Movie Reviews & Ratings Application

Product Documentation

CS300: Software Engineering
Spring Semester 2018
Professor Thomas

Mom's Spaghetti Code
Kyle Wyse
Nick Smith
Anh Vu
Koki Omori

Requirements

Project Overview

An Android application that anyone can use to search movies and find ratings. The ratings will be based on data from multiple online review websites. The way it works is as follows: A user opens the app and searches a movie, and the app will take them to a new page which contains a synopsis on the film and the movie's review scores. The synopsis would include details like release date, plot overview, actors, producer, … etc., and would be pulled from the web. Reviews from all the most popular review sites would be shown under that. The idea is that this app would combine many of the features of sites like Rotten Tomatoes, Metacritic, and IMDb. This would be an app that consolidates the essential data and makes it more convenient to use than visiting multiple sites.

Business Model

The app would be trying to tap into a similar market as sites like Trivago, but with movies and potentially other forms of entertainment. Many review websites have their own app, but our app would show multiple scores side by side. After implementation, the app would make money through advertisements, and a perhaps a "pro" model which would be ad-free and have more features. There is also the potential that this could expand into being a website which would also generate revenue through ads.

Think Tank Results

We started with an idea for an entertainment review consolidation app. Primarily focusing on movie ratings, we could pull review scores and other info from popular review sites,

giving the user an overview and allowing them to make more informed decisions on what they want to watch.

This app has the potential to expand to different forms of entertainment (e.g. TV shows and video games). But we have also come up with some long-term goals that we would hope to get implemented after the base product. Some of these "reach goals" would be:

- List available services the movie or show is available on (Netflix, Hulu, Amazon Prime Video, etc.), or where it can be bought if it cannot be streamed
- User profiles and forums for users to discuss movies
- Categories of movies (Genre's, Decade, Box Office, Company)
    - A top-rated list
    - Recommendations
    - Must Watch List
- Extra information on cast (similar to IMDb)
- News section for entertainment
- Search by Genre, Director, etc.
- List local theaters and showtimes for movies

Functional Requirements

- Allow user to search movies
- App will pull information on the movie
    - Release Date
    - Director
    - Rating (G, PG, PG-13, R, etc.)

- Pull review scores from several different websites
- Be able to cleanly display this information to the app

Nonfunctional Requirements

- Platform: Android Studio
- Need to connect to the internet through Wi-Fi or mobile data
- Ratings must be available for the desired movie

Use Cases

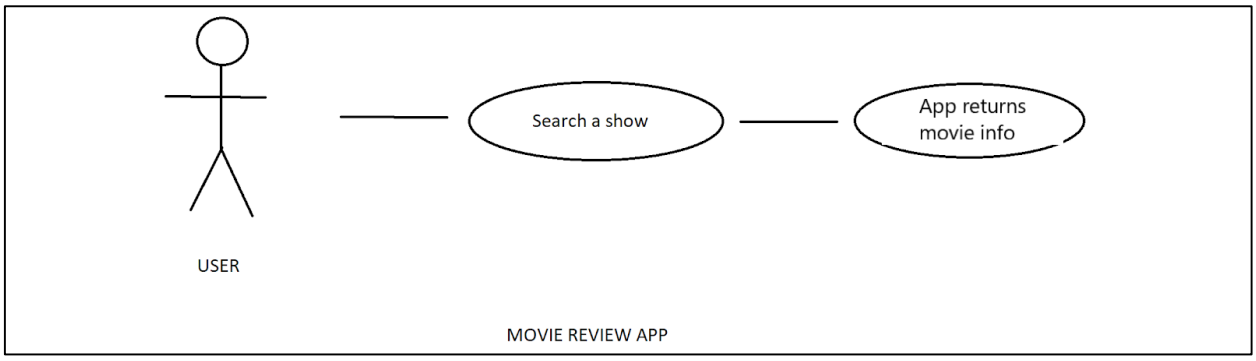


Figure 1.

Case One: A user would like to see information about a movie.

1. The user enters the name of the movie.
2. The app returns relevant movie information.
3. The user may search another movie or exit the app.

Possible Alternatives:

1. The user spells the name of the movie incorrectly.
2. The user enters the name of a series of movies without specifying which entry they mean.

Proof of Concept/Rapid Prototyping

Using an existing web crawler, we tested to see if major movie review sites (e.g. Rotten Tomatoes, Roger Ebert, The Guardian, IMDB, etc.) would permit crawling so that we can retrieve the data we need to display in our app. Nearly all sites permitted such behavior, and so we can safely move into the next stage of development. The web crawler in question, being our own code, will serve as the guide for further development.

Specifications

Overview

A free Android app used to search reviews for movies. Upon opening the app, the user will be presented with a search bar into which they enter the movie title. Pressing enter or the search button will take them to a new screen which displays search results. The user can then use the search bar on that page to search a different title.

Product Functions

The app will maintain a list of URLs which contain review datasets or API databases. With the mobile application, the user will be able to search for reviews by the name, and the app will crawl the sites for relevant review data. The flow of data through the app can be seen in Figure 2 below.

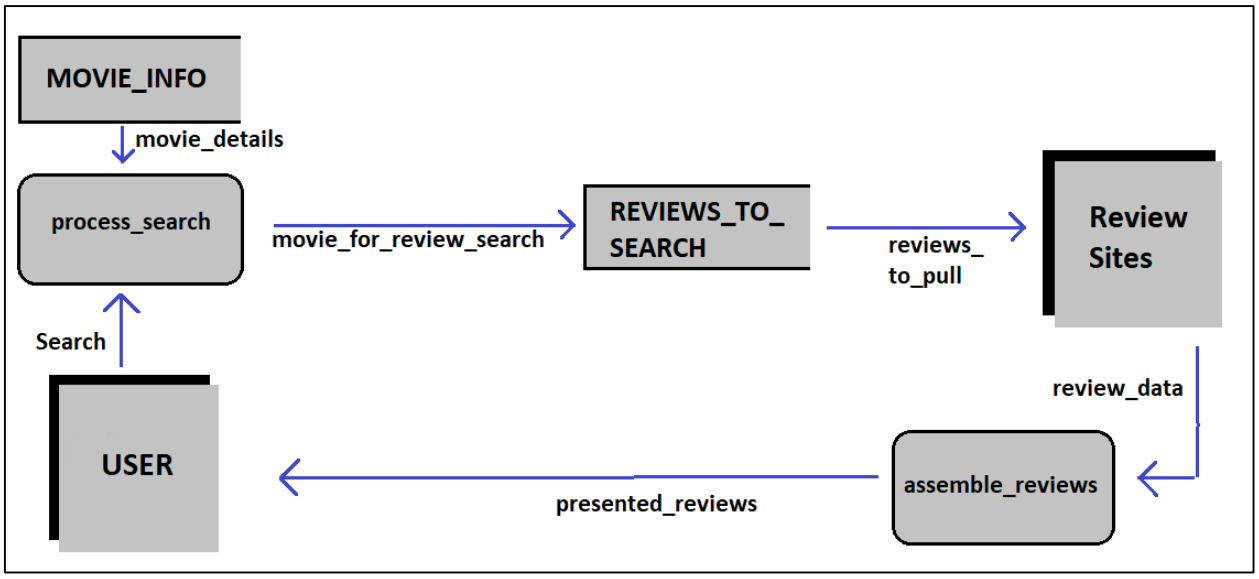


Figure 2.

Layouts

Figure 3: In this photo, we want users to have a view of what they need to do to look for the movie rating. Basically, when you open the app, you will see a search bar above a splash screen. Then, when you click to the search bar, the touch keyboard will appear on the screen and all you need to do is type the correct name of the movie you are looking for.

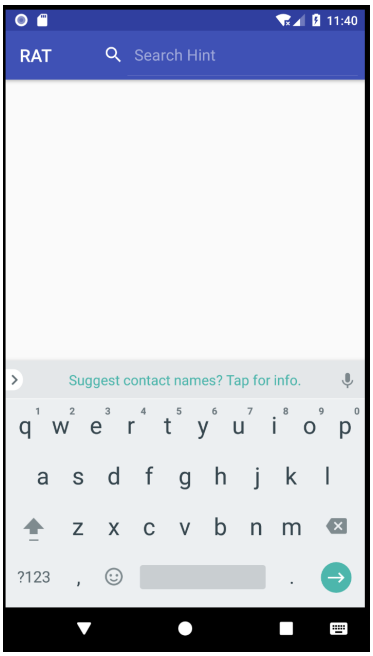
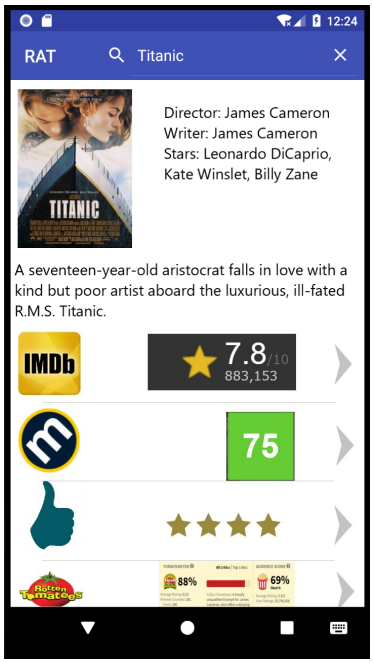


Figure 4: For the second picture, we want to show how the results will be displayed. the app will collect information about the movie at some reputable sites. Then it will show the results according to the data it received originally. For example: if that site is using % unit, the app will show the rating with % unit. The app will not convert into a single unit but will keep the original from the sites. This is the final step that the app will deliver to users. It gives the user some basic information of the movie that user was looking for such as Director, Actor, Writer, the and the plot synopsis.



Parallel running

Parallel running does not work for this project, as we do not have an old system to use for running alongside a new system. It will be a new system incorporating code from a project we have made previously, along with new code necessary to make the application to fulfill its purpose. If parallel running was available, it would slow down production anyways because of the cost to implement it and the maintenance that would go along with getting it to run.

Portability

Since this is an Android app, it will need to be downloaded onto an Android device. The users will need Wi-Fi or a cellular data connection to run this app on their device. Because the app only stores URLs to go to and not whole datasets, it should be very lightweight, but search performance may be influenced considerably by having to go out and fetch data. Another concern is the API data itself, which may come in the form of whole databases which need to be downloaded to search, which would be unrealistic for an app. In this case we may need to use a server to house this data or try to limit the amount of internet we are using. More testing will need to be done to determine the fastest and most lightweight way to implement this feature.

Reliability

The app will be able to gracefully handle many errors from review sites such as 403 and 404 errors and will display whatever data it does manage to collect.

Response Time

This will vary depending on the version of the user's operating system, and how powerful the user's hardware is. Also, a strong internet connection will be required to get a response from the app.

Input/Output Specifications

Inputs:

- User will type into a search box
- User can tap a button to go back from the reviews screen
- User can use search bar to search more movies

Outputs:

- The app will output results from the movie search
- The app will output info on the movie
- The app will output the reviews that it gathered from sites

Acceptance Criteria

This project will not be considered complete until the following requirements have been fully implemented and tested:

- Searching titles
    - A functional search bar will be available to search for movies
    - The search bar will be available in the menu bar on all pages of the app
- Returning ratings and reviews
    - A detailed overview of the searched movie must be available which contains ratings from multiple sites including but not limited to IMDb, Metacritic, Rotten Tomatoes, and Roger Ebert
- The more detailed review page will attempt to include, at a minimum, the following:
    - A plot synopsis
    - A list of major cast members, directors, and producers

- An image associated with the title

Projected Timeline

28 February: Class Extraction

5 March: Object Oriented Analysis Available

12 March: Design Documentation Complete

14 March: Working Search Capabilities Implemented

16 March: Complete List of Review Site URLs and API data

17 March: One Week Hiatus (Spring Break)

26 March: Implementation Deadlines Available

6 April: Layouts Complete

13 April: Working Search Bar

20 April: Data Retrieval Implemented

27 April: User Interface Complete

7 May: Implementation Document Complete

7 May: Final Product Release

Object-Oriented Analysis

Overview

This document serves as a basis for all the classes which will be used in this project. Beginning with a detailed description of the app's functions, all nouns have been analyzed as potential classes. Using this in conjunction with the use cases will provide a final list of classes to be implemented.

Noun Extraction

Concise Problem Definition:

The app will maintain a list of URLs from which to pull API data. A search bar is used to search these sites for movies, returning a list of ratings from different review websites. Selecting a rating from the list will display a new page containing a plot overview, detailed review, director, writer, cast members, and a picture.

Nouns:

App, URLs, API database, Search bar, Sites, Movies, List of ratings, Rating, List, Review websites, Page (Activities in Android Studio), Plot overview, Detailed review, Director, Writer, Cast Members, Picture.

Candidate Classes:

- Search Bar
- Ratings & Reviews
- Page

Use Case Analysis

The only use case is where a user wants to search reviews for a movie. In this case, the user will interact with the search bar and the list of returned ratings, which are part of the user interface.

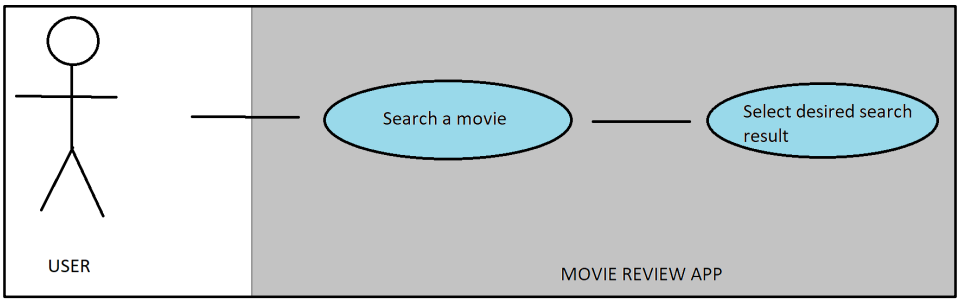


Figure 5.

Brief Description:

Search Move: A user would like to see information about a movie.

Step-by-Step Description of Search Movie:

1. When the search bar is tapped, it should pull up a keyboard that will be typed into. The enter button (on the keyboard) will be pressed to complete the search.
2. Crawl the web to find the movie(s) that is being searched for.
3. Assemble data in order to show reviews and info for desired movie:
4. Basic review data listed (the review scores) in a page, below a list item containing essential info such as movie release date, director, writer, cast, and a plot overview, as well as a picture. Note: see updated layout images (attached).

Class Modeling

Entity Class

- Site Directory

  Attributes:

  - URLs

Boundary Class

- User Interface

  Attributes:

  - Search bar text
  - Search hint
  - Layout
  - Colors
  - Strings
  - Menu
  - Dimensions

Control Classes

- Ratings and Reviews

  Attributes:

  - Fetch Data

- Item Search

  Attributes:

  - Send Search Query
  - Search Results

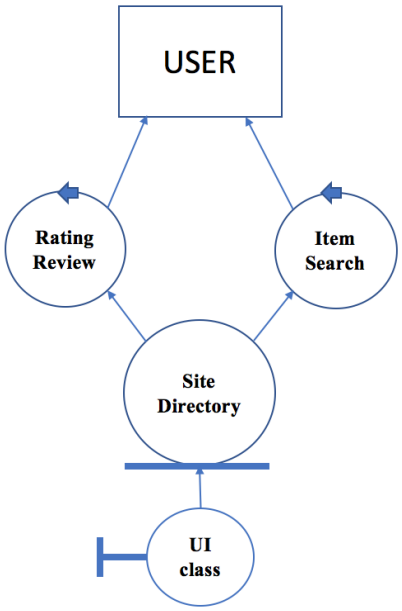


Figure 6.

Normal Scenario

1. User opens app.
2. User presses search bar on the top of the screen.
3. Keyboard appears.
4. User types in the name of movie that user wants to check.
5. The app shows detailed list of results from reputable sites.
6. Press search bar to start new search.

Exception Scenario

1. User opens app.
2. User presses search bar on the top of the screen.
3. Keyboard appears.
4. User types in the name of movie that user wants to check.
5. The app shows list of results from reputable sites.
6. User decides to search a different movie.
7. User presses search bar on the top of the screen.
8. Keyboard appears.
9. User types in the name of movie that user wants to check.
10. The app shows list of results from reputable sites.
11. User taps an item in the list.
12. The app shows more detailed information.
13. Press search bar to start new search.

Sequence Diagram

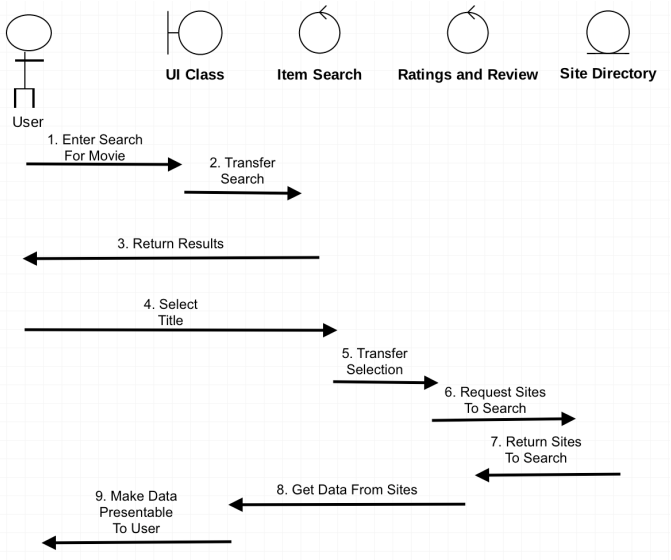


Figure 7.

Use Case Realization

The app is only intended for movie reviews, so there is only one case: searching for movies ratings. Accordingly, the boundary class User Interface will take the information from the user's input and look for the review of the movie that the user chose. There is one entity class, which is Site Directory. It contains all the URLs from which to pull data, as well as any other essential site info. The process of searching/crawling review websites is divided into two classes: Ratings and Reviews and Item Search. These classes will fetch the reviews from sites and send them to User Interface to be displayed for users.

Project Design

Design Pattern

The User Interface for this app will consist of approximately 2 Activities in Android Studio, each with an XML layout, as well as an Async Task. A Site Directory class is used to interact with the URLS maintained by the app. It is called after a successful search for a particular movie. Movie searches are performed by the Item Search class, and review fetching is performed by Ratings and Reviews.

Database Design & Third-Party Libraries

We decided that a database would not be necessary for this project, as we will not be using APIs but rather a third-party HTML parser (jsoup). This makes data retrieval much simpler and allows us to store the review site URLs as a simple list within the code. jsoup also is used for retrieving information on the movie from Rotten Tomatoes and pulling review scores from different sites. Another third-party library being used is Picasso, it is used for viewing images from the web. Using Picasso means that only a single line is needed to load the image, whereas writing our own to do the same job would require a lot more code.

Classes

- Item Search
    - AsyncSearch using jsoup
    - Progress Dialog Box
- User Interface 1 & 2

- Search bar
- Search hint
- Search results

Pseudocode for Ratings and Reviews

```
public ITEM_SEARCH  //NOT an android activity
{
   Document doc; //will contain jsoup results
   ProgressDialog progressDialog; // needed for the async task
   Pattern urlLinkPattern = "http...[any caracters]";
   String rottenTomatoesRating, metacriticRating, ... ;
      // will refer to as rATING_sTRINGS
   String title, summary, ... ;
      // will refer to as oTHER_iNFO
   ArrayList results; // to be returned containing ratings, etc.

   search(String query)
   {
      execute new AsyncSearch();
      return results;
   }

   onCreateDialoge(%)
   {
      create, show progressDiagloge with message "Processing..";
      progressDialog.setCancelable(false);
         // user must wait for search to finish
   }

   private AsyncSearch extends AsyncTask
   {
      onPreExecute()
      {
         progressDialog = new ProgressDialog(0%);
      }

      doInBackground()
```

```
      {
         try
         {
            String movieRating;
            //use jsoup to fetch ratings, reviews, and info
            for (each review-site)
            {
               siteLink = doc.find(review-site);
               if (jsoup.found(siteLink))
                  pull movieRating from siteLink;
               pull oTHER_iNFO from siteLink;
            }
         }
         catch(Exception e)
         {
            print e, call_stack; // for support & debugging
         }
      }

      onProgressUpdate(%)
      {
         show progressDialog(%);
      }

      onPostExecute()
      {
         destroy progressDialog();
         add rATING_sTRINGS and oTHER_iNFO to results;
      }
   }
}

public FIRST_ACTIVITY // splash screen/wrapper activity
{
   /* essentially same as the second activity but with a splash
      screen with basic instructions and no need to use getExtra
      or Picasso */
}

public SECOND_ACTIVITY // restarts when user enters a new search
{
```

```
  String query;    //will contain user's search params

  onCreate()
  {
    Intent intent = getIntent();
    ArrayList<String> results = intent.getExtra();
    title = TextView(results(title));
    ratingsView = ListView(results(ratings));
    otherInfoView = ListView(results(otherInfo));
    image = new ImageView
    Picasso.get().load(imageURL).into(image); // fetch image

    if(the user has entered a new search)
    {
      ArrayList<String> results;
      reuslts = ItemSearch.search(query);

      create newIntent(SECOND_ACTIVITY);
      newIntent.putExtra(results);
      startActivity(newIntent);
    }
  }

  onCreateOptionsMenu()   //the searchbar
  {
    create searchManager as a SEARCH_SERVICE;
    prevent searchbar from being an icon;
  }
}
```

Wireframes/Screen Layouts

Figure 8: In this layout, we want users to have a view of what they need to do to search for the movie rating. The search bar is the only menu item and must be displayed at all stages. When the user clicks the search bar, the touch keyboard appears and the user types the name of the movie, which will be sent to Item Search.

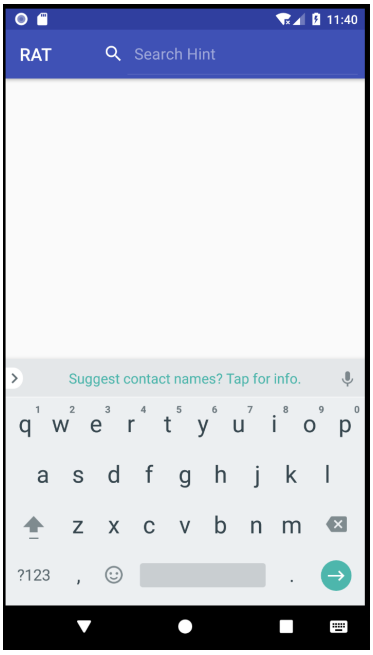


Figure 9: For the second layout, we want to show how the results will be displayed. The app will collect information about the movie from the list of sites by calling the AsyncSearch. It will give some basic information about the movie that the user was looking for (e.g. director, actors, plot summary), followed by a list of the simple ratings collected from websites in their native format.

Implementation

HCI Comments

Because our user interface was only displaying scores at the time of our meeting with the HCI team (in accordance with our expected milestone progress), and because we did not receive any feedback from them until May 6, it we were unable to benefit from their review. However, our user interface has come a long way since that meeting. Currently, we are able to display all the relevant movie data on a single page, with sections that scroll to reveal more information. These improvements have really made the product more user friendly and more useful in general.

While most of the issues that were mentioned in the HCI report were handled later in development, two things were brought up that we think were very valid suggestions. One was making the progress bar update. Currently, it remains at 0% until the search process has completed. This was something which we had planned to change and would have, given more time. The other was to include a search history which would store a list of previous search queries which could be selected to repeat that search. This would likely have been implemented as a menu item with a drop down, and could even have been incorporated into the splash screen.

Test Plan

A test plan is an essential part of any SPMP. We have detailed below the specific parts of our project which must be tested extensively before release, all relevant stipulations for testing, and any bugs we know of at this time.

Documentation checks:

- Black-box testing based on the specifications document (i.e. testing the input and output to see if it matches the *I/O Specifications*)
- Numerous trials should be performed to determine whether the app meets the acceptance criteria

Product checks:

- Searching a wide variety of movies, particularly ones which are less well-known
- Searching movies from a series
- Searching old movies with reboots either as shows or movies (e.g. Fargo)
- Spelling movie names incorrectly and/or incoherently

How and where to test the product:

- Ideally, so long as the user has an internet connection, the app should work fine. Therefore, testing should be done on a variety of different connections, including cellular data and Wi-Fi of varying strengths.
- The product is also expected to work on all android devices and should be tested on devices or emulators of varying size and hardware/software specifications.

Flaws we know about at this time:

- Duplicate Activities: Each time a search is completed, an extra copy of the results page is created on top of the original. This means the user would have to press back twice to see the previous search result. While not ideal, this issue is minor, as the user will still be able to access those reviews and will be likely to use the search bar as their primary navigation tool anyway. All attempts so far to fix this issue have broken the app.

- Scaling Issues: Scales fine on devices like the Pixel 2, that are 1080p and 420dpi. It starts to get cramped and overlaps elements with lower dpi's and resolutions. Larger phones with higher resolutions like 1440p, it scales to but there can be some whitespace at the bottom of the app.
- Progress Updates: Due primarily to time constraints, we were unable to make the progress bar show any signs of actual progress. As noted in the HCI report, "this gives the impression that the app has frozen," and should either be fixed or replaced by "a spinner to show that the app is not frozen."[1]
- Possible IP blocking: An emulator on one of our laptops only supports this app when on Juniata Wi-Fi or when using the JC VPN. Thus far, we have not been able to replicate this issue anywhere else.

[1] Vastine, Taylor. (2018). *Heuristic Evaluation for Mom's Spaghetti Code*. Unpublished Report, Juniata College. pg 1.