İhsan Doğramacı Bilkent University

CS 443 Cloud Computing

Project Proposal

February 27, 2023

Bilgehan Akcan 21802901
Miray Ayerdem 21901987
Ferhat Korkmaz 21901940

Overview

Our project is a cloud-native movie ticket purchase system, which will be a web application that utilizes Google Cloud technologies. We will use cloud-native design to build the application, allowing us to scale it in response to customer demand. Previous and now showing movies will be stored in Google Cloud Storage. Google Cloud Functions will be used by simply being triggered periodically by Google Cloud Scheduler. These services will provide a robust and scalable system. The application will allow the user to log in the system, select a movie now showing in theaters, direct the user to select showtime and seat, and pay the ticket price. Additionally, it will be possible to see the additional information about movies such as cast, director, rating, genre. Moreover, users will get an email confirmation of the ticket purchase. Users will also be able to view movies they have previously watched in their profile pages. Also, users will be notified via email if they have a movie ticket for a film that will be shown the following day.

Veni Vidi Cloud / Team Members

21802901, Bilgehan Akcan, CS, 2023, bilgehan.akcan@ug.bilkent.edu.tr
21901987, Miray Ayerdem, CS, 2023, miray.ayerdem@ug.bilkent.edu.tr
21901940, Ferhat Korkmaz, CS, 2023, ferhat.korkmaz@ug.bilkent.edu.tr

High-Level Choices

Implementation Language and Relevant Environment Choices

For the backend application that will be hosted in a Compute Engine instance, we will be using Javascript's Node.js framework since it is easy to create a server with it and it has less boilerplate. For the client-side of the application that will be hosted in a Cloud Storage instance, we will be using React.js. The reason we chose React.js is because all of us have experience with it and it is pretty common around the world of developers. For the development phase, we will utilize Git and GitHub to main version control. Also, we will try to stick with the local development before deploying the codes to the cloud.

Dataset

As a dataset, we have enrolled in themoviedb.org's HTTP API. The Movie Database(TMDb) API provides access to a wide range of information about movies containing titles, summaries, posters, trailers, release dates, ratings, and more. It has a request path that returns the movies that are currently playing around the world [1].

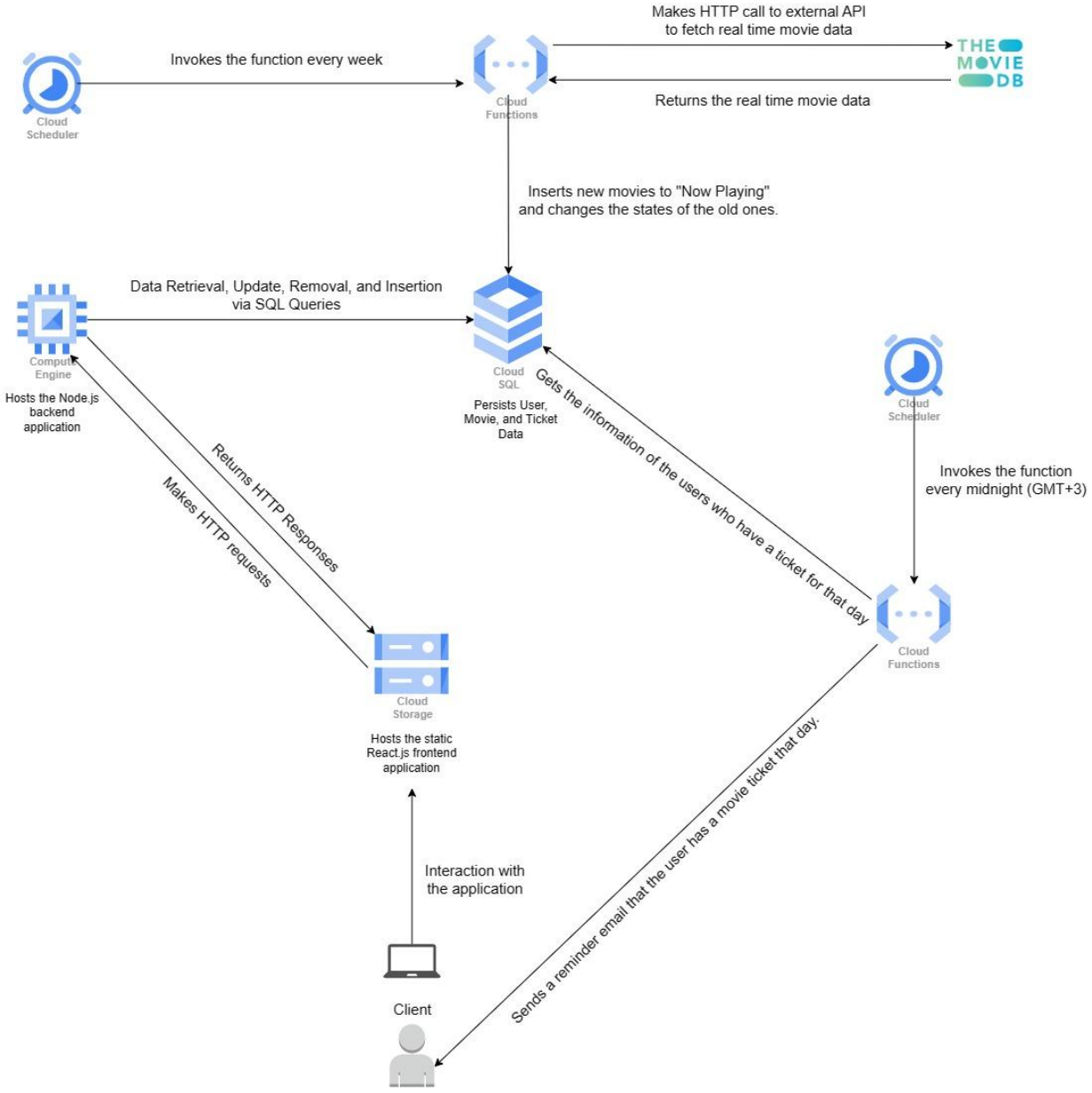
Sample response from the API:


```
{
            "adult": false,
            "backdrop_path": "/xDMIl84Qo5Tsu62c9DGWhmPI67A.jpg",
            "genre_ids": [
                28,
                12,
                878
            ],
            "id": 505642,
            "original_language": "en",
            "original_title": "Black Panther: Wakanda Forever",
            "overview": "Queen Ramonda, Shuri, M’Baku, Okoye and the
Dora Milaje fight to protect their nation from intervening world
powers in the wake of King T’Challa’s death.  As the Wakandans strive
to embrace their next chapter, the heroes must band together with the
help of War Dog Nakia and Everett Ross and forge a new path for the
kingdom of Wakanda.",
            "popularity": 3952.862,
            "poster_path": "/sv1xJUazXeYqALzczSZ3O6nkH75.jpg",
            "release_date": "2022-11-09",
            "title": "Black Panther: Wakanda Forever",
            "video": false,
            "vote_average": 7.4,
            "vote_count": 3583
}
```


We will utilize this API to populate our movie database dynamically. As seen from the high level architecture diagram, we will schedule a Cloud Function weekly to update our database. Users should sign up to our application in order to purchase tickets. When they purchase tickets with

their accounts, we will collect their personal information like their names, email addresses, and payment information safely. Moreover, we will store the ticket sales to be able to track the available seats in our application.

High Level Architecture Diagram



The Cloud Architecture that we are planning to use mainly has 1 Compute Engine Instance, 1 Cloud SQL Instance, 1 Cloud Storage bucket, 2 Cloud Functions deployed, and 2 Cloud Schedulers. The compute engine will host the backend application, which is a Node.js application. It will listen to HTTP requests from the web application and based on the requests,

to generate responses, it will access the database, which is a Cloud SQL instance. The Cloud Storage bucket will host the React.js frontend application [2]. Users use that website to interact with the application. The Cloud Function at the top will be invoked by a Cloud Scheduler every week. The function will fetch the currently playing movies worldwide and update our movie database respectively. The other Cloud Function on the right hand side of the diagram will be invoked by a Cloud Scheduler every day at midnight (Istanbul Timezone). It will send an email to the users who have a movie ticket on that day. The function can do it by accessing the data that is stored in the Cloud SQL instance.

References

[1] The Movie Database. https://developers.themoviedb.org/3/movies/get-now-playing. Feb. 27, 2023.

[2] wesionaryTEAM. “Deploy Your Static React Application on Google Cloud Platform”. https://articles.wesionary.team/deploy-your-static-react-application-on-google-cloud-platform-bc8afb9b9b6. Feb. 27, 2023