Texts for the first Use-Case

Actor Relationship Game

The Actor Relationship Game should be a program which allows users to explore relationships between actors. These relationships are established by the films in which actors took part in. The data for the informations about actors and the respective films they took part in comes from the The Movie Database (TMDB) API. Interactions with TMDB API must adhere to its terms and usage limits. With the given data the Actor Relationship Game has to be able to identify the shortest path between actors. Thus this Java-based programm has to be able to represent the shortest path visually. To realise the aforementioned functionalities, the Actor Relationship Game has to handle API calls to fetch actor and movie data. The api-key, which has to be used to retrieve curcial movie and actor data, has to be stored in a an environment variable called „TMDB_API_KEY". With the api key and the TMDB data, this program should be able to search for specific actors and films by their respective IDs. Aside from the programming language Java, the programm has to use Gradle, a TMDB API key, java.util, java.io, java.nio, com.google.gson, org.junit.jupiter.api and since this is a Java-based programm the Java Runtime Environment has to be utilized as well. The objects this program has to work with are actors and movies. Additionally there may be utility classes for a gameplay interface or the creation of the shortest path between two actors. An actor has to have an unique ID, a name and a set of movie IDs they participated in. A movie has to have an unique ID, a name, and a set of unique actor IDs involved in the movie. To save the shortest graphs that could be found, the results should be saved inside a .txt file called „actor_connection_results.txt".

Hone

The Hone programs main purpose is to convert CSV files into nested JSON files. In order to do so, this program has to be written in Python and utilize built-in libraries to parse CSV files and generate respective nested JSON files like csv, json, argparse or contextlib. Therefore Hone has to be able to parse CSV files and generate nested JSON files. This program should also be able to deal with different delimiters which might be used for CSV files. The generated nested JSON files need to have proper indentations and sorted keys for better readability. For the implementation different utility classes may be used. The program should also be usable via CLI. Furthermore one should be able to set variables like custom delimiters for the delimiters of the CSV file which the user refers to or a schema variable which allows the user to set a template for the output of the nested JSON file. The aforementioned variables have to be optional. In addition to that there have to be two required variables with the CSV- and the JSON filepath. Apart from the mentioned libraries no external dependencies are required for the core functionality.

Hybrid-Images

The Hybrid-Images program has the objective to generate hybrid images with the usage of different image processing and filtering techniques. These techniques and filters include cross-correlation, convolution, Gaussian blur, high-pass- and low-pass-filters. With these techniques the program should be able to generate a hybrid image using two different images as an input. The merged image has to be visuallsy coherent to the two input-images. This program needs to be implemented with

Python. Concerning external libraries NumPy and OpenCV can be used for the image processing tasks.

Login-Registration

The Login-Registration program serves as an user management interface which features three pages. The first one is for an user authentication which has two input fields for the username and password. The latter input should have the input type password. Additionally this program should provide two buttons for the user authentication page. One should be for the login and the other should allow users to register. If the user klicks on the register button there should be a page which offers four input fields asking the user to type in his first name, last name, username and password. Moreover the password should be constrained to be at least six characters long. Of course the user has to fill all the inputs in order to register. A cancel button should allow the user to return to the previosly described login page. The third and last page this program has to have appears after the user has logged in successfully. On this page the user should be greated with his first name that was set during the registration process. Under the greating message there should be a list with all the users who registrated for the page. The list features both the first- and last -name of the respective users and a delete button which allows the user to delete user-accounts. The user interface has to be intuitive, user-friendly and flexible. In addition to that the user interface has to be both compatible with all common browsers like Chrome, Firefox, Safari and Edge as well as mobile devices such as smartphones or tablets. For the implementation process this program should be implemented with JavaScript and use Vue2.x as the frontend framework, Babel6.x for the compilation process, NPM for third-party libraries and CSS for styling if necessary.

Readtime

Readtime is a program which estimates the reading time of different types of content. The estimated readtime should be based on the words-per-minute rate. This program has to be able to process plain text, HTML and markdown formats. In order to give realistic estimates this program should take both complexity and length of the input texts into account. If the user tries to use an unsupported input such as TeX-Code or AsciiDoc there has to be a proper exception handling with error-messages for the user. As for the technologies to use, this program has to be implemented with Python 3.x and should use libraries like beautifulsoup4, lxml, markdown2, pyquery and pytest.