Cinequotes

The product management team at Cinequotes has discovered that there exists an underserved market demand for finding quotes attributed to films and actors. After a successful experiment with a focus group of film enthusiasts, a delivery team has formed with a mission to develop an early minimum viable product to assess user experience and gather feedback. You are a key member of an autonomous delivery team, consisting of a Product Manager, Architect, Frontend Developer, Site Reliability Engineer and you! It is your responsibility to work with the team and build the backend to support the initial release.

MVP Requirements

Your team gets together for a few hours in the morning working with the Product Manager to develop a shared understanding of the product to be built. By the morning's end, the team has agreed upon the requirements for the first release. In order to make sure that you meet the delivery target of one week, your team has also agreed to build only what is necessary to meet the requirements of the MVP and *nothing* more.

Scenario: Viewing available films

Given that a user navigates to the Cinequotes home page
When they look at the page
Then they should see a list of all available Films

Scenario: Viewing available quotes

Given that a user navigates to the Cinequotes home page
When they click on a specific Film
Then they should see a list of all available Quotes attributed to the Film

Given that a user is presented a list of Quotes attributed to a Film
When they click on a specific Quote
Then they should see the Quote, and to which Actor the Quote is attributed

Scenario: Adding new quotes

Given that a user navigates to the Cinequotes home page
When they click the “Add Quotes” button
Then they should should see a form where they input a Film Title, Actor and Quote
And they should see a Submit button which when clicked, will add the Quote to Cinequotes

Scenario: Supporting multiple languages

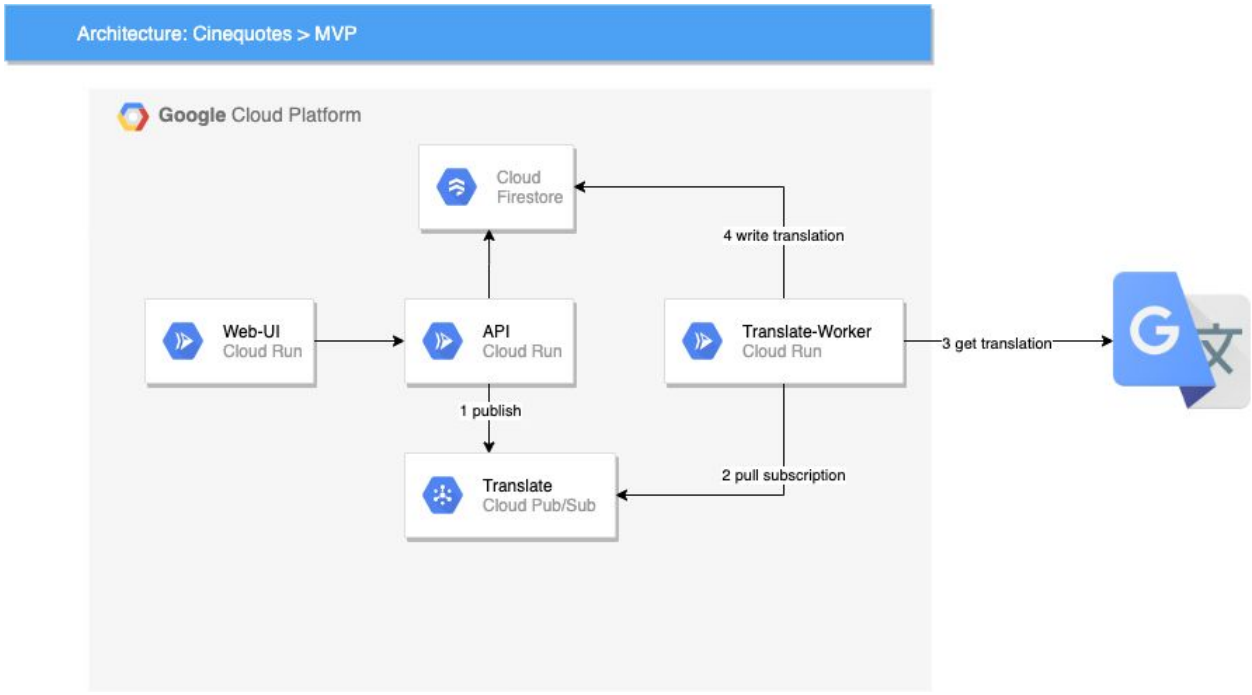
Given that a user has set their language preference to English
When they click on a specific Film
Then they should see a list of all Quotes attributed to the Film
And the quotes should be displayed in English

Given that a user has set their language preference to Francais (French)
When they click on a specific Film
Then they should see a list of all Quotes attributed to the Film
And the quotes should be displayed in French

Architecture

After working with the Product Manager in the morning, you and the team have an afternoon session with the Architect. Your Architect has deep knowledge of Google Cloud Platform services, and has provided a diagram to show how the Cinequotes service should be implemented.
The Architect walks the team through the diagram, and you take the following notes:

- The User Interface will be a web client for the API
- Google Cloud Firestore will be used as the database for Cinequotes.
- The API will query Firestore directly in order to return film quotes data.
- The API will write data to Firestore when new film quotes are added to the database.
- Multiple language support will be provided by using an asynchronous flow:
    - When a new film quote is added to the database it will first be written to Firestore.
    - A message will also be published to a Cloud PubSub *Topic* called Translate
    - A separate service called the Translate-Worker will use a PubSub *pull subscription* to subscribe to these messages.
    - Upon receiving a message, the Translate Worker will GET the English to French translation for the *quote* from the Google Cloud Translation API.
    - The Translate-Worker will then write the French translation to the Firestore document for the quote.



In order to achieve a fast feedback loop during development, it is recommended to you that you use emulators and mocks. Specifically, you can utilize the Firebase emulators for Firestore and PubSub. The architect has given you some helpful tips on how to get the emulators running locally using Docker. You should also mock the responses from Google Cloud Translation API during development rather than accessing the real API. For this you could, for example, use the popular nock library.

Firestore emulator

The following Dockerfile provides a Firestore emulator installation:

```
FROM google/cloud-sdk:290.0.1
RUN apt-get install google-cloud-sdk-firestore-emulator
CMD [ "gcloud", "beta", "emulators", "firestore", "start",
"--host-port=0.0.0.0:8505" ]
EXPOSE 8505
```

Run with:

```
docker run --rm -d -p 8505:8505 <image>
```

To connect to the Firestore emulator from Node.js, ensure that you start the process with the environment variable `FIRESTORE_EMULATOR_HOST` set to `<host>:8505`

```
// dependency @google-cloud/firestore: 4.2.0
const { Firestore } = require('@google-cloud/firestore');
const client = new Firestore({
  projectId: 'dummy' // projectId doesn’t matter while using the
emulator
});
```

PubSub emulator

The following Dockerfile provides a PubSub emulator installation:

```
FROM google/cloud-sdk:290.0.1
CMD [ "gcloud", "beta", "emulators", "pubsub", "start",
"--host-port=0.0.0.0:8085" ]
EXPOSE 8085
```

Run with:

```
docker run --rm -d -p 8085:8085 <image>
```

To connect to the PubSub emulator from Node.js, ensure that you start the process with the environment variable `PUBSUB_EMULATOR_HOST` set to `<host>:8085`

```
// dependency @google-cloud/pubsub: 2.3.0
// dependency @grpc/grpc-js: 1.1.3
const pubsub = require('@google-cloud/pubsub');
const grpc = require('@grpc/grpc-js');
const [pubsubHost, pubsubPort] =
process.env.PUBSUB_EMULATOR_HOST.split(':');
const options = {
  projectId: 'dummy', // projectId doesn’t matter while using the emulator
  servicePath: pubsubHost,
  port: pubsubPort,
  sslCreds: grpc.credentials.createInsecure()
};
const client = new pubsub.PubSub(options);
```

Films and Quotes to use for development

Film_Title	Actor	Quote
Scarface	Al Pacino	EN: Say hello to my little friend. FR: Dis bonjour à mon petit ami.
The Wizard of Oz	Judy Garland	EN: Toto, I've got a feeling we're not in Kansas anymore. FR: Toto, j'ai le sentiment que nous ne sommes plus au Kansas.
Star Wars	Harrison Ford	EN: May the force be with you. FR: Que la force soit avec toi.
Taxi Driver	Robert DeNiro	EN: You talking to me? FR: Tu me parle?
Jerry Maguire	Renee Zelwegger	EN: You had me at hello. FR: Vous me avez eu à bonjour.

Mock response for Google Translation API request:

Code: 200

```
{
  "data": {
    "translations": [{
      "detectedSourceLanguage": "en",
      "translatedText": "Dis bonjour à mon petit ami."
    }]
  }
}
```

After a good day planning with the team, you are ready to start development! Your mission is to:

- Develop the API
- Develop the Translate Worker
- Your teammate will be developing the Web UI, so you should NOT develop this. You just need to make sure she is able to use the API.
- Because each service should be independently developed and deployed, you should create one git repository for the API and another for the Translate Worker.
- You will only be setting this up to run locally, so you do not need to worry about deploying this. You will however need to run the supporting infrastructure, so be sure to get the emulators running locally to support your testing. (Hint: A docker-compose file might be really handy to create).
- Focus on your tests. Each component should be testable in isolation. Think about the testing pyramid – remembering that you want a fast feedback loop with tests that are fast and reliable. Unit tests are the developers best friend.
- When you think you are done, push your code to a publicly accessible Github repo. If you don't have a Github account, you can create one for free.

Notes: You do not need to worry about Authentication or Authorization. You also do not need to worry about pagination or sorting. Assume that all new quotes are added in English and translated to French. Lastly, you are not expected to make real calls to the Google Translate API, you should only mock the responses that are expected from the API.

And of course if you have problems, ask your teammates! For this exercise you can email shayne.clausson@extendaretail.com with any questions or problems.