Assignment 1: Webservices

Benjamin Vandersmissen & prof. José Oramas

DS 2022-2023

1 Introduction

This document provides an overview of the requirements and background information for the first practical assignment of the Distributed Systems course. The goal of this assignment is to familiarize yourself with accessing public-facing REST APIs, implementing your own webservice via a REST API and consuming the webservice in a (small) webpage. This assignment is **individual**.

2 Goal

The goal of the assignment is twofold:

- Firstly, using the API of `https://themoviedb.org`, your goal is to create a RESTful API that functions as a very basic recommendation engine for movies.
- Secondly, you should implement a simple webpage that consumes your API. This will mainly serve to easily illustrate your functionalities and as such, wont be graded on the design.

3 Requirements

3.1 Webservice

The webservice needs to implement at least the following requirements:

- List the first x popular movies. (x can be any arbitrary number)

- Given a movie, return the movies that have **exactly** the same genres.
- Given a movie, return the movies that have a similar runtime. (You can assume a similar runtime has a maximum of 10 minutes difference)
- Given a movie, return the movies that have two overlapping actor(s). (You can assume the first 2 actors listed)
- Given a set of movies, have the ability to generate a barplot comparing the average score of these movies.
- Be able to 'delete' a movie, i.e., the movie won't be returned from the API after 'deletion', unless the API is restarted.
- Be able to 'like' and 'un-like' a movie.

In addition, your API should also follow the principles of RESTfulness. This means that you may need to implement additional endpoints not listed above, depending on your URI scheme implementation.

To implement these requirements, you will need to use two APIs. The first API is TMDB API (`https://developers.themoviedb.org/3`) and will be the main API that will provide you with all necessary data.

The second API is the QuickChart API (`https://quickchart.io/documentation/`) which you will be using to create plots.

3.2 Webpage

The webpage will mainly serve to illustrate the functionality of your API. As such, the only requirement here is that via the interface you can easily demonstrate all requirements during the demo session. As such, **this will not be graded on appearance**!

4 Tools

The following tools and links might be useful in your implementation.

- REST API in python: Implementing a REST API in python is actually pretty easy using a couple of libraries. The main libraries are Flask (`https://flask.palletsprojects.com/en/2.0.x/quickstart/`) and Flask-RESTful (`https://flask-restful.readthedocs.io/en/latest/`). You should already have some familiarity with Flask from Programming Project Databases.

- Web technologies: If you want to use any extensions of vanilla javascript or CSS, you might take a look to JQuery (`https://jquery.com/`) which provides advanced javascript functionality in an easy fashion and Bootstrap (`https://getbootstrap.com/docs/5.0/getting-started/introduction/`) which is simple and beautiful CSS. Of course, you are free to choose any javascript and CSS libraries you want.
- Documentation for the TMDB API can be found at `https://developers.themoviedb.org/3`. When designing your own API and website you should take care that you follow the best practices and limitations defined in their documentation. If not, your application might suddenly stop working.

5 Deliverables

The following are the minimum requirements and deliverables for a passing grade.

- You should include a way for us to easily run your solution with the inclusion of a `run.sh` that will automatically start your web service and user interface.
- You need to include a manual with your code that details the API endpoints you implemented along with the arguments they require, their return values and a small description.(for inspiration you can look at the documentation of the given APIs, or you can use specialized packages such as flask-restful-swagger)
- The manual also needs to include a section on the design considerations of your API. Be sure to follow the principles of RESTfulness and only use web communications!
- Make sure that your submitted solution is complete and has no missing files. If you use additional libraries, be sure to mention this clearly in the manual and either include them with the source, or provide us with an easy way to install them. (for python this is with a `requirements.txt`). This also means that you need to provide the API keys you use as well, and need to make sure there is enough quota for us to test your solution.

6 Deadline and Submission

The deadline for this assignment is the **18th of April** at **23:59**. Late submissions will automatically have points deducted for tardiness. In addition to the submission, you will also be required to give a small demo of around 5 minutes. Details on the demo session will follow later.

Make sure that your submission has the following format and upload it via Blackboard!

`DS-Assignment1-`*`Snumber-Lastname`*`.zip`.