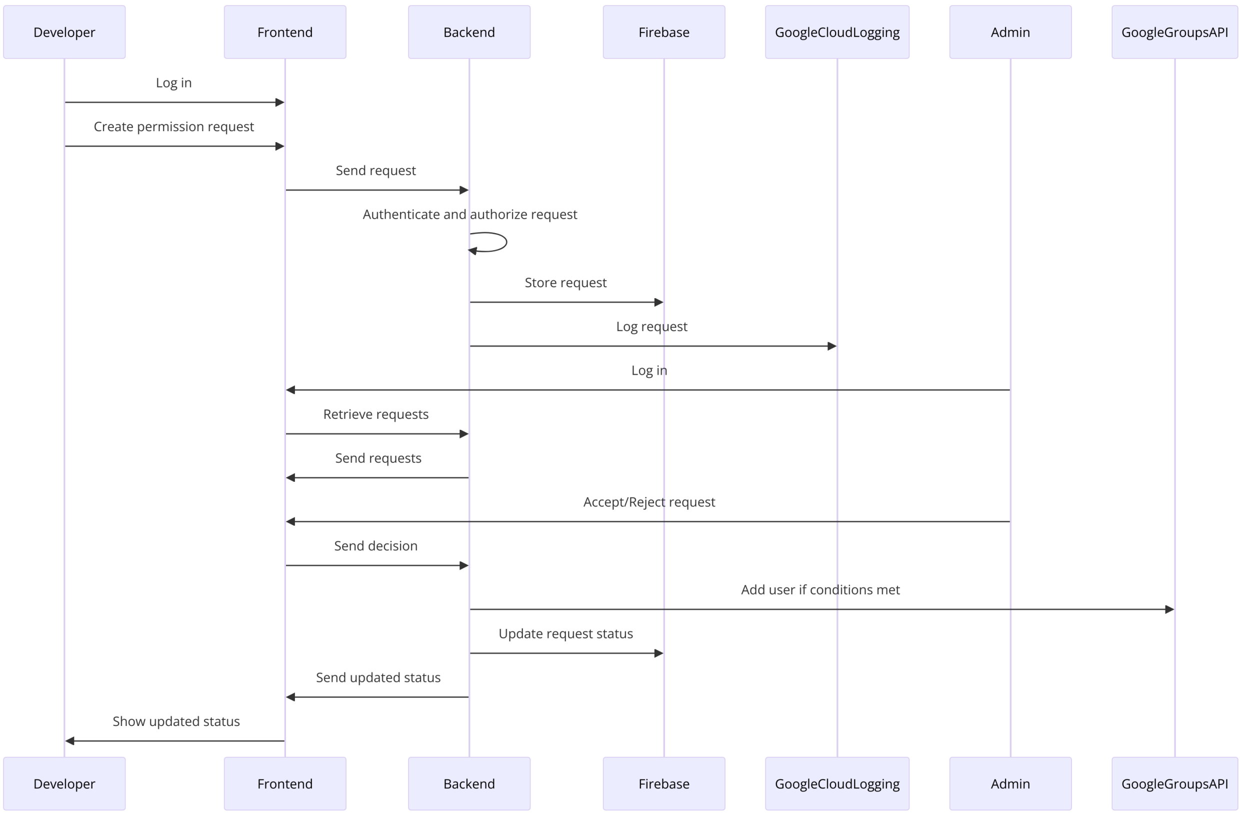
Detailed Architecture and Workflow

Overview

The objective is to develop a routine that enables developers to request elevated permissions through a web application, utilizing native Google Cloud Platform (GCP) components. The architecture includes the following key components:

1. **Front-End Application**: Built using React, styled with Tailwind CSS and Material UI (MUI), and deployed via Docker.
2. **Backend Services**: Google Cloud Functions powered by Flask to handle API requests.
3. **Authentication**: Google Identity Platform for user authentication with Gmail.
4. **Logging and Auditing**: Google Cloud Logging for tracking permission requests.

Architecture Diagram



Workflow

User Authentication

1. **Login**:
   - Developers log in using their Gmail accounts via the Google Identity Platform.
   - The application checks the user's role (developer or super admin) and renders the appropriate view.

Permission Request

1. **Request Form**:
    - Developers access the web application and request elevated permissions by filling out a form with the desired group and a description of the request.
    - The form submission triggers a `POST` request to the backend API.

Backend Services

1. **Handling Requests**:
    - The backend API (Google Cloud Function) receives the request and logs the details (user email, description, timestamp) using Google Cloud Logging.
    - If the requester is an owner/manager of the super admin group, they can view and manage all requests. Otherwise, developers can only view their own requests.
2. **Endpoints**:
    - **Get Groups**:
        - `GET /groups`: Returns a list of groups and roles.
    - **Get Requests**:
        - `GET /requests`: Returns a list of permission requests. Access is filtered based on the user's role (developer or super admin).
    - **Post Request**:
        - `POST /requests`: Accepts a new permission request.
    - **Accept Request**:
        - `POST /accept?id=<request-id>`: Accepts a pending request.
    - **Reject Request**:
        - `POST /reject?id=<request-id>`: Rejects a pending request.

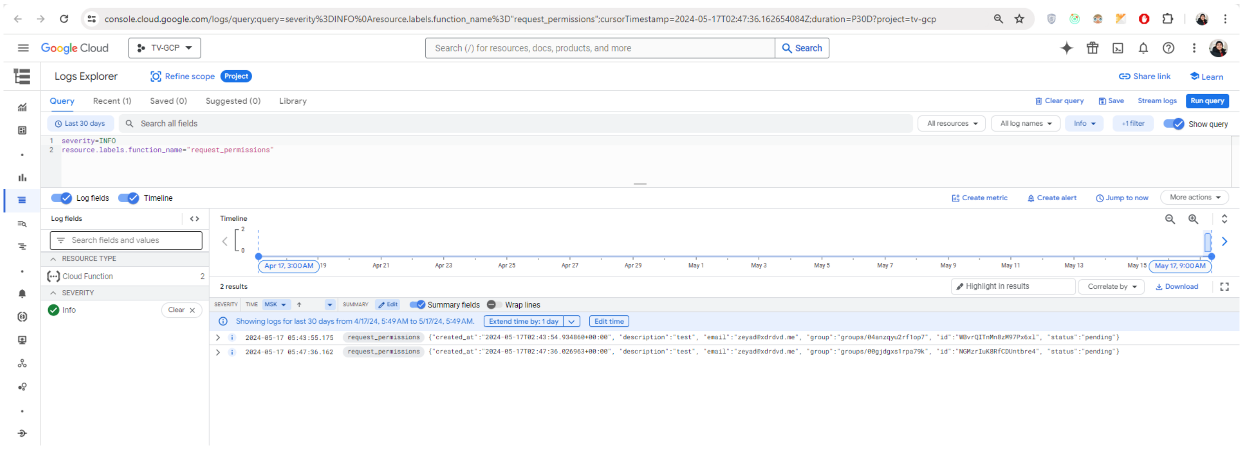
Logging and Auditing

1. **Logging Function**:
    - Google Cloud Logging is used to keep a detailed log of all actions.
    - The logging function writes log entries for each request and action taken.

```python
from google.cloud import logging

def write_entry(logger_name, entry):
    """Writes log entries to the given logger."""
    logging_client = logging.Client()

    # This log can be found in the Cloud Logging console under 'Custom Logs'.
    logger = logging_client.logger(logger_name)

    # Struct log. The struct can be any JSON-serializable dictionary.
    logger.log_struct(
        entry,
        severity="INFO",
    )
```

2. **Audit Logs**:

    - The log entries include the user's email, description of the request, and timestamp.
    - Actions such as request creation, acceptance, and rejection are logged for auditing purposes.
    - Check the logs link



```
console.cloud.google.com/logs/query;query=severity%3DINFO%0Aresource.labels.function_name%3D"request_permissions";cursorTimestamp=2024-05-17T02:47:36.162654084Z;duration=P30D?project=tv-gcp
```

```
severity=INFO
resource.labels.function_name="request_permissions"
```

2 results

Showing logs for last 30 days from 4/17/24, 5:49 AM to 5/17/24, 5:49 AM.

```
2024-05-17 05:43:55.175  request_permissions {"created_at":"2024-05-17T02:43:54.934860+00:00", "description":"test", "email":"zeyad@xdrdvd.me", "group":"groups/04anzqyu2rf1op7", "id":"WBvrQITnMn8zM97Px6xl", "status":"pending"}
2024-05-17 05:47:36.162  request_permissions {"created_at":"2024-05-17T02:47:36.026963+00:00", "description":"test", "email":"zeyad@xdrdvd.me", "group":"groups/00gjdgxs1rpa79k", "id":"NGMzrIuK8RfCDUntbre4", "status":"pending"}
```

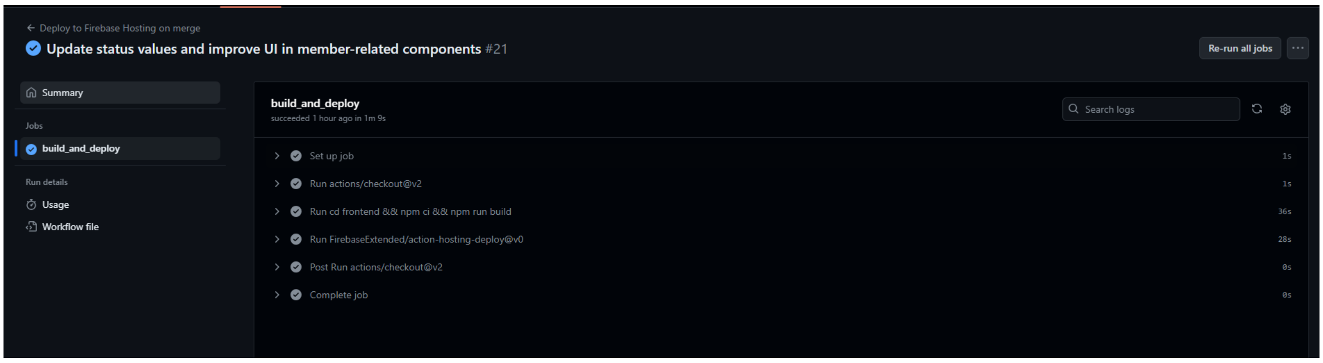
Dockerization and Deployment

1. **Containerization**:

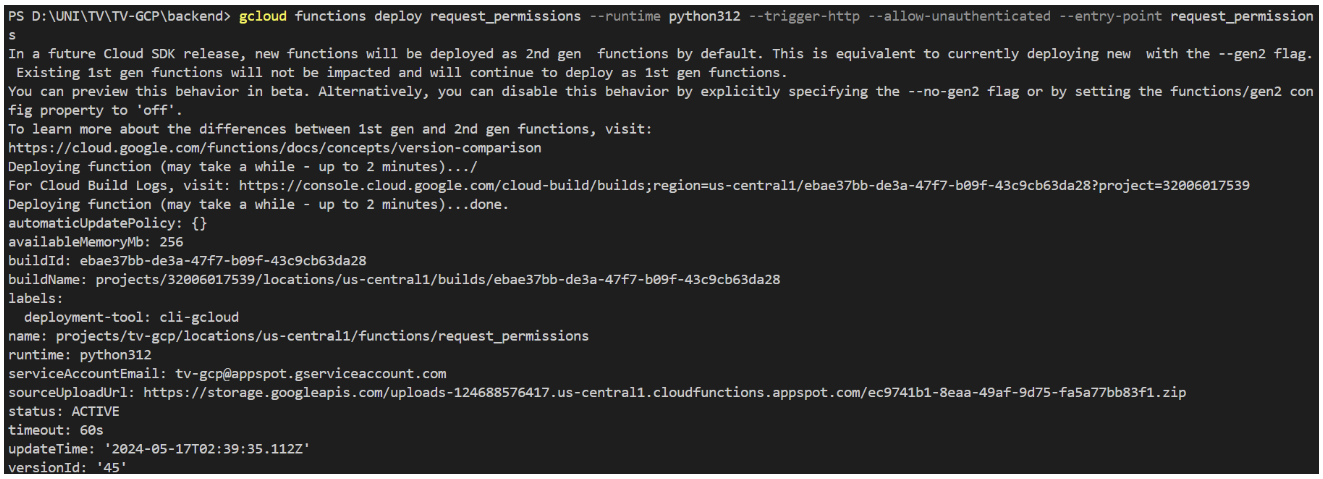
    - The front-end application is containerized using `Docker`, However, we have user the `CI/CD` of firebase to deploy since it is a static simple web app.

2. **Deployment**:

    - The front-end application is deployed to Firebase Hosting using the Firebase CLI.
    - The backend Google Cloud Functions are deployed using GCP's deployment tools.



```
← Deploy to Firebase Hosting on merge
Update status values and improve UI in member-related components #21          Re-run all jobs

Summary            build_and_deploy
                   succeeded 1 hour ago in 1m 9s
Jobs
build_and_deploy   > Set up job                                              1s
                   > Run actions/checkout@v2                                 1s
Run details        > Run cd frontend && npm ci && npm run build             36s
Usage              > Run FirebaseExtended/action-hosting-deploy@v0          28s
Workflow file      > Post Run actions/checkout@v2                            0s
                   > Complete job                                            0s
```



```
PS D:\UNI\TV\TV-GCP\backend> gcloud functions deploy request_permissions --runtime python312 --trigger-http --allow-unauthenticated --entry-point request_permission
s
In a future Cloud SDK release, new functions will be deployed as 2nd gen  functions by default. This is equivalent to currently deploying new  with the --gen2 flag.
 Existing 1st gen functions will not be impacted and will continue to deploy as 1st gen functions.
You can preview this behavior in beta. Alternatively, you can disable this behavior by explicitly specifying the --no-gen2 flag or by setting the functions/gen2 con
fig property to 'off'.
To learn more about the differences between 1st gen and 2nd gen functions, visit:
https://cloud.google.com/functions/docs/concepts/version-comparison
Deploying function (may take a while - up to 2 minutes).../
For Cloud Build Logs, visit: https://console.cloud.google.com/cloud-build/builds;region=us-central1/ebae37bb-de3a-47f7-b09f-43c9cb63da28?project=32006017539
Deploying function (may take a while - up to 2 minutes)...done.
automaticUpdatePolicy: {}
availableMemoryMb: 256
buildId: ebae37bb-de3a-47f7-b09f-43c9cb63da28
buildName: projects/32006017539/locations/us-central1/builds/ebae37bb-de3a-47f7-b09f-43c9cb63da28
labels:
  deployment-tool: cli-gcloud
name: projects/tv-gcp/locations/us-central1/functions/request_permissions
runtime: python312
serviceAccountEmail: tv-gcp@appspot.gserviceaccount.com
sourceUploadUrl: https://storage.googleapis.com/uploads-124688576417.us-central1.cloudfunctions.appspot.com/ec9741b1-8eaa-49af-9d75-fa5a77bb83f1.zip
status: ACTIVE
timeout: 60s
updateTime: '2024-05-17T02:39:35.112Z'
versionId: '45'
```

Team Contributions

1. **Front-End Development**:
   - Responsible: `Zeyad Alagamy`
   - Email: `z.alagamy@innopolis.univeristy`
   - Tasks: Developed the React application, integrated Tailwind CSS and MUI, implemented the login and home pages, assisted with `GCP API` integration, and reports.

2. **Backend Development**:
   - Responsible: `Ahmed Saber`
   - Email: `a.soliman@innopolis.univeristy`
   - Tasks: Developed the Flask API, implemented Google Cloud Functions, defined API endpoints, and set up the environment user groups and accounts, and demo video.