SE480 – Software Architecture I
Autumn 2020
Homework 3
Due Date: Thursday, October 22, 2019 5:30PM
100 points possible. Will be scaled to 10% of final grade.

Submit your answers to D2L by Thursday, October 22, 2019 5:30PM. Late homework will not be accepted.

Assignment Overview

This assignment will give you practical experience applying a few of the architectural tactics we've discussed in class to help build a deeper understanding of how they work.

In this assignment, you will:

1. Implement a simple, microservices-style movie recommendation site
2. Use a build automation tool to manage builds and automatically run tests
3. Use dependency injection, mocking, and a unit test framework to thoroughly unit test your code in an automated way
4. Use a third-party fault tolerance library to gracefully handle component failures

All code must be implemented in Java and use either Maven, Gradle, or SBT as the build system.

Assignment Details

You are a developer creating an movie recommendation site. This site consists of many services, but we will focus on the following two:

1. A user service, which is used to retrieve information about the currently authenticated user
2. A movie recommendation service, which recommends movies based on information about the currently authenticated user

The movie recommendation service uses the user service to determine information about the currently authenticated user in order to make recommendation decisions.

Do the following:

1. Create a Java class to represent the user service. Ensure it has the following method:
   - `public int getAge();`

   In a real program this service would have many more methods and be connected to a user database of some kind, but for this assignment, you can have the user service return a hard-coded value.

2. Create a Java class to represent the movie recommendation service. Ensure that it has the following method:
   - `public List<String> getRecommendedMovies();`

   Have `getRecommendedMovies()` call the user service and return the following values based on the user's age:

Age	Recommended Movies
< 13	Shrek, Coco, The Incredibles
≥ 13 and < 17	The Avengers, The Dark Knight, Inception
≥ 17	The Godfather, Deadpool, Saving Private Ryan

3. Using a unit test framework such as JUnit and a mocking framework such as Mockito, write a set of unit tests that mocks the user service and tests all of the above cases. This may require you to refactor your above code to use dependency injection to inject the user service into the movie recommendation service if you did not do so originally. Connect these unit tests into a build automation tool (Maven, Gradle, or SBT) so that these tests are automatically run on every build.

4. Using Netflix Hystrix, modify the movie recommendation service such that if the user service is unavailable, the recommendation service will return the movies which it recommends for people under 13 years of age. Introduce new automated test cases which simulate these failures (by using mocks which have `getAge()` throw an exception) and verify that the recommendation system behaves correctly in this case.

5. Using Netflix Hystrix, modify the movie recommendation service such that if the user service does not respond within 100 milliseconds, the recommendation service will return the movies which it recommends for people under 13 years of age. Introduce new automated test cases which simulate this case (e.g. by using mocks which have `getAge()` sleep for a long period of time) and verify that the recommendation system behaves correctly in this case.

Turn in all the above code in a single `.zip` or `.tar.gz` file. I must be able to uncompress the package, run the build automation tool (e.g. `mvn package`), and see all automated unit tests pass.

References

The following references may be useful for this assignment:

1. Maven (build automation tool): https://maven.apache.org/
2. Gradle (build automation tool): https://gradle.org/
3. SBT (build automation tool): https://www.scala-sbt.org/
4. JUnit (unit testing framework): https://junit.org/junit5/
5. Mockito (mocking framework): https://site.mockito.org/
6. Netflix Hystrix (fault tolerance library): https://github.com/Netflix/Hystrix