roblox is an interactive gaming platform that allows developers to create games millions of users (predominantly children) can play. With a vast and ever growing array of games, first time users might find it a little daunting finding the right game to play and how to play it

Proposal: We will create a service that will allow users to view the most popular Roblox game and display the top youtube tutorials on how to play the Roblox game. We will accomplish this by scraping information from the Roblox site for the most popular game, query the Youtube API for the top 5 videos of the search for each game, and display each game with the top 5 tutorial videos on our service.

Benefit: By creating a service that consolidates the most popular Roblox games and some of the best tutorial videos of those games into one easy to use location, we will create a resource that new and existing gamers will turn to when looking for their next Roblox game to play and have the resources to play the game well.

# Potential Challenges

* Need to find ways to transform the data to differentiate our service from the Youtube API and Roblox site
* Finding the best ways to query the Youtube API
* Don’t know which dataframe we want to use to store our service
  + Different benefits and challenges for various options
    - PGAdmin - a much easier query function, with more space for user error BUT would require a PGAdmin connection to be created in order to use our database **(Have to make sure DB exists in server before starting script)**
    - MongoDB - easier access for the end user but the query needs to be 100% accurate in order to be used
      * A little more difficult for us to load our results properly
* Timing of our beautiful soup object was being created too quickly so when we tried to connect our database to the website, no results were rendered. Need to find a way to delay the soup object
  + Trying time.sleep to delay the soup object creation

Swati - Start the query of the youtube API

Kim - start the scrape of the Roblox site

Alyson - Query the Youtube API

Felix - Document

# Thought process for Roblox site scrape

## Extract

Roblox has multiple categories for that are available for users of the Roblox site to select from with approximately 60 games in each category:

1. Most Engaging
2. Recommended for You
3. Sponsored
4. Up and Coming
5. Popular
6. Top Rated
7. Learn & Explore
8. (ETC.)

We will scrape information from only one category to limit the scope of our project. After discussing which category we want to examine, we focused on the Most Engaging category. Games from the Most Engaging category are the games that will likely draw in the most players, be most enjoyable and lead to the largest usage of our service to learn the fun games.

The Roblox site uses Javascript. When viewing the site, more content is available than is available immediately using a simple scrape. As a result, it was necessary to enable the Web Driver for the browser to allow faster loading speeds to facilitate scraping.

We pulled the URL for the Most Engaging category as our baseline URL. Since we are scraping the Roblox site, we will need to use splinter to browse the website. We created a beautiful soup item with the results of our web scrape. Once prettified, we identified the “game-card game-tile” class as the class that would provide us with the key information about the top games within the category.

Once we have the results of our scrape in the beautiful soup object, we identified the following key information:

* Game title
* Number of users
* Overall rating
* Game thumbnail URL (for later use with website)
* Game link (for later use with website)

When thinking about the end user, we gathered the above information to best inform the gamer using our service to determine which game they want to play. Game name, the image of the game, and the link to said game all are key to let the customer know what game they are looking at. On top of that information, gamers want to know how other gamers enjoyed their experience so including an overall rating will give our users a sense of how enjoyable the new game will be. Lastly, providing the number of users for each game helps balance out the overall rating of the game. A game might be super popular with the users playing, but not many people play could inform our users that the game might not be as great as rated.

The information we gathered from the object isn’t always complete so we filtered our results further to only include the games that had all of the following:

* Game title
* Number of users
* Overall rating
* A link to the game

## Transform

With the finalized and filtered list of game results, we created a pandas dataframe to further manipulate the results. The results of the beautiful soup object kicked out the overall rating of each game as a percent liked vs disliked. The Percentage Approval rating was reformatted to remove the “%” and enable sorting by this figure in the dataframe. The text was converted to a float. Similarly, the Number of Users was reported in numbers up to 1,000, then with a “K.” In order to ensure that the numbers were reflected appropriately and enable sorting by these figures, the strings were transformed to the actual numbers and converted to a float. This resulted in the ability to sort descending by this statistic. The higher Number of Users reflects the relative popularity of the games in the “Most Engaging” category.

With our finalized results transformed to our liking, saved the dataframe to combine later with our results from the Youtube API scrape

This information is useful to our end user, but we want to show the rating on a scale from 1 to 100 and show the overall

Most Engaging category (most people, typically most enjoyable and will lead to a larger usage of our service to learn these fun games).

Roblox gives us a large list of the most engaging, we will only scrape for the **game title, number of positive ratings, number of people playing, game thumbnail (for later use with website) and game link (for later use with website).**

Roblox returns string values so we need to manipulate the data to only include game name and key metrics of the **game.**

Once the service is more fleshed out, we can expand our service to scrape across multiple categories

**[INCORPORATED] Extract: The Roblox site uses Javascript. When viewing the site, more content is available than is available immediately using a simple scrape. As a result, it was necessary to enable the Web Driver for the browser to allow faster loading speeds to facilitate scraping.**

**[INCORPORATED] On each site (based on the Roblox categories), there are approximately 60 games. Although we plan to narrow down to fewer, our scrape included pulling all of the games on the “Most Engaging” page.**

**[INCORPORATED] We pulled the following fields for each game on the page:**

* **Game title**
* **Number of users playing**
* **Number of positive ratings**
* **Game thumbnail URL (for later use with website)**
* **Game link (for later use with website).**

**[INCORPORATED] Transformation: The Percentage Approval rating was reformatted to remove the “%” and enable sorting by this figure in the dataframe. The text was converted to a float. Similarly, the Number of Users was reported in numbers up to 1,000, then with a “K.” In order to ensure that the numbers were reflected appropriately and enable sorting by these figures, the strings were transformed to the actual numbers and converted to a float. This resulted in the ability to sort descending by this statistic. The higher Number of Users reflects the relative popularity of the games in the “Most Engaging” category.**

# Thought process for Youtube API query

## Extract

Once the results from the Roblox webscrape are successfully stored in our dataframe, we leveraged the game names in our results to query the Youtube API for tutorial videos on how to play those games.

We determined that the top 5 results provided by the API would be sufficient for our service as too many videos would be overwhelming to our users. Youtube as a website, must have a large and complex algorithm to determine the top results of any search you enter so we elected to trust that algorithm for establishing which results we pulled from our query. The rough idea was to query the API with the following baseline key word search: f’learn roblox {game\_name}’.

We began our query with a stud game name and key word search. Adopt Me is an extremely popular Roblox game so we used it as our baseline game. By beginning our query with a search with our baseline game and key word search, we established that our connection with the Youtube API was successful. We jsonified the results of our stud query to establish what information our query returns and determine what portions are useful to us.

As Youtube users ourselves, we know some of the metrics we would find most useful when deciding what video to select. We gathered the following information for each video result in our query:

* Video title
* Video duration (publication time)
* A thumbnail of the video for our website
* A URL link to the video for our website

Providing the video titles and durations, users on our website will be able to

want to display to our end users

We are grabbing the top 5 results for tutorial videos for each Roblox game our webscrape returns

Using Youtube’s criteria for top 5 results

Query: f’learn roblox {game\_name}’

One potential issue is that each hit to the API could return different results

What we pull outside of the video is an interesting wrinkle for this project

If we are going to create a website for our service, we should draw out all the things we want to include in the website to guide our scrape

We can update the key words to update the results coming from API query

Thought about making the key words something that the end user would enter to update the results for each Roblox game in our service

Query for each game will include

* The video name
* The video thumbnail
* The video URL
* The publication date of the video

**E**xtract: your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).

**T**ransform: what data cleaning or transformation was required.

**L**oad: the final database, tables/collections, and why this was chosen.