Github Link: https://github.com/OldsterJoey/bad_film_finder

Statement of Purpose and Scope

My terminal application is being designed to supply its audience with the most appropriate terrible movie, per answers provided within a simple three-step questionnaire. The questionnaire will ask three simple questions: if the user wants to watch a terrible movie, what genre the movie should be, and whether the user wants to laugh, cringe, embarrass themselves with friends, or die of boredom. The app will then output a movie following the answers provided, retrieving information from numerous film hashes, linked to a movie identification number that will allow access to a description and rating of whatever film is outputted. The selection of films will be chosen and coded in by me, as to ensure the user has a tailored experience.

For many people, finding the perfect movie to watch can be found using numerous online tools and websites. However, for the niche audience of people who instead want to seek out a movie they'll enjoy the least or one they can make their own fun from, there appears to be a gap in helpful tools for this purpose. My application is designed specifically to fill that gap, and allow users to find films that they'll hate, laugh at (not with) or visibly cringe to. As a bad film lover myself, I realise that bad movies can be bad for a large variety of reasons, whether it be that the film is mind-numbingly boring, or cringe-inducing. I want to create a tool that can separate these different bad movie types to find the most personalized bad film for an individual possible, utilising my knowledge of each film.

This application will be tailored towards the needs of an audience that enjoys and actively seeks out bad movies, as opposed to a general film-watching audience, and will be presented in a way that does not take itself too seriously. The app is meant to be used for fun, so it should also show within the design. The app will be extremely streamlined and simple to use, with built-in menus to make the app a very streamlined and user-friendly experience no matter what level of computer knowledge you have.

Features

The core feature of my app is its use of an external API that will give access to an external description and rating of whatever film is outputted to the user. This API will give access to a movie database, which will print relevant information based on an external identification number linked to each film retrieved from major film sites such as IMDb. This ID will allow access to a large variety of information for each film, such as aggregate scores and reviews, which can be implemented very easily if I choose to do so.

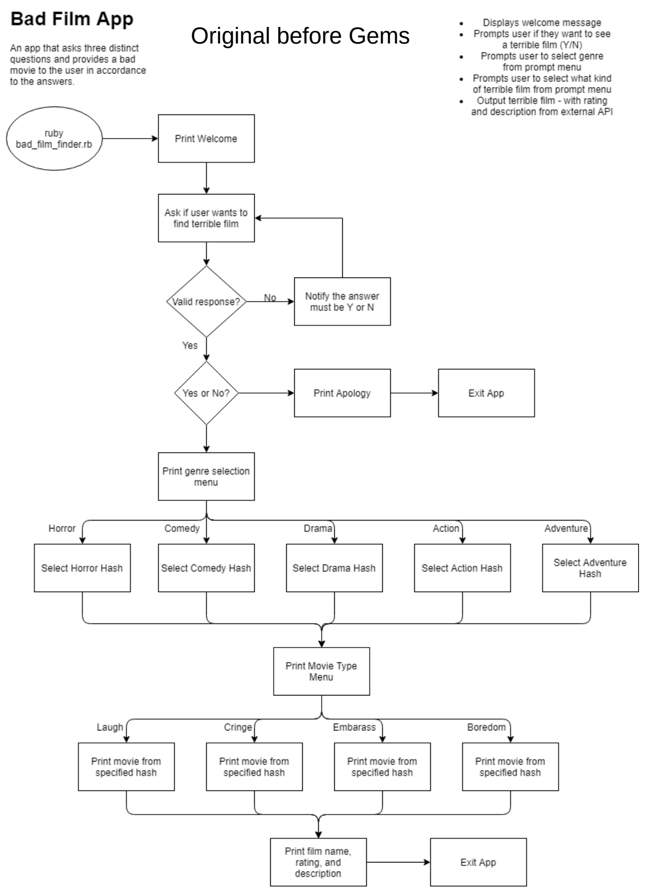
My app will also feature the use of an interactive selection prompt menu, created using TTY-Prompt, a part of a larger toolkit of many different components usable within a terminal application. This interactive menu will create a very appealing user experience, as it will effectively cut out any need for computer knowledge from the user. All selections within the app will be visibly seen as self-explanatory buttons the user can select, and each selection will provide user feedback so there is no possibility for feeling lost within the app. This feature also cuts out a lot of possibility for error within the code, as all selections are predefined and not typed by the user.

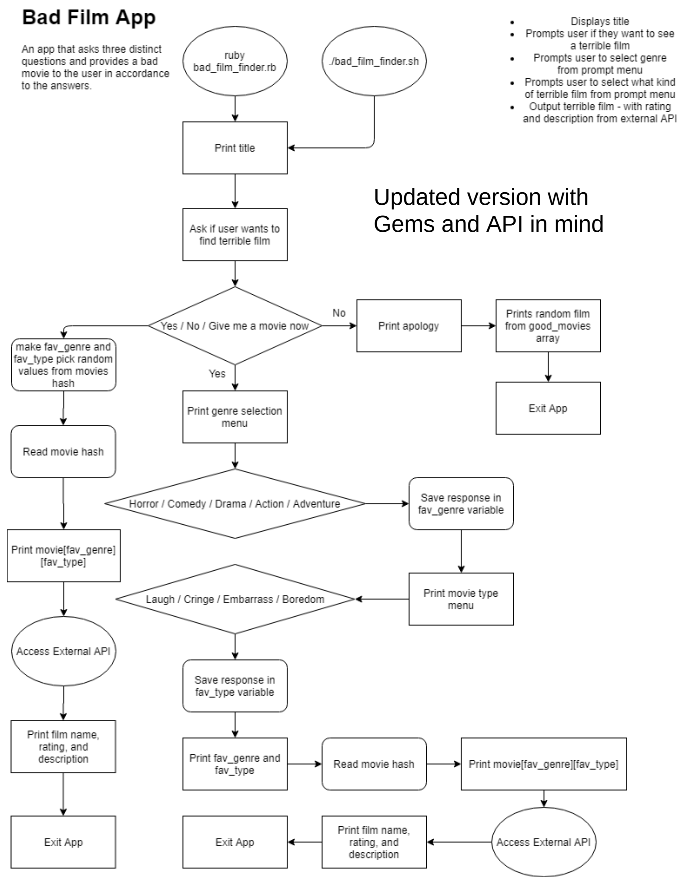
If there ever is a case where the application is unable to access data for the movie, whether it be due to being unable to access the internet, the movie id number being changed by the database, or error within the source code, the application will still be able to print the movies name, along with a user-friendly error informing the user the application is unable to access the movies information. The loop will then continue as normal. As all film names will be hardcoded in, the app can still function offline, just with fewer features.

User Interaction Outline

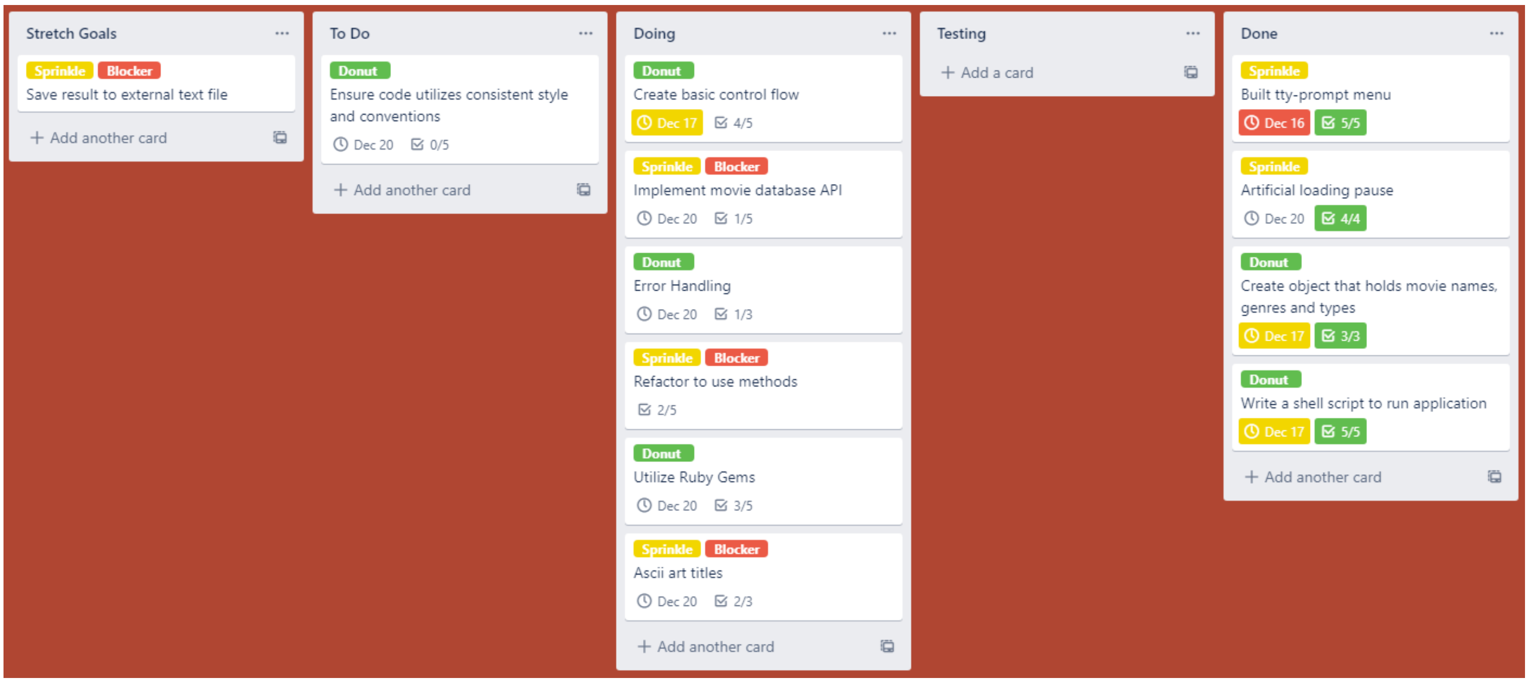
- Displays application title
    - Begins questionnaire loop
    - Prompts user if they want to watch a terrible film - Yes / No / Give me a movie now
    - - No - Display quit message, print out random award winning film
    - - Give me a movie now - Bypass questionnaire, save favourite genre and favourite movie type as randomised values and print a film
    - - Yes - Continue through loop
    - Prompts user to select genre from prompt menu - Horror / Comedy / Drama / Action / Adventure
    - - Save user selection as favourite genre variable
    - Prompts user to select what kind of terrible film from prompt menu - Laugh / Cringe / Embarrass / Boredom
    - - Save users selection as favourite movie type variable
    - Output terrible film - with rating and description from external API
    - Prompts user if they would like to start again Yes / No
    - - Yes - Loop back to start
    - - No - Print goodbye message and quit application

Control Flow Diagram


Bad Film App
An app that asks three distinct questions and provides a bad movie to the user in accordance to the answers.
Original before Gems
Displays welcome message
Prompts user if they want to see a terrible film (Y/N)
Prompts user to select genre from prompt menu
Prompts user to select what kind of terrible film from prompt menu
Output terrible film - with rating and description from external API
ruby bad_film_finder.rb
Print Welcome
Ask if user wants to find terrible film
Valid response?
No
Notify the answer must be Y or N
Yes
Yes or No?
Print Apology
Exit App
Print genre selection menu
Horror
Comedy
Drama
Action
Adventure
Select Horror Hash
Select Comedy Hash
Select Drama Hash
Select Action Hash
Select Adventure Hash
Print Movie Type Menu
Laugh
Cringe
Embarass
Boredom
Print movie from specified hash
Print movie from specified hash
Print movie from specified hash
Print movie from specified hash
Print film name, rating, and description
Exit App


Bad Film App
An app that asks three distinct questions and provides a bad movie to the user in accordance to the answers.
ruby bad_film_finder.rb
./bad_film_finder.sh
Displays title
Prompts user if they want to see a terrible film
Prompts user to select genre from prompt menu
Prompts user to select what kind of terrible film from prompt menu
Output terrible film - with rating and description from external API
Print title
Updated version with Gems and API in mind
Ask if user wants to find terrible film
Yes / No / Give me a movie now
No
Print apology
Prints random film from good_movies array
Exit App
make fav_genre and fav_type pick random values from movies hash
Yes
Print genre selection menu
Read movie hash
Horror / Comedy / Drama / Action / Adventure
Save response in fav_genre variable
Print movie[fav_genre][fav_type]
Laugh / Cringe / Embarrass / Boredom
Print movie type menu
Access External API
Save response in fav_type variable
Print film name, rating, and description
Print fav_genre and fav_type
Read movie hash
Print movie[fav_genre][fav_type]
Exit App
Exit App
Print film name, rating, and description
Access External API

Implementation Plan



Stretch Goals
- Sprinkle Blocker — Save result to external text file
- Add another card

To Do
- Donut — Ensure code utilizes consistent style and conventions — Dec 20 — 0/5
- Add another card

Doing
- Donut — Create basic control flow — Dec 17 — 4/5
- Sprinkle Blocker — Implement movie database API — Dec 20 — 1/5
- Donut — Error Handling — Dec 20 — 1/3
- Sprinkle Blocker — Refactor to use methods — 2/5
- Donut — Utilize Ruby Gems — Dec 20 — 3/5
- Sprinkle Blocker — Ascii art titles — Dec 20 — 2/3
- Add another card

Testing
- Add a card

Done
- Sprinkle — Built tty-prompt menu — Dec 16 — 5/5
- Sprinkle — Artificial loading pause — Dec 20 — 4/4
- Donut — Create object that holds movie names, genres and types — Dec 17 — 3/3
- Donut — Write a shell script to run application — Dec 17 — 5/5
- Add another card

Installation Guide

External requirements

A Movie Database API Key is required for application usage. In order to retrieve this key you must make an account at The Movie Database (https://www.themoviedb.org/), and follow the guide within this Documentation (https://developers.themoviedb.org/3/getting-started/introduction). API key must then be saved in a new file within app directory named 'credentials.json', in the format of {"api_key":"**YOUR API KEY GOES HERE**"}

Gem Dependancies

```
gem 'tty', '~> 0.7.0'

gem 'rspec', '~> 3.10'

gem 'rainbow', '~> 3.0'

gem 'lolize', '~> 0.0.3'

gem 'faraday', '~> 1.1'
```

Accessing required Gems

Use 'bundle install' command to install all necessary gems for app usage. All Gems can be found within Gemfile

Running the application

Bash script 'bad_film_finder.sh' included, permissions may need to be changed to execute. Alternatively, use 'bad_film_finder.rb' command to launch

Command Line Arguments

-help:

Refers to README

-info:

Prints Ruby version