Team 10 - Alex Rosenfeld, Cole Secor, Reda Ijaz and Tudor Dorobantu
CS 411 A2 Professor Donham Perry
Couch Potato Application
5/5/2017

Final Documentation

1. Initial Pitch

The ever-increasing number of online movie streaming services has raised the time and cost for searching such entertainment. Each of these services have different subscription prices and different entertainment offerings. Our app seeks to give transparency and save time for users interested in watching movies online through streaming services such as Amazon Prime Video, Netflix, HBO Now, Hulu and many others. We will begin by gathering data from TMDB, and determining whether the selected movie is available on Netflix vs Amazon. As we move forward, we will add other sources like HBO, etc., to the application.

2. Pitch Changes

The team struggled with receiving access to API's such as Amazon's Product Information Services as well as Fandango's Movie Search API. Amazon's Product Information API required extensive documentation as well as an active website in order to receive an API Key. The Fandango API was also inaccessible although all documentation was provided. Additionally, some online streaming services such as Hulu and HBO Now did not offer any API's. Our luck seemed to turn when we uncovered Nielsen's Gracenote movie API (http://developer.tmsapi.com). The API would had solved all the issue regarding all of our API's since Gracenote API calls return movie metadata such as links to all of the online streaming services that offer a specific movie. We applied for an API Key and received it. However, as of the writing of this paper the API is still inaccessible as our API Key remains labeled as "Developer Inactive". Admittedly, these issue impact the original functionality of our application since users are not longer able to quickly access all the online services. However, the application still offers users the ability to access online services such as Netflix and iTunes. Overall, although the project did not turn as we would have expected we still learned a great amount of skills such as the Django framework, implementation of 3rd party authentication and API web application implementation.

3. Database

We will store user login information and search history in the database.

4. Public Data API

We applied for more API access however only the following did reach back to us:

Itunes: https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/
Netflix: https://www.programmableweb.com/api/netflix
TMBD: https://www.themoviedb.org/documentation/api/

5. States

1. Login Page
2. Sign-up Page
3. Login/Signup Confirmation Page
4. Movie Search Page w/ history
5. Movie Search Result page

6. Changes to States

Originally we intended to have an account settings/preferences page where users would specify the online services that they subscribed to. We decided to remove this page since it only cluttered our design. Instead, the user has access to all the online streaming service that contain the movie and are supported by our platform. If the movie is not an particular platform than the user will not see a button appear. This design choice goes back to our applications ethos of being a easy-to-use movie picker application.

7. Third Party Authentication

Our application will allow users to login with Facebook. Documentation to Facebook 3rd party authentication using django is provided below.

Facebook: https://github.com/batcole/T10/tree/login/fbauth

8. User Stories

User stories changed from previous version in order to reflect applications new focus on ease-of-use as well as issues encountered with APIs.

User Story 1 - As a current user, I want to log-in with either Facebook or Couch Potato Credentials

The user will presented with the login screen and can either choose to authenticate via Facebook or Login with their Couch Potato Credentials (username and password). They click on the button for which authentication method they prefer to use and advance to a confirmation page. From here may chose to either logout or search for a movie. The user will not log-in if either the password or username is wrong. If this is the case, a message will be displayed to user on the login page in order to let him know that he provided the wrong password and/or user.

User Story 2 - As a new user, I want to create a new account with Couch-Potato

The user chooses to sign-up for the Couch Potato service by clicking on the Sign-up button on the Landing page. The user is taken to the sign-up page where has to provide a username and his desired password twice.

The sign-up form will not be submitted and appropriate error messages will be displayed to user on the sign-up page if:

- the two provided passwords are not the same
- the password is either too short or similar to the username
- The username is already taken (exists in the database)
- The password does not meet the alphanumerical requirements

If the sign-up form is valid the user will be taken to a confirmation page. From here may chose to either logout or search for a movie by clicking on the appropriate buttons.

User Story 3 - As a logged-in user, I want to search for a new movie and see the movie's description as well as access the website of the online services that offer it.

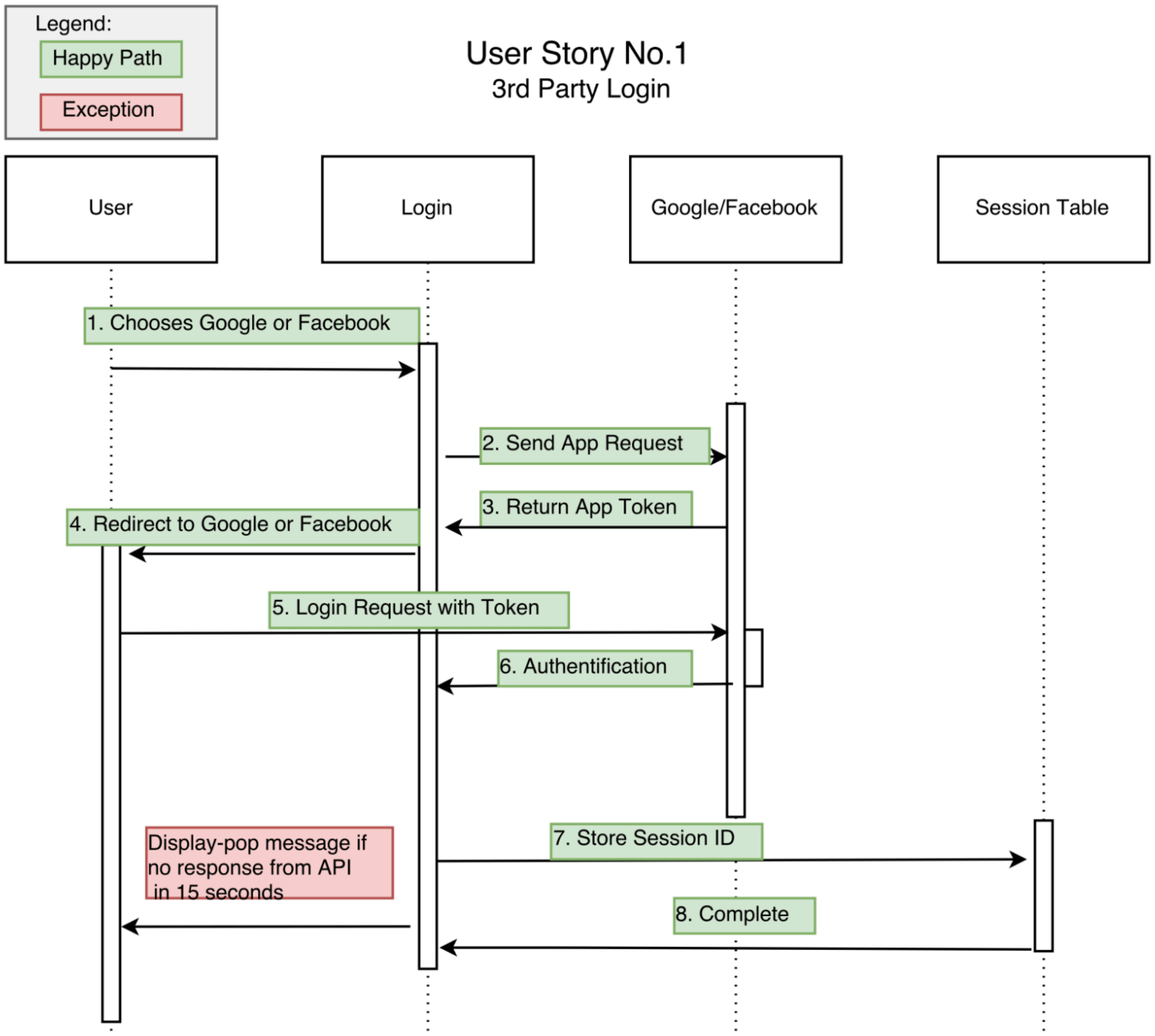
See *User Story 1* in order to see log-in process. From the login confirmation page, the user choses to search for a movie by clicking on the *search for movies!* button. If user chooses to search for a movie he will be taken to the search page. The search page will display a list of movie names and links that are cached on the user's machine. The user may access the detail pages of the movies in the list or he may search for a new movie. The user will be shown a list of movies similar to the search keyword(s) after a search. The user may press on the link of the movie to access the details page were movie details and links to online streaming services are shown. The user may press the buttons with a certain online streaming service to access its offering of the movie.

9. Framework Choice

We will be using Python/Django since it has open source documentation and tons of packages to work from. Django provides well organized documentation and example code to help guide us through challenges. Django also comes with a fully-featured web interface that gets automatically generated, this will allow non-developers to also have control over CRUD Activities.

We decided not to go with Node because our team does not have experience with web development and structures and Django enforcing good development principles. While learning a framework is important, but it more important to learn good development principles.

10. Use Case Diagram



11. Use Case Path

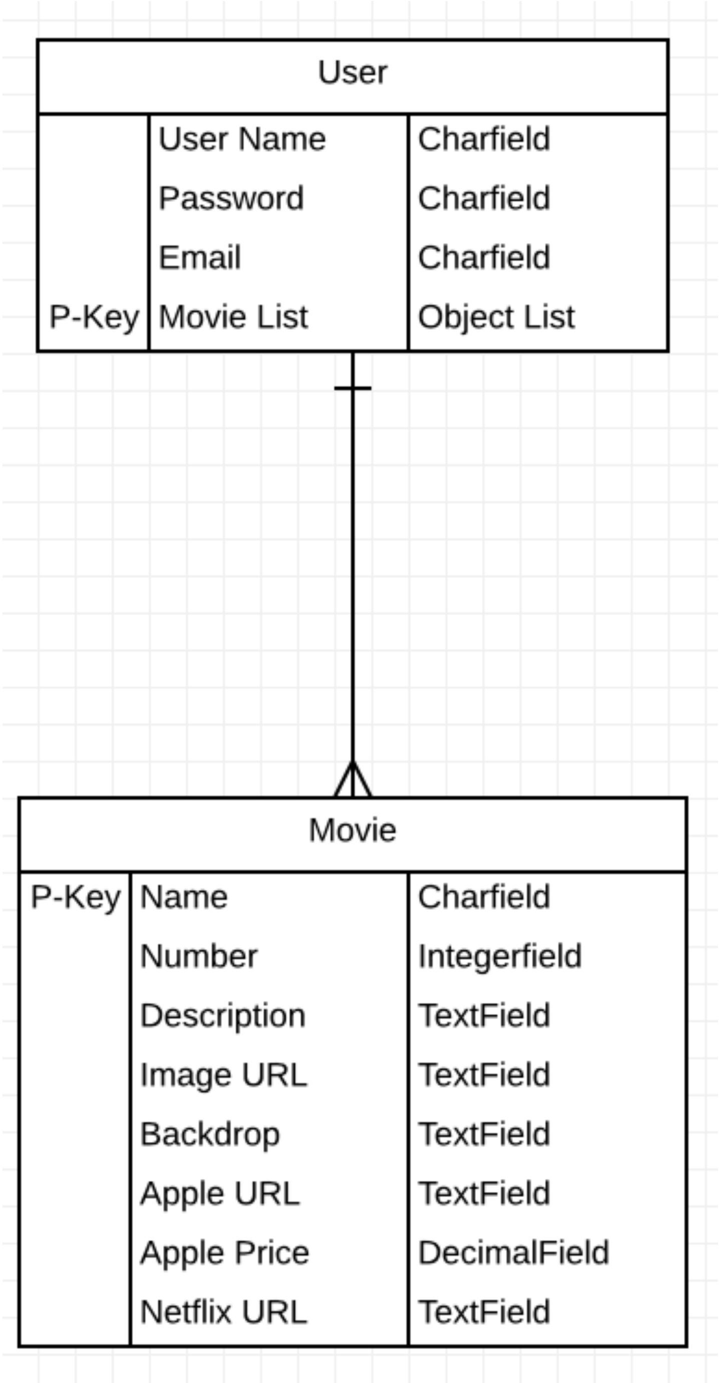
Happy Path
Exceptions

1. User chooses Google or Facebook for authentication
2. Send app request to either Google or Facebook
    a. App request isn't returned in 5 seconds
        i. User is returned to step one
3. App token is returned
4. Redirect user to Google or Facebook
    a. Error message shown to user to check login and/or Internet connectivity

5. Login request with token taken to Google or Facebook
6. Authentication
    a. User asked to login again if no response
7. Session ID is stored
8. Login complete

12. Database Schema Diagram

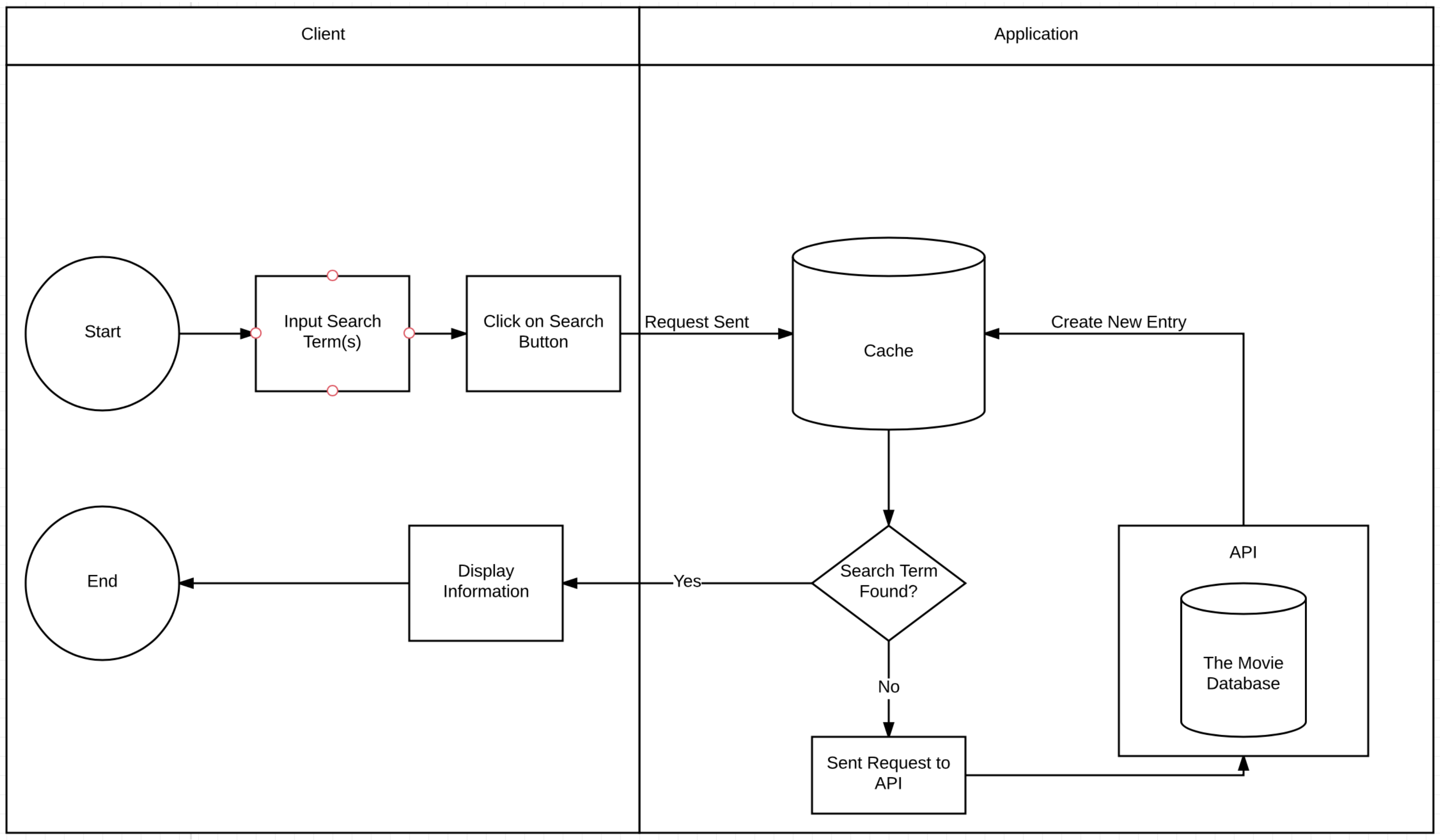
Database Schema changed from initial built. New Schema new fields such as Apple URL, Apple Price and Netflix URL. Other changes include removal of movie list table. This change improves application efficiency.



User		
	User Name	Charfield
	Password	Charfield
	Email	Charfield
P-Key	Movie List	Object List

Movie		
P-Key	Name	Charfield
	Number	Integerfield
	Description	TextField
	Image URL	TextField
	Backdrop	TextField
	Apple URL	TextField
	Apple Price	DecimalField
	Netflix URL	TextField

13. API Call Sequence Diagram


Client
Application
Start
Input Search Term(s)
Click on Search Button
Request Sent
Cache
Create New Entry
End
Display Information
Yes
Search Term Found?
No
API
The Movie Database
Sent Request to API