Fresh Tomatoes: All You Search for in Movies

By: Team Fresh Tomatoes
Yvonne Chan, Kevin Zhang, Jonathan Buhler, Nala Sharadjaya

APIs being used:

Rotten Tomatoes (FOR RATING): TBD (sent request)
Moviedb (FOR SEARCHING + RANDOM MOVIE): https://developers.themoviedb.org/3/getting-started
New York Times Movie Review (FOR RATING): https://developer.nytimes.com/movie_reviews_v2.json#/Documentation/GET/critics/%7Bresource-type%7D.json

Task Delegation:

Yvonne - Project Manager
- Will debug and test everything
- Ensures everything works and is running smoothly
- Comments on other people's code to provide suggestions/ideas

Kevin - Front End
- Writes all HTML templates
- Help connect templates to app.py and backend files

Jonathan - Back End (+ database)
- Write authen.py to handle login/register
- Handle all writing to databases in dataAccess.py and other files

Nala - Back End
- Write interactAPI.py to handle retrieving information
- Access APIs and providing all information from them

Database Schema - database.db:

Users

TEXT user	TEXT pass
Stores usernames	Stores salted passwords

Profile

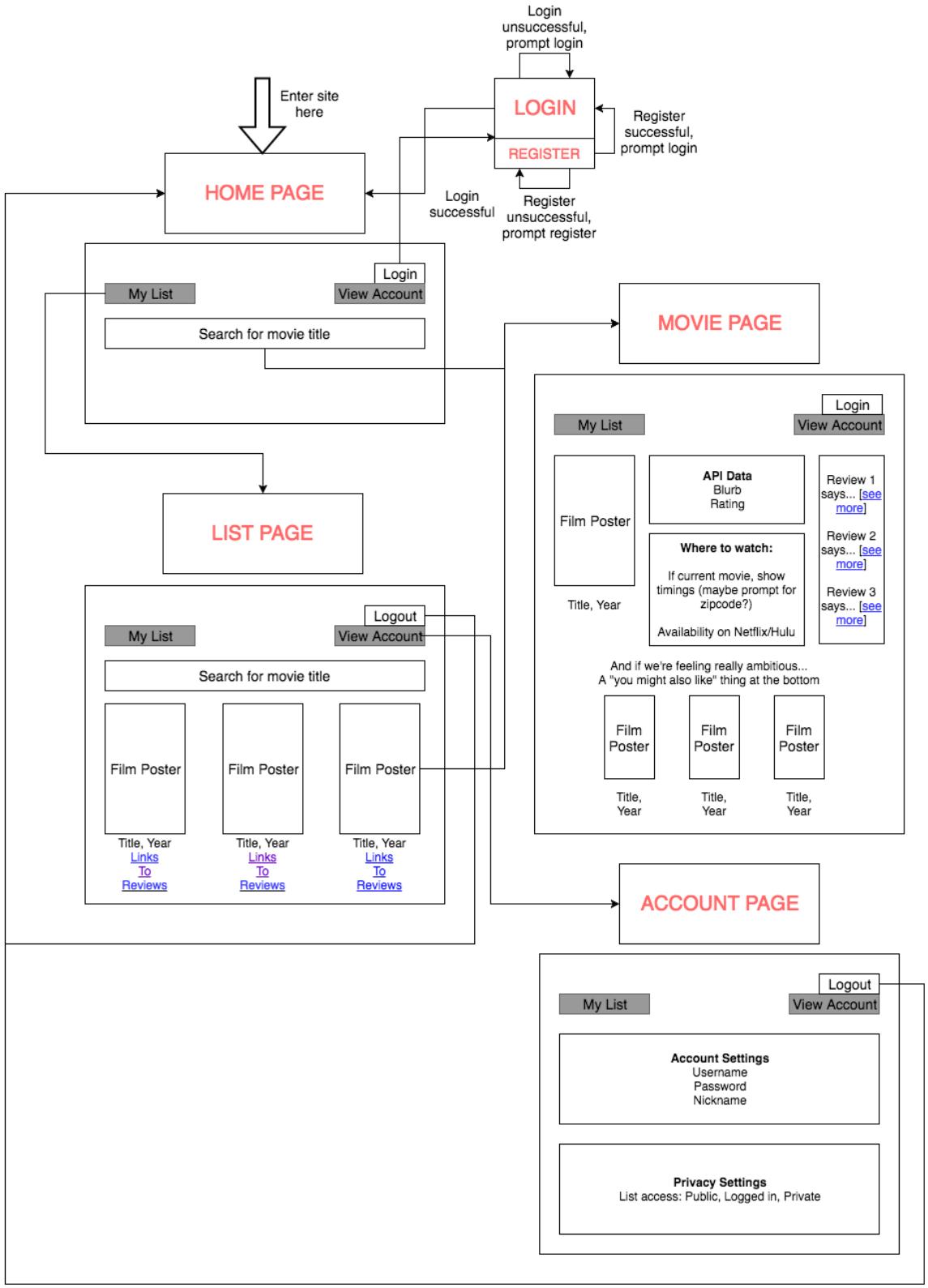
TEXT user	TEXT movie	TEXT genre	TEXT fav
Stores usernames	Stores saved movies; called from when user clicks on My List button	Stores user's favorite genre shown on profile	Stores user's favorite movie

Timeline

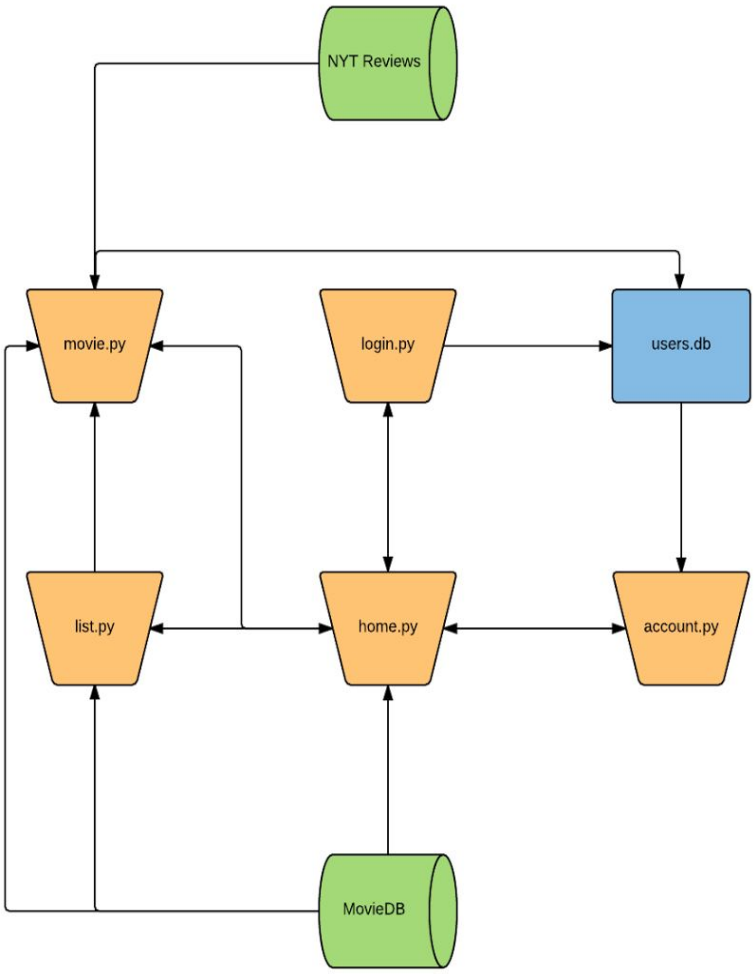
[Will update devlog everyday, subject to change depending on problems we encounter]

12/5 - design document done, create files and get started
12/6 - get login/register/logout working and having working templates (no bootstrap/css)
12/7 - get search working initially
12/8 - put together movie pages with ratings, summaries, etc
12/9 - allow users to save movies and edit their profile
12/12 - make site look pretty with bootstrap/css, add "I'm feeling bored" button for random movie, ensure everything's working

Site Map (Nala)



Flow Chart (Jonathan)



projectName: **freshTomatoes**

I. Directory Structure

~/Github/FreshTomatoes/

- app.py
- /utils/
  - authen.py
  - dataAccess.py
  - interactAPI.py
- /templates/
  - login.html
  - searchPage.html
  - results.html
  - movie.html
  - savedMovies.html
  - profile.html
  - editProfile.html
- /static/
  - /bootstrap/
  - jquery.min.js
  - style.css
- /data/
  - database.db

II. File Functionality

app.py

Main file that controls routing. Calls the functions of other files.

authen.py

Contains functions for login and register. It will access *database.db* for authentication information and check if information matches. In addition, it will check username for redundancy. Also edits *database.db*.

dataAccess.py

Control of *database.db*. Contains functions for modifying the database, getting user information from database, and etc. Serves as the main bridge between files and the database. User information, particularly for *profile.html* will be accessed through this file.

interactAPI.py

Interacts with APIs for movie information. Any new movies, previously unknown, will be added to the database, with its specified information. This is to allow users to save movies and their data. Will work alongside *dataAccess.py* to write to database and etc.

login.html

Web page that contains form for logging in and registering. It will contain two fields, one for username and one for password. Two buttons will also be included, one for register and one for submit.

searchPage.html

Essentially the homepage. Looks a lot like the Google homepage, with a search bar in the middle and a profile icon along the top. Also includes a random "I'm feeling bored" button, which will redirect the user to a random movie.

results.html

Returns the results of the search, particularly useful for multiple editions of the same movie. This page will give a list of movies, organized alphabetically by title. A brief description is included as well, but for more information, you have to click on the movie link itself.

movie.html

Displays the information regarding a specific movie, such as release date, ratings, links to reviews, summaries, etc. Includes any data that we could get using our APIs, although if the information is missing, then we will fill out whatever we can. If the user has mistyped, then it will simply display a 'movie not found' message.

savedMovies.html

List of saved movies that the user has recorded. This looks very similar to *results.html*. Each movie has a separate link to its own html page.

profile.html

Allows the user to see their profile. This will generally be filled by some personal and some movie-related information (such as favorite genres). The user is able to edit any of this information at any time.

editProfile.html

Html page for editing the profile information.

database.db

Database file. Includes username and salted passwords. Also includes users and their saved movies.