UNIVERSITÉ Concordia UNIVERSITY

ACADEMIC YEAR: 2020-2021

ASSIGNMENT 1

Assignment Posted: October 20, 2020

Assignment Due: November 09, 2020 before 11.59pm

Final Deadline with 5% Penalty per day: November 14, 2020 before 11.59pm

This is the first of two programming assignments in our course. Note that these assignments are not 'stand-alone', but rather build on top of each other to form a bigger project.

The assignments focus on applying the concepts covered in the lectures, such as functional programming, lambdas, streams, asynchronous programming with futures, and actor-based programming, in the context of a web application.

Both the assignments must be developed based on the Play Framework, a full-stack reactive framework for JVM, which is also used in large-scale commercial deployments by companies such as LinkedIn, Morgan Stanley and Walmart.

Note that this assignment has both a group part and an individual part. Your group must submit a single application. Your own marks will be based on the group work (50%) and your individual contribution (50%).

Play Application: "YouTubeAnalyzer" (Group Part). The overall goal of this project is to develop a web application that can analyze the live feed of the YouTube Data API.

The group part contains the setup of the overall framework for this web app, as well as a simple web interface that takes a string with a single or multiple search keywords (input text field at top of the page, see Figure 1) and then displays data pertaining to the latest ten videos matching the keyword(s), below the search field. Each match must include the video title, owner/channel, number of views, time since the video was posted.

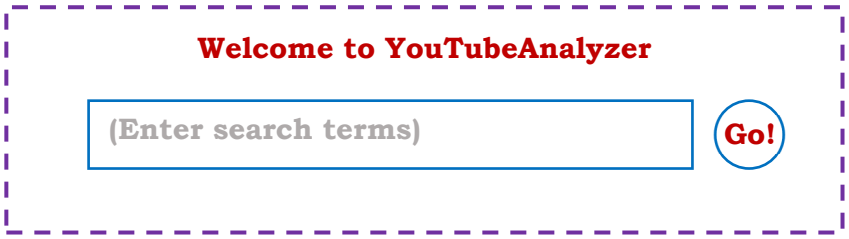

Figure 1: YouTubeAnalyzer page when first started

The user must be able to enter new search terms on the output page, which will result in data for ten more videos being displayed (*i.e.*, a second search will add 10 more results below the initial ten results from the first search and so on – see Figure 2).

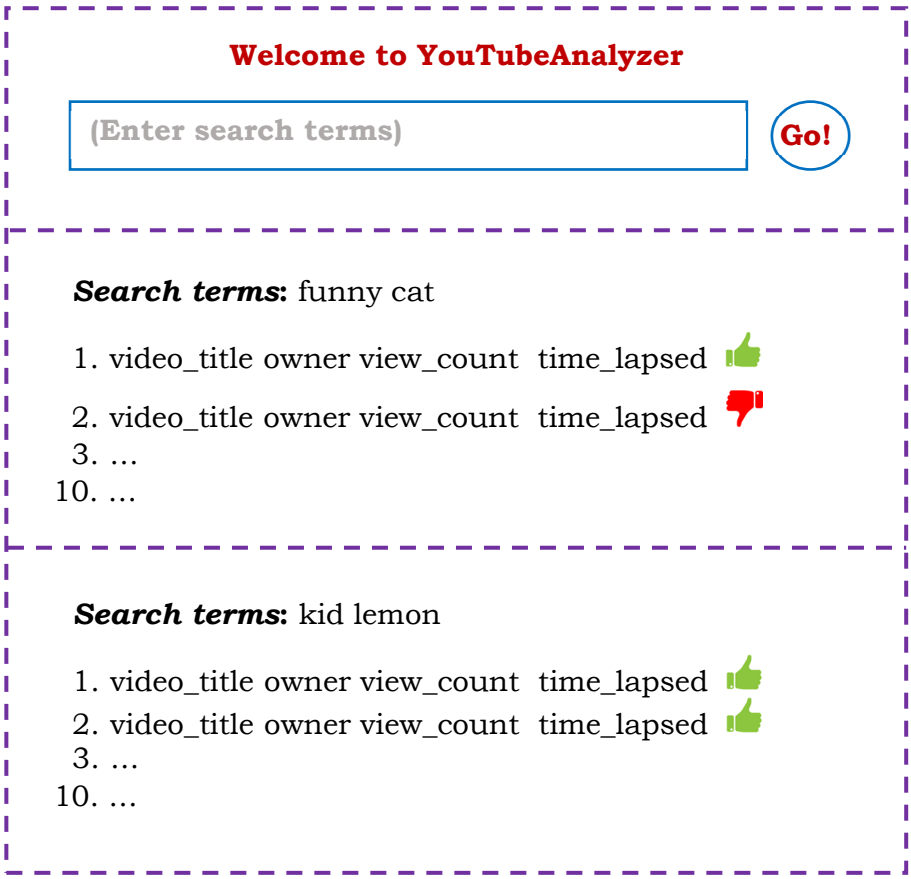

Figure 2: YouTubeAnalyzer page after two searches

You must use the (free) YouTube Data API to access the information from YouTube, Please refer to https://developers.google.com/youtube/v3

Process the results using the Java 8 Streams API to add hyperlinks to video titles, owners/ channels, view count and time lapsed, (used in the individual parts detailed below).

Individual Part. In addition to the group part above, each team member must work on one of the following features. Your team must decide who will work on which part – two members cannot work on the same task!

a) Owner/Channel Profile: Create a web page containing all the available profile info about an owner/channel (Please refer to YouTube owner/channel page). This profile page must be hyperlinked with the owner/channel name from the results on the main search page.

b) Similar Content: For a search query (one or multiple keywords), display a similarity-level statistic for hundred latest videos from search results, counting all unique words in the video title in descending order. This statistics page must be hyperlinked from the search terms above the results. You must use the Java 8 Streams API to process the data.

c) Owner/Channel Videos: Display the info for ten latest videos posted by the same owner/ channel as the one from the search results (where available, linked through owner field). The videos must be sorted by upload date followed by the search terms.

d) Video Sentiment: Given a stream of comments from (up to 100) videos, determine if these comments are overall positive 👍, negative 👎 or neutral 👌. This sentiment will have to be displayed on the main search page for each search result (see Figure 2). Process the stream of comments, finding happy and sad emoticons and counting them. If the stream of comments contains more than 70% happy emoticons, return a positive sentiment 👍, for more than 70% sad, return an unhappy emoticon 👎, otherwise a neutral one 👌. You must use the Java 8 Streams API to process the comments.

Coding guidelines. Your submission must satisfy the following requirements (these apply to both the group work and the individual work):

Play Framework: Note the following general guidelines for using the Play Framework and building the application:

1. Your application must be based on the Play framework.
2. It must be possible to build and run it using the standard sbt run command.
3. Any required third-party libraries must be automatically resolved through sbt.
4. Standard build targets (sbt run/test/jacoco/javadoc/clean) must all work correctly.
5. You can optionally use a library, but you have to wrap any synchronous APIs to make them asynchronous (see below).
6. Do not put business logic into the controller. Use dedicated model classes (MVC pattern) for your application functions.
7. Write a single controller for your app, integrating the individual parts defined below.
8. Make sure your server correctly handles user sessions for multiple searches – *i.e.*, if you open your app from a second browser, you must not see the search results from an ongoing session!
9. Do not fetch the same info multiple times (*e.g.*, for passing them between different tasks) – develop an appropriate caching strategy for your application.

   You can setup a DevOps server for your team (e.g., using Jenkins), but this is not required for the project.

Documentation: The following guidelines concern your documentation:

1. Document all your classes and methods with Javadoc.
2. This includes private methods and unit tests!
3. Make sure all classes and methods include an @author tag (both for group and individual work).
4. Include the generated HTML Javadoc in your submission (make sure you generate it for the private methods as well).

Reactive Programming: The following guidelines refer to using reactive programming techniques. These will become especially important for the second assignment (second part of the project), so you should do your best to get it right the first time.

1. Your controller actions must be asynchronous (non-blocking), using Java 8's CompletionStage<T> /CompletableFuture<T>.
2. Do not use .get()/.join(), thread.sleep or similar to block for a future's result anywhere in your code (only exception are unit tests, see below)!

Note: any blocking code in your app will lead to significant marks reduction!

Testing: Testing your code is a very important part of the project. Make sure you write your unit tests at the same time as you implement your features. Code that is not tested will count as not implemented! – In other words, if you completely implement the whole project, but do not have a single test, your submission will receive 0 marks!

1. Create JUnit tests for all your classes. You must have tests for every method in your controller and every controller action, as well as any additional classes you wrote. Compute your test coverage with JaCoCo. You must have 100% test coverage of your classes, methods, and lines (excluding classes automatically generated by Play).
2. Within a test, you can use .get()/.join() to wait for a result.
3. Never call the live API from a unit test, use a mock class for testing instead. For testing
4. your classes using the API, you must use either your own, factory-generated mock class (you can use Dependency Injection (DI) with the Google Guice library) or use the Mockito framework.
5. Your unit tests must be properly designed by finding the equivalence classes and doing partition testing. A unit test that does not properly check the result values (e.g., only checking if a result is non-empty, rather than comparing the values), will count as not written for the purpose of test coverage!

Submission. There are two deadlines, first for submitting the information which developer will handle the individual part and the second for the system. Note that for each submission, only one upload per group is required.

Submission 1: Group Information. You must submit the following team information on Moodle by the first due date (Sunday, October 25, 2020):

Github: The URL of your Github (or Gitlab) repository containing the project. Create project repository and grant access to the TA (Github: malhotraarpit, GitLab: @malhotraarpit)

Team setup: List all the team members, indicating which of the individual parts above are assigned to each team member. An individual task can only be assigned to one team member and the marks (50% of the total marks) for that task will go to the designated team member only.

Submission 2: System. You must submit your code electronically on Moodle before the due date (late submission will incur a penalty, 5% per day, up to a maximum of 5 days). Include a signed (by all team members) Expectation of originality form with your submission.

Demo. You must also demo your code to your TA in one of the lab sessions (time slots for the demo will be reserved through Moodle). Note that all group members must be present for the demo and ready to answer questions about the code.