Assignment 1

GD Radloff, (u21435872)

May 26, 2024

Overview

Cloud security is shifting from focusing on the edge of the architecture, to securing the inside of the architecture[6]. The Zero Trust model for instance constantly verifies users and devices before granting access[8], which is something that I used in my own solution, and that google has adopted[9]. Another development that supports this claim in a slightly different way is the Cybersecurity Mesh Architecture, which secures **individual** elements within the cloud environment instead of the entire network[7]. And like it is with everything these days, Artificial Intelligence is playing a big role in automating threat detection and response[5], allowing us to react faster to security incidents because of Santa's little slaves working tirelessly in the background. Cloud Computing Security is about solving the problem of how to build a system of components where any component can be linked to any other component in an intuitive and easy to understand manner. It is not as simple as access control or lock everything down. It is about making a streamlined, scalable, and secure development environment.

The configuration of the firestore rules makes it impossible for: users to read data from documents they don't own or that are not public, based on their auth token.[3] The rules also restrict all writes outside of the admin sdk (cloud functions) to occur. The client authenticates themselves without interacting with my cloud functions, but their authenticity is can still be validated (even though none of their writes are trusted). This emulates a zero trust approach combined with a mesh, where clients "secure and validate themselves". Because of how the rules are set to allow reads, the user requires a auth token, this auth token is always validated, this means even if the website breaks, because users read based on firestore rules, it is not possible for data to be "leaked". [4] In this way the Confidentiality of the system is secured. Auth tokens in combination with Identity Platform and Google Logging provide a digital forensic readiness solution for preserving Accountability and Authenticity behind every user action and behavior in the system. Every artifact in the system including functions and the system itself become entities in the logs that contain important security information that is integrated with Google Monitoring to alert me of any suspicious events, activity, or behavior, from any entity in the entire architecture. The integrity of references within firestore is strongly validated with "hooks"

that listen to firestore events such as writes and updates. Even after a function commits a document, it is validated and additional processing can occur. Every object is validated using a strictly parsed zod schema which makes it impossible for any uncharacteristic or ill defined object to enter or exit the system. This addresses one of the downsides of schemaless databases [10]. This ensures integrity of the data even across server and client as both share the same shared library and types. Users are given least privilege with regards to document modification in cloud functions, and administrators actions are carefully logged and backups are made of all administrators changes which can be reverted to at any time if needed. Finally, because most reads are client side, and because the entire application (even hosting) is serverless, the availability of the application is all but ensured as it can be scaled up and down on demand. Each function has its own dedicate set of computing resources to saturate, meaning each individual operation of the backend can be scalled up and down individually to meet demand and can ofcourse be extend or pipelined into other google cloud services. The system is protected by the google cloud content distribution network, ensuring high availability, and the exposed functions ignore all requests not made by a app with valid credentials (which is per domain). [1] [2]

To finnish of my bragging and to tie things up: the application utilizes new technologies such as SSR (even though there is no server in this case), a serverless architecture as stated before, the state of the art webframe work svelte with svelte -kit, and is fully configured and billed on a single platform.

Requirement Specification

Student requirements

Students must be able to:

- Log in using their existing credentials
- Register for a degree/degrees
- Register for modules
- Logout

Admin Requirements

Administrators must be able to:

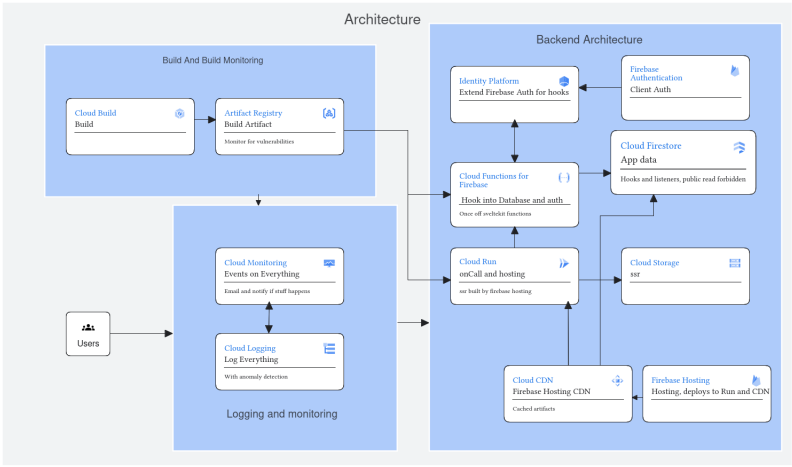
- Log in using their existing credentials
- Register new degrees
- Register new modules
- Add modules to existing degrees
- Remove modules from existing degrees

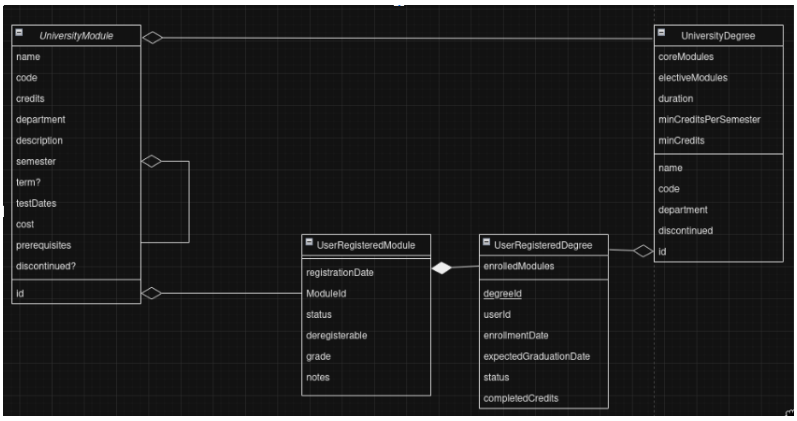
General Restrictions

- Users can **never** modify a document that does not belong to them.
- Users can **never** read from a document that belongs to another user.
- Administrators can **never** see the personal details of other users.
- Administrators can **not** escalate the privilages of another user.

UML and GCP

GCP Diagrams are the UML standard of google for google cloud, this shows my architecture deployment, with focus on security.





Deployment

In order to deploy, you have to first download/fork the repo. Once you have done that, make sure you have firebase -tools,turbo repo, and the firebase admin sdk installed globally using your package manager of choice. Run `pnpm i` (do not use `yarn` or `npm`). Once you have done that, login to firebase and either select this project or create a new project using this one as a source. Now run `pnpm run` build in the `/code` directory, this will build the project. The project is deployed in two steps:

Functions

Function deployment is a bit more cumbersome. First make sure that you have either run `turbo build` in `/code` or `run pnpm run build` in the`/code/functions` and `code/shared` directory. After that, move to the top directory and run `turbo prune @cos720project/functions` . This will create a directory labeled as `out`. Copy everything under `code/functions/lib` and move it to `code/out/functions/lib`. Open `code/out/functions/package.json` and remove `@cos720project/shared` under dev dependencies. DO not build. Within `code/out/functions` run `firebase deploy --only functions`. Whenever you want to redeploy, just copy over the `/lib` file and run `firebase deploy` again. The first deployment will usually fail on a new project due to things being set up in gcloud and you may need to delete and re -upload functions. Once you have done that, you must go to the gcloud console, and open the Cloud Run section. For every function, you must expose it publicly so that it uses firebase Auth and not IAM, it will give you a warning but it's fine because it's just the default and firebase functions v2 doesn't change it to what it needs to be in order for onCall to work correctly. To do this, click on a function, navigate to security, and then select allow unauthenticated invocations. If you don't do this, you will get a misleading cors error in the application.

Frontend

Register and enable hosting in the firebase console, and add a app, copy and paste your App credentials (these can be public they are just used to identify the app to firebase and doesn't matter if others can see them, these are NOT your service account credentials, you do not need to download service account credentials for this project.), Navigate to `/code`, and run firebase deploy -only hosting and it should work. It will take a while.

Firestore Rules

`firebase deploy --only firestore` in the `/code` directory

References

[1] Ekaba Bisong and Ekaba Bisong. "An overview of google cloud platform services". In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners* (2019), pp. 7–10.

[2] Pankaj Chougale et al. "Firebase-overview and usage". In: *International Research Journal of Modernization in Engineering Technology and Science* 3.12 (2021), pp. 1178–1183.

[3] Ram Kesavan et al. "Firestore: The nosql serverless database for the application developer". In: *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE. 2023, pp. 3376–3388.

[4] Laurence Moroney and Laurence Moroney. "Using authentication in firebase". In: *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform* (2017), pp. 25–50.

[5] Samuel Oladiipo Olabanji et al. "AI-Driven cloud security: Examining the impact of user behavior analysis on threat detection". In: *Asian Journal of Research in Computer Science* 17.3 (2024), pp. 57–74.

[6] Godwin Olaoye and Ayuns Luz. "Future trends and emerging technologies in cloud security". In: (2024).

[7] Bruno Ramos-Cruz, Javier Andreu-Perez, and Luis Martínez. "The cybersecurity mesh: A comprehensive survey of involved artificial intelligence methods, cryptographic protocols and challenges for future research". In: *Neurocomputing* (2024), p. 127427.

[8] Sirshak Sarkar et al. "Security of zero trust networks in cloud computing: A comparative review". In: *Sustainability* 14.18 (2022), p. 11213.

[9] Rory Ward and Betsy Beyer. "BeyondCorp: A New Approach to Enterprise Security". In: *;login:* Vol. 39, No. 6 (2014), pp. 6–11.

[10] Asadulla Khan Zaki. "NoSQL databases: new millennium database for big data, big users, cloud computing and its security challenges". In: *International Journal of Research in Engineering and Technology (IJRET)* 3.15 (2014), pp. 403–409.