OutFit AI

By Daniel, Tim, and Vicente

App Name: Outfit AI

App URL: https://outfit-cs651.uc.r.appspot.com/

Purpose: This app allows users to upload a photo locally or from a recent post on X/Twitter to the website and give a relevant clothing prompt. The app then generates a clothing suggestion based on the image and prompt, which the user can then save to their account or search Google for similar items.

Status: Brief description of what is and is not working as it relates to your proposal. Are the following working:

- Frontend (React)
- Backend (Express/NodeJS)
- Social Netowork X Authentication + User Photo Retrieval
- Google Cloud Vision API calls
- Google Gemini API calls
- Google Analytics
- Google Cloud logging

Proposal

Fo over proposal: Bring up document of proposal and show the text and talk about what was able to achieve and not

Our proposal in the GitHub wiki.
The core features of the proposal were successful, however a few things had to change.

- We swapped to use X's API
- We had to lower the output to 1 image with no extra text because just 1 image already takes long enough
- Our logs aren't as detailed as we'd like

DEPLOYMENT

NOW go to the Google Cloud project & show the deployed URL pointing to your webapp

Our app is deployed to:
https://outfit-cs651.uc.r.appspot.com/

Run Demonstration

Now run app & functionality of app and fully demonstrate it

We will: create a user, login, generate an image, save it to the profile, logout, login as a different user, show a different profile, create an image with twitter api

Google Cloud Console/Dashboard

Bring up your choice of Google Cloud PAAS & show the usage information of the webapp.

Google Cloud data(base) solution

Discuss the Google Cloud data solution you implemented (i.e. Firestore, etc). Show the contents of the actual app data that is collecting/ being used. SHOW from the console directly.

Show before and after when data is altered/created (if your app does this)

For data, we used two main pieces: Datastore and Cloud Storage. User auth profiles and image metadata are saved in Google Cloud Datastore using the user's email as the key, so their history doesn't disappear when the server restarts. The actual generated images are stored in a Cloud Storage bucket under per-user folders, and we return a public or signed URL to the client. In short: metadata in Datastore, image files in Storage

Google Cloud Vision solution

Discuss how Google Cloud Vision API is used.

Show results in app / data created & stored

We use the google cloud vision api to extract objects and labels from a picture, then use the gemini image api to generate images based on the data from those object and labels. If a user uploads a photo, it's going to extract the piece of apparel with the highest score, usually like a shirt or hat. If they upload and submita query "top", we are able to find something similar in that photo "t-shirt" through a series of whitelists and filtering.

Google Gemini solution

Discuss how GoogleGeminiAPI is used.

Show results in app / data created & stored

Gemini was discussed with vision on the previous slide.

Google Analytics

Discuss how Analytics is used

Show results in console

We barely used google analytics service and instead mostly relied on structured logs. Each request logs things like user, path, and duration, and image generations log the user plus details about the item. We’re not currently calculating totals (like number of users or average generations per user) inside the app those would need to be pulled later by querying the logs in Cloud Logging.

Google Cloud Logging

Discuss how Google Cloud Logging is used

Show results in console

We use Winston with the LoggingWinston transport so our structured logs are sent directly to Cloud Logging. Middleware logs each HTTP request with helpful context like user ID, timing, IP, and user agent, and errors are logged the same way. This gives us a clear trail in Cloud Logging for debugging and understanding how the app is behaving.

Summary

- If you had and problems that didn't work, show them (the results if any) and discuss why you think it didn't work
- Biggest problem was using X API. Testing was difficult and we were only allowed to authenticate every 15 minutes

- Discuss how you might improve this app
- Right now everything is mostly in raw logs. A nice next step would be to stream those logs and build proper analytic dashboards for metrics like active users, generations per user, and most popular apparel type.
- I would probably have use some other APIs. There was one for shopping but it had a pretty hard cap of 50 api calls a month and then $50 fee after. Definitely something other than X api

- Discuss what you learned from making this app- what were the challenges, what you learned, how you can help others.
- Biggest take away is learning more about APIs and how to integrate them.
- Using Google Cloud console and features went smoothly from the usage in class
- You don’t have to be an expert in all of the main parts of a web app (frontend, backend, database and how everything connects), but knowing the basics makes it a lot easier to use cloud services and not feel totally lost.