



**KAZAKH-BRITISH
TECHNICAL
UNIVERSITY**

Final Project Report
Cloud-Powered Event Management System

Done by:

Amen Azat, 21B030774

23.11.2024

Almaty, 2024

1. Executive Summary

The goal of this project is to develop a secure, scalable and efficient event management system using various Google Cloud services. The app will allow users to create, manage and attend events by providing features such as registration, ticket purchase, notifications and real-time updates.

2. Table of Contents:

- Introduction
 - System Architecture
 - Table Entities
 - Development Process
 - API Design and Implementation
 - Database Design and Optimization
 - Event-Driven Architecture
 - Machine Learning Integration
 - Security Measures
 - Scalability and Performance
 - Challenges and Solutions
 - Conclusion
 - References
 - Appendices
-

3. Introduction

Background:

- Google Cloud Platform is one of the best cloud platforms on a par with platforms such as Amazon Web Services (AWS), Microsoft Azure. The main advantage of such platforms is scalability, flexibility and fault tolerance. Also, an important advantage of such cloud platforms is a wide range of ready-made services and tools, which significantly saves development time and resources.

The Google Cloud Platform offers a user-friendly console with a user interface and analytics tools. GCP includes technologies such as the Compute Engine, App Engine, Cloud Kubernetes Engine, Cloud Function and Cloud Endpoints. Various cloud storages such as Cloud SQL, Cloud Storage, Firestore for data storage. And a wide range of different services, technologies and APIs.

Project Goals:

1. Creating a database to store users, events, tickets, etc.
2. Implement API endpoints for user registration, event creation, and ticket purchase.
3. Containerization of the application using Docker and Kubernetes.
4. Creating an event-driven architecture using Pub/Sub and Cloud functions.
5. Protecting the application with authentication, data authentication, and HTTPS.
6. Scaling the application using GKE features.

Scope:

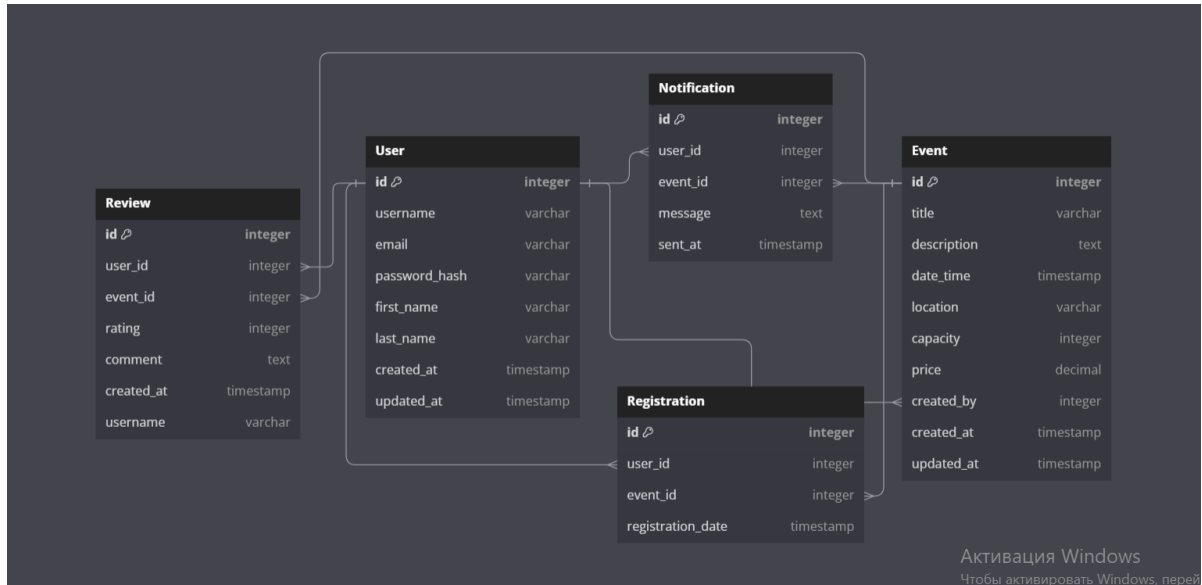
- In my project included various technologies such as Flask (for backend), React (for frontend), Docker (for containerization) and GCP services that I listed above. This all helped

me to develop event-driven, scalable and flexible web application. The limitations are: time frame set by the teacher, some requirements for the implementation of the project, as well as limited financial opportunities.

4. System Architecture

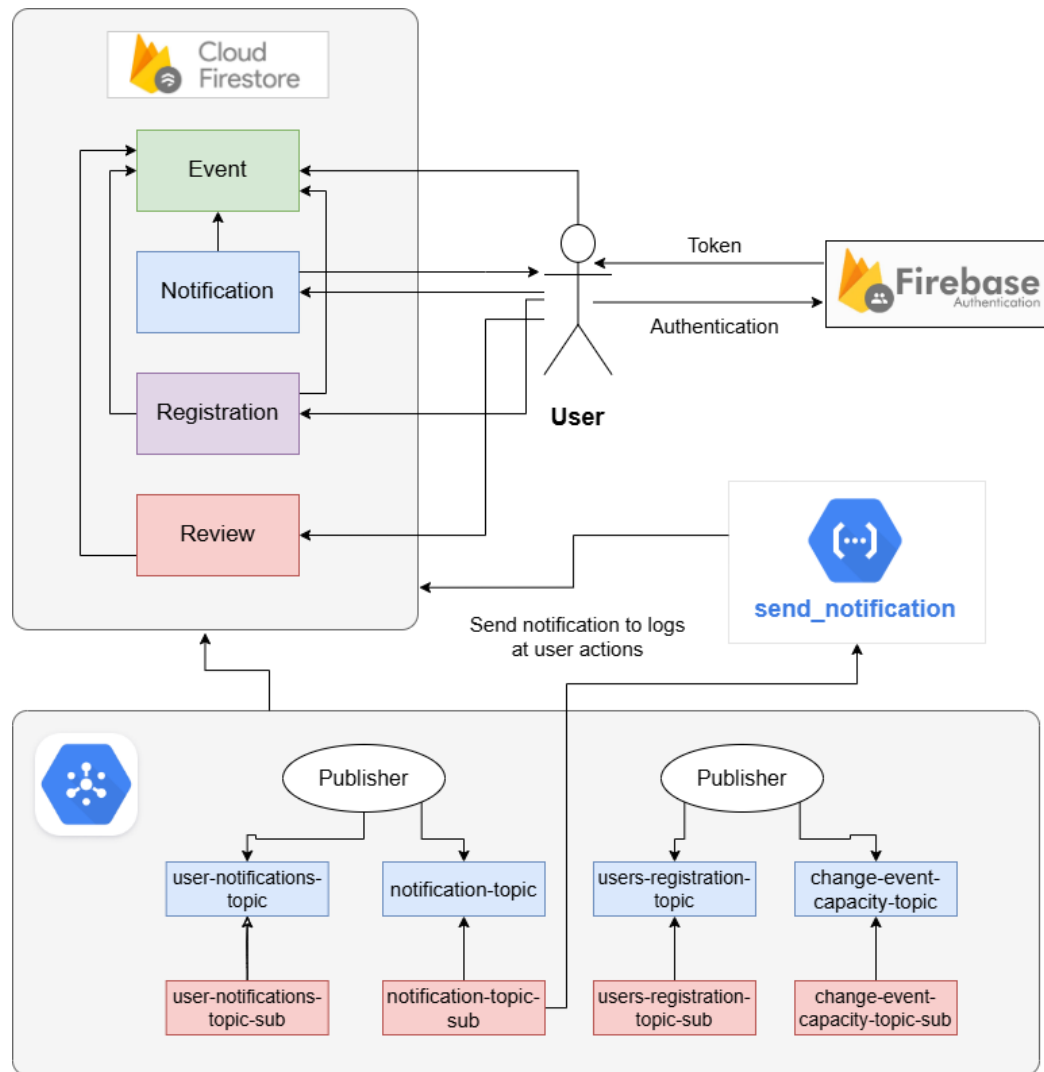
The architecture of the system is event-driven. Below I leaved diagrams with key components of the event management system:

- **Database Diagram** with entities and their relationships:



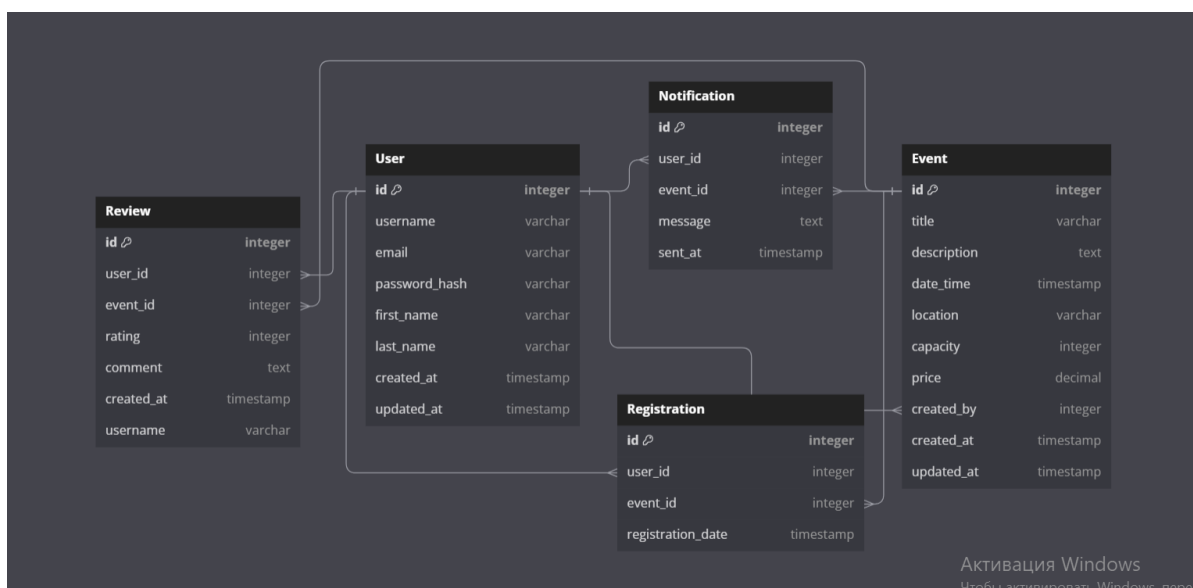
Link: <https://dbdiagram.io/d/Event-Management-System-67568cbbe9daa85aca14f4ce>

- **System Design Diagram** with Google Cloud services:



5. Table Entities

Instead of manually writing down all entities and their relationships, I used the DbDiagram platform (*Free online tool for drawing entity relationship diagrams by writing code*) to create them. In the picture below, you can see each entity and their connections between each other.



The main one is **User**; nobody can exist without User. Because only user can create events and manage them, leave reviews and register to events. Also, if there is no user, then notifications will have nowhere to go.

Event also very important, because all system built around events. User can create and manage events, leave review to event and register to event.

Review and **Registration** is related to event parts because they are really part of event. Users leave review to event and register to event.

And **Notifications**, its also important part of my app, because notifications sending when user do something in the system. They are sent to the logs and to the users, for example when User events got reviews and when someone register to User events.

Also, I want to share link to this diagram. So, you can explore it by yourself:

- <https://dbdiagram.io/d/Event-Management-System-67568cbbe9daa85aca14f4ce>

6. Development Process

Technologies Used:

In my project I used following technologies:

1. Flask – python framework, used to develop server side of my project
2. React – javascript framework, used to develop client side of my project
3. Docker – used for containerization my application
4. Firebase (Firebase Authentication and Firestore) – used for users authentication and data storage
5. Google App Engine – for deploying my server side
6. Google Cloud Functions – serverless functions used to logging user actions in system
7. Google Cloud Pub/Sub – used to build even-driven system
8. Google Cloud Endpoints – used to identify my app endpoints

Implementation:

- **Google Cloud SDK and Cloud Shell:**

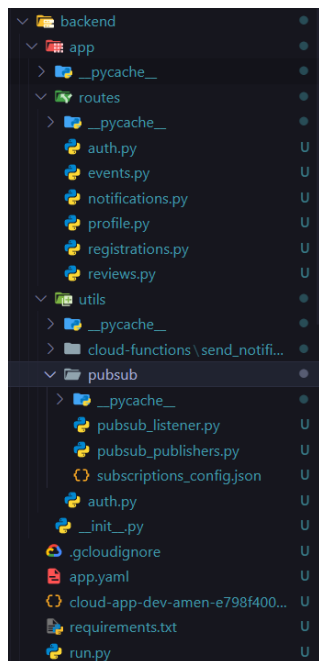
The Google Cloud SDK is a powerful tool for working with Google Cloud and its services directly from the terminal of a local device. Therefore, I installed the Google Cloud SDK on my device to interact directly with the Google Cloud Platform.

Cloud Shell is a direct console, a development environment from Google Cloud, which is located right in the console itself. The first option was more convenient for me, so I used it only.

My choice fell on the Google Cloud SDK, because it is much faster and more convenient for me, so I can access some resources or GCP data directly from the IDE terminal in which I write code.

- **Google App Engine:**

1. First thing I did, it of course writing code my project. I used flask framework and firebase for that.



2. Secondly, after finishing developing, I created app.yaml configuration for deployment to App Engine:

```
Final Project > backend > app.yaml
1 runtime: python39
2
3 entrypoint: gunicorn --bind :8080 run:app
4
5 handlers:
6   - url: /*.*
7     script: auto
```

3. Next, I added requirements.txt with all libraries and dependencies for project.

```
Final Project > backend > requirements.txt
1 Flask==3.0.0
2 requests
3 firebase_admin
4 google-cloud-pubsub
5 flask_cors
6 uuid
7 gunicorn
8 datetime
```

4. And finally, deploy application to app engine using gcloud app deploy:

```
azikkw@DESKTOP-TLGP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A/Final Project/backend (master)
$ gcloud app deploy
Services to deploy:

descriptor:          [E:\Study I KBTU\Semester VII\Cloud Application Development I Serek A\Final Project\backend\app.yaml]
source:              [E:\Study I KBTU\Semester VII\Cloud Application Development I Serek A\Final Project\backend]
target project:      [cloud-app-dev-amen]
target service:      [default]
target version:      [20241213t075753]
target url:          [https://cloud-app-dev-amen.uc.r.appspot.com]
target service account: [cloud-app-dev-amen@appspot.gserviceaccount.com]

Do you want to continue (Y/n)? Y

Beginning deployment of service [default]...

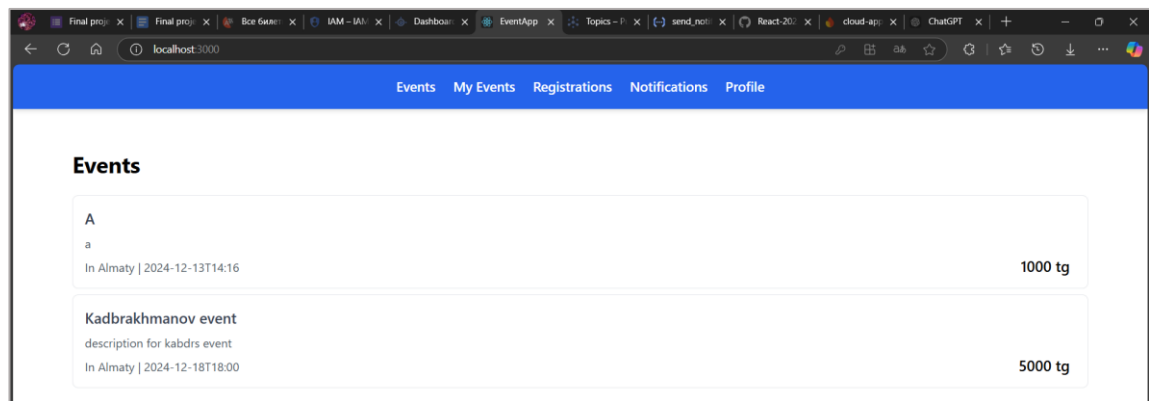
[Progress bar: = Uploading 14 files to Google Cloud Storage =]

File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://cloud-app-dev-amen.uc.r.appspot.com]

You can stream logs from the command line by running:
$ gcloud app logs tail -s default

To view your application in the web browser run:
$ gcloud app browse
```

5. And finally, front side of the project works properly, and it means that back side also works perfect:



- **Google Cloud Functions:**

I implemented “send_notification” cloud function to my system:

1. Firstly, I created a function code using python:

```
Final Project > backend > app > utils > cloud-functions > send_notification > main.py > send_notification
1 import functions_framework
2
3 @functions_framework.http
4 def send_notification(request):
5     if request.content_type != 'application/json':
6         return {"error": "Unsupported Media Type"}, 415
7
8     request_json = request.get_json()
9     notification = request_json['notification']
10    message = notification['message']
11
12    print(message)
13    return message, 200
```

This function sends notification (in logs) related to all actions in the system, such as authentication, event managing, registration to events and reviewing.

2. Add requirements.txt file with dependencies:

```
Final Project > backend > app > utils > cloud-functions > send_notification > requirements.txt
1 functions-framework
```

3. Deployed cloud function using following configuration

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/s
$ gcloud functions deploy send_notification \
--runtime python39 \
--trigger-http \
--allow-unauthenticated \
--region us-central1
```

4. And finally integrated cloud function to the application using pub/sub

```
elif subscription_name == 'notification-topic-sub':
    requests.post(
        SEND_NOTIFICATION_URL,
        headers={'Content-Type': 'application/json'},
        json={"notification": data}
    )
    print(f"Notification sent: {data['message']}")
```

5. Function works real example:

The screenshot shows a web application with a 'Register' form. The form has fields for Username, First Name, Last Name, Email, and Password. The Username field contains 'lol', First Name contains 'lol', Last Name contains 'lol', Email contains 'lol@gmail.com', and Password contains '.....'. Below the form is a blue 'Register' button and a link 'Already have an account? Log In'.

Below the form is a terminal log showing the following requests:

```
2024-12-13 06:14:49.244 NPT POST 200 203 B 12 ms python=requests/2.32.3 https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification
2024-12-13 06:14:49.244 NPT Notification 5929c462-7042-45de-92ba-81ae73f63ca0. User wOpfrh6qz7Qdr2ZSS3h5dR5xJL43 with username: lol registered and added to DB at 2024-12-13 06:29:44
2024-12-13 06:16:54.245 NPT GET 415 156 B 3 ms Chrome 131 https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification
```

- **Containerization:**

Steps taken to deploy the containerized application on GKE.

1. Create a **Dockerfile** with instructions for creating Docker Image:

```
Final Project > backend > Dockerfile
1 FROM python:3.9
2
3 WORKDIR /app
4
5 COPY . .
6
7 RUN pip install --no-cache-dir -r requirements.txt
8
9 EXPOSE 8080
10
11 CMD ["gunicorn", "--bind", "0.0.0.0:8080", "run:app"]
```


2. Create a Image using **docker build**:

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A/Final Project/backend (master)
$ docker build -t event-app .
[+] Building 131.1s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 208B                                0.0s
=> [internal] load metadata for docker.io/library/python:3.9       3.2s
=> [auth] library/python:pull token for registry-1.docker.io       0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [1/4] FROM docker.io/library/python:3.9@sha256:dd8b65c39a729f946398d2e03a3e6defc8c0cfec409b9f536200634ad6408b54 77.2s
=> => resolve docker.io/library/python:3.9@sha256:dd8b65c39a729f946398d2e03a3e6defc8c0cfec409b9f536200634ad6408b54 0.0s
=> => sha256:5bd71677db44bb63b94de61b6f1f95d5540b4ba2d6a8a6be4d19f422b25e0c2b 23.87MB / 23.87MB 3.8s
=> => sha256:f327fe247a0655427e3cfd615405bfb360e70acf34bf0385e7f1b0a07e4f8d8 6.17kB / 6.17kB 0.0s
=> => sha256:fd894e782a221820acf469d425b802be26aedb5e5d26ea80a650ff6a974d488 48.50MB / 48.50MB 40.9s
=> => sha256:551df7f94f9c131f2fec0e8063142411365f0alc88b935b9fac22be91af227e0 64.39MB / 64.39MB 33.9s
=> => sha256:dd8b65c39a729f946398d2e03a3e6defc8c0cfec409b9f536200634ad6408b54 10.35kB / 10.35kB 0.0s
=> => sha256:1b5da9c0831aad3a6704e68a966ce622d60971525c9c1e88cf29c2d7025d0de4 2.32kB / 2.32kB 0.0s
```

3. Tag Docker Image to send it to Google Container Registry (GCR):

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A/Final Project/backend (master)
$ docker tag event-app gcr.io/cloud-app-dev-amen/event-app:latest
```

4. Push Image to GCR:

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A/Final Project/backend (master)
$ docker push gcr.io/cloud-app-dev-amen/event-app:latest
The push refers to repository [gcr.io/cloud-app-dev-amen/event-app]
4733fc4981c3: Pushed
17d3ddb5f50d9: Pushed
624d2d65bb58: Pushed
fe5bbd4f8a42: Layer already exists
24f0c2413cd7: Layer already exists
8f9a13bfb118: Layer already exists
0aeceb7c293d: Layer already exists
c81d4fdb67fc: Layer already exists
0e82d78b3eal: Layer already exists
301c1bb42cc0: Layer already exists
latest: digest: sha256:43aa442039126a5e35a6272f1cdf904b24221fddc035874a251e37747db7224d size: 2423
```

5. Next step is to create kluster in Google Kluster Engine (GKE):

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A/Final Project/backend (master)
$ gcloud container clusters create event-cluster --zone us-central1 --num-nodes 1 --disk-type pd-standard
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See
https://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for ways to check usage and for migration
instructions.
Note: Your Pod address range ('--cluster-ip4-cidr') can accommodate at most 1008 node(s).
Creating cluster event-cluster in us-central1... Cluster is being health-checked (Kubernetes Control Plane is healthy)...done.
Created [https://container.googleapis.com/v1/projects/cloud-app-dev-amen/zones/us-central1/clusters/event-cluster].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/_gcloud/us-central1/event-cluster?project=cloud-app-dev-amen
kubeconfig entry generated for event-cluster.
NAME          LOCATION    MASTER VERSION  MASTER IP          MACHINE TYPE  NODE VERSION  NUM_NODES  STATUS
event-cluster  us-central1  1.30.5-gke.1699000  34.122.116.203    e2-medium    1.30.5-gke.1699000  3          RUNNING
```

6. Connecting to **event-cluster** cluster:

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A/Final Project/backend (master)
$ gcloud container clusters get-credentials event-cluster
Fetching cluster endpoint and auth data.
kubeconfig entry generated for event-cluster.
```

7. Next, create deployment.yaml with deployment instructions for Kubernetes:

```
Final Project > backend > deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  ~ metadata:
4    name: event-app
5  ~ spec:
6    replicas: 1
7    selector:
8      ~ matchLabels:
9        app: event-app
10   ~ template:
11     ~ metadata:
12       ~ labels:
13         app: event-app
14     ~ spec:
15       containers:
16       - name: event-app
17         image: gcr.io/cloud-app-dev-amen/event-app:latest
18         ports:
19         - containerPort: 8080
20       resources:
21       ~ requests:
22         cpu: "100m"
23         memory: "100Mi"
24       ~ limits:
25         cpu: "400m"
26         memory: "400Mi"
```

8. Deploying the application using deployment.yaml:

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU
r)
$ kubectl apply -f deployment.yaml
deployment.apps/event-app created
```

9. And finally, check deployed app:

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU
r)
$ kubectl apply -f deployment.yaml
deployment.apps/event-app created
```

<input type="checkbox"/> Status	Name ↑	Location	Tier ?	Number of nodes	Total vCPUs	Total memory
<input checked="" type="checkbox"/>	event-cluster	us-central1	Standard	3	6	12 GB

- **Google Cloud Endpoints:**

Step by step implementation of Google Cloud Endpoints:

1. The first step is create [openapi.yaml](#) configuration file in the project directory:

```
Final Project > backend > app > openapi.yaml
1 openapi: 3.0.0
2 info:
3   title: Event Management API
4   description: API for managing users, events, notifications, and registrations.
5   version: 1.0.0
6 servers:
7   - url: https://your-api-url.com/api
8     description: Production server
9 paths:
10  /auth/register:
11    post:
12      summary: Register a new user
13      requestBody:
14        content:
15          application/json:
16            schema:
17              type: object
18              required:
19                - email
20                - password
21                - username
22                - first_name
23                - last_name
24              properties:
25                email:
26                  type: string
27                  format: email
28                password:
29                  type: string
30                username:
31                  type: string
32                first_name:
33                  type: string
```

This image does not contain all [openapi.yaml](#). As you can see on image, in this file we need to write version of swagger, info about your API, host and schemes, after that you write Endpoints and describe them including method type, parameters, responses and security.

2. **Important to now.** The host name must in this format:
“[YOUR_PROJECT_ID].appsspot.com”, otherwise it doesn’t work. Below official documentation:

- ```
API_NAME.endpoints.YOUR_PROJECT_ID.cloud.goog
```

- YOUR\_PROJECT\_ID*  .appspot.com

## gcloud endpoints services deploy openapi.yaml

```
azikkw@DESKTOP-T1GF8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A/Final Project/backend (master)
$ gcloud endpoints services deploy openapi.yaml
Waiting for async operation operations/serviceConfigs.cloud-app-dev-amen.appspot.com:08181e37-8352-4ff1-bac2-8a344b1359ae to complete...
Operation finished successfully. The following command can describe the Operation details:
 gcloud endpoints operations describe operations/serviceConfigs.cloud-app-dev-amen.appspot.com:08181e37-8352-4ff1-bac2-8a344b1359ae
Waiting for async operation operations/rollouts.cloud-app-dev-amen.appspot.com:db921e14-0a18-4725-bf6b-1defdc3a681 to complete...
Operation finished successfully. The following command can describe the Operation details:
 gcloud endpoints operations describe operations/rollouts.cloud-app-dev-amen.appspot.com:db921e14-0a18-4725-bf6b-1defdc3a681
Service Configuration [2024-12-13r0] uploaded for service [cloud-app-dev-amen.appspot.com]

To manage your API, go to: https://console.cloud.google.com/endpoints/api/cloud-app-dev-amen.appspot.com/overview?project=cloud-app-dev-amen

Updates are available for some Google Cloud CLI components. To install them, please run:
 $ gcloud components update
```

←

Event Management API

SHOW PERMISSIONS PANEL

OVERVIEW

QUOTAS

DEPLOYMENT HISTORY

▲ No data is available for the selected time frame.

600ms

400ms

200ms

0

8:20

8:25

8:30

8:35

8:40

8:45

8:50

8:55

9 PM

9:05

9:10

9:15

Select columns

Requests, 4xx, 5xx, Latency 99%, Latency 95%, Latency median, Avg resp size, an...

| Method                    | Requests ↓ | 4xx | 5xx | Latency 99% | Latency 95% | Latency median | Avg resp size | Avg req size | Links                     |
|---------------------------|------------|-----|-----|-------------|-------------|----------------|---------------|--------------|---------------------------|
| POST /auth/register       | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| GET /events               | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| POST /events              | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| GET /events/{event_id}    | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| DELETE /events/{event_id} | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| PUT /events/{event_id}    | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| GET /notifications        | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| GET /registrations        | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| DELETE /registrations     | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |
| POST /registrations       | 0          | 0   | 0   | -           | -           | -              | -             | -            | <a href="#">View logs</a> |

Rows per page: 10 ▼

1 – 10 of 12

◀

▶


1. Methods list:

- **POST:** Create user (Register), Create event, Create registration to event, Create review
- **GET:** Get user, Get event/events, Get my-event/my-events, Get registrations, Get reviews, Get notifications

- **PUT:** Update event
- **DELETE:** Delete events, Delete registrations

## 2. Endpoints URL:

- **REGISTER endpoint:**

**POST**  <https://cloud-app-dev-amen.uc.r.appspot.com/auth/register>

### Register

Username

First Name

Last Name

Email

Password

**Register**

Already have an account? [Log In](#)

Functionality: This endpoint creates User and add it to database.

- **LOGIN:**

## Log In

Email

Enter your email

Password

Enter your password

Log In

Already have an account? [Register](#)

- **GET Profile endpoint:**

GET

https://cloud-app-dev-amen.uc.r.appspot.com/profile/:user\_id

## Profile

**Username:** azikkw

**First Name:** Azat

**Last Name:** Amen

**Email:** azat.amenov@gmail.com

Log out

Functionality: Returns current user data by user\_id

- **GET Events endpoint:**

GET

https://cloud-app-dev-amen.uc.r.appspot.com/events

## Events

|                              |                              |         |
|------------------------------|------------------------------|---------|
| A                            |                              |         |
| a                            | In Almaty   2024-12-13T14:16 | 1000 tg |
| Kadbrakhmanov event          |                              |         |
| description for kabdrs event | In Almaty   2024-12-18T18:00 | 5000 tg |

Functionality: Returns all events with creator\_id that does not match with current user\_id

- **GET Event by event\_id endpoint:**

GET

https://cloud-app-dev-amen.uc.r.appspot.com/event/:id

### Kadbrakhmanov event

Description: description for kabdrs event

Date and time: 2024-12-18T18:00

Location: Almaty Capacity: 100

Price: 5000

Created by: kabdr

Register to event

#### Reviews

azikkw 13.12.2024

★★★★★

5 out of 5

Functionality: Returns selected event page by event\_id

- **CREATE Event endpoint:**

POST

https://cloud-app-dev-amen.uc.r.appspot.com/events

### Create event

Title

New

Description

New description

Date and time

13.12.2024 14:00

Location

Almaty

Capacity

30

Price

10000

Create event

New

New description

In Almaty | 2024-12-13T14:00

Functionality: This endpoint creates new event

- **UPDATE Event by event\_id endpoint:**

PUT

https://cloud-app-dev-amen.uc.r.appspot.com/event/:id

Edit Event

Title:

First Event

Description:

This is testing of first event

Date and Time:

10.12.2024 18:00

Location:

Almaty

Capacity

50

Price

20000

Save

Cancel

Functionality: This endpoint updates an event data by event\_id

- **DELETE Event by event\_id endpoint:**

DELETE

https://cloud-app-dev-amen.uc.r.appspot.com/event/:id

First Event

Description: This is testing of first event

Date and time: 2024-12-10T18:00

Location: Almaty Capacity: 50

Price: 20000 tg

Edit

Delete

Functionality: This endpoint deletes event by event\_id

- **GET Reviews endpoint:**

GET

https://cloud-app-dev-amen.uc.r.appspot.com/reviews/

Kadbrakhmanov event

Description: description for kabdrs event

Date and time: 2024-12-18T18:00

Location: Almaty Capacity: 100

Price: 5000

Created by: kabdr

Register to event

Reviews

azikkw 13.12.2024

★★★★★

5 out 5


myessim 13.12.2024

★★★★★

The best event!!!


Functionality: Just returns all reviews for an event by event\_id

- **ADD Review to event endpoint:**

**POST**  <https://cloud-app-dev-amen.uc.r.appspot.com/reviews/>

**Add your review**

Оценка:

4


Comment:

good

Leave review

Functionality: Adds review to event by event\_id

- **GET Registrations endpoint:**

**GET**  <https://cloud-app-dev-amen.uc.r.appspot.com/registrations/>

**Registrations**

38ddb6c6-9f7a-4074-9a93-d3ff0e91e9bc

Registration date is 2024-12-13 04:17:21

Kadbrakhmanov event

description for kabdrs event


In Almaty | 2024-12-18T18:00

Price is 5000 tg

Delete

Functionality: Returns all user registrations user\_id

- **DELETE Registration endpoint:**

**DELETE**  [https://cloud-app-dev-amen.uc.r.appspot.com/registrations/:user\\_id](https://cloud-app-dev-amen.uc.r.appspot.com/registrations/:user_id)

38ddb6c6-9f7a-4074-9a93-d3ff0e91e9bc

Registration date is 2024-12-13 04:17:21

Kadbrakhmanov event

description for kabdrs event


In Almaty | 2024-12-18T18:00

Price is 5000 tg

Delete

Functionality: Deletes registration to event by registration\_id

- **GET Notifications endpoint:**

**GET**  [https://cloud-app-dev-amen.uc.r.appspot.com/notifications/:user\\_id](https://cloud-app-dev-amen.uc.r.appspot.com/notifications/:user_id)

**Notifications**

352a0615-2298-4720-bfa6-8914834ff881

User Ob1LXxY5MAXFI2ReN5nPwBeXbXv1 registered to your event ce4d2270-2bc7-4df4-b06e-190949ffa3c1

2024-12-13 04:14:27

5a2a0297-0af1-4051-aa59-4f475a16c7d3

User Ob1LXxY5MAXFI2ReN5nPwBeXbXv1 registered to your event 22cb287b-7189-4bfd-bd2a-b906eeff32fe

2024-12-13 02:54:51

5e1d2edf-95fd-4875-9924-fd04d2818373

User Dld7chaoTZPidmU1MiEwWgji71C2 registered to your event 854be538-0313-464f-81ae-0f2b4b9ac73a at 2024-12-13 02:19:35

Functionality: Returns all notifications of user by user\_id field



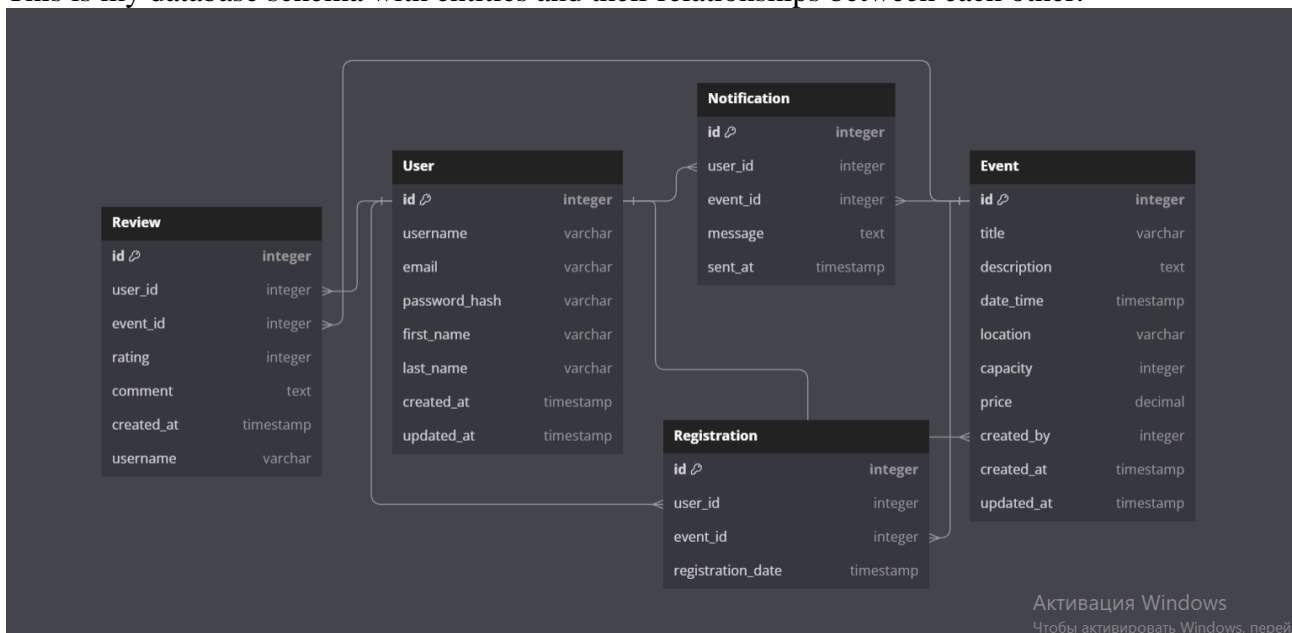
This was full overview of my API with methods, endpoints, and UI visualizations.

## 8. Database Design and Optimization

For my project, I selected Firebase. It is NoSQL database developed by Google. I chose Firebase for several reasons:

1. Easy implementation and integration (for example, it already has predefined methods for searching for matches, filtering, sorting).
2. Monitoring tools for tracking user actions.
3. Also, access to data from Firebase can be obtained very quickly.
4. Very nice and user-friendly interface.

This is my database schema with entities and their relationships between each other:

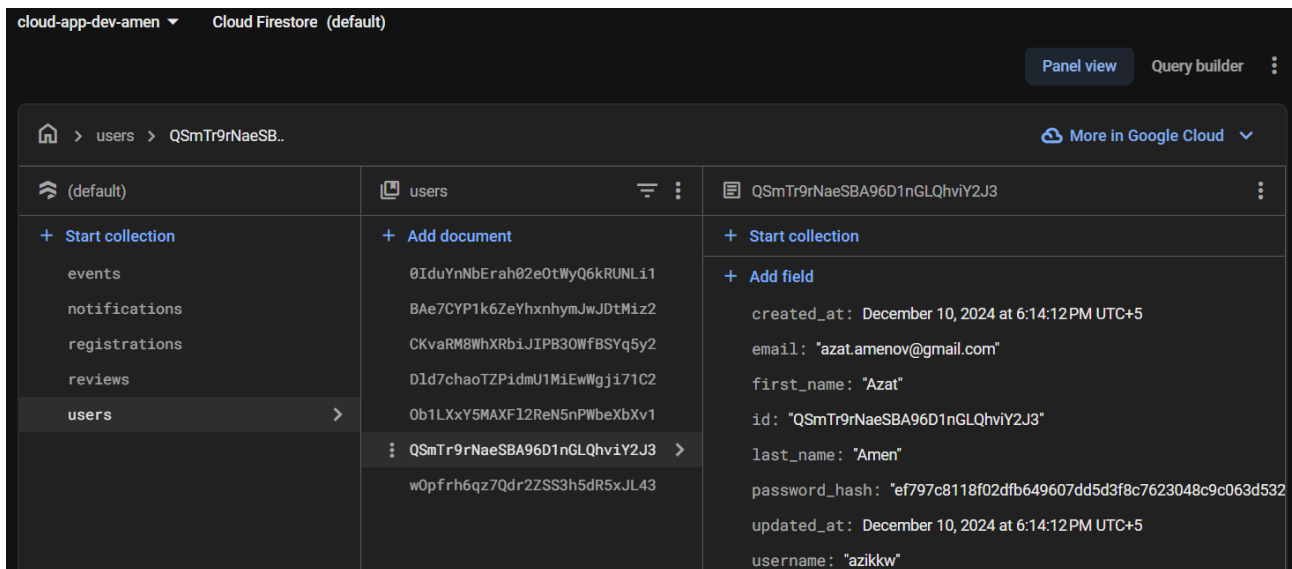


The main one is **User**; nobody can exist without User. Because only user can create events and manage them, leave reviews and register to events. Also, if there is no user, then notifications will have nowhere to go.

**Event** also very important, because all system built around events. User can create and manage events, leave review to event and register to event.

**Review** and **Registration** is related to event parts because they are really part of event. Users leave review to event and register to event.

And **Notifications**, its also important part of my app, because notifications sending when user do something in the system. They are sent to the logs and to the users, for example when User events got reviews and when someone register to User events.



This is how looks Firebase Firestore database where I store all my data.

## 9. Event-Driven Architecture

Event-Driven architecture implementation using Google Cloud Pub/Sub:

1. Firstly, to start to work with Pub/Sub I installed **google-cloud-pubsub** library.

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester
r)
$ pip install google-cloud-pubsub
```

2. After that, in the next step, I created topics that will publish Publishers.

| Topics                                                               |                |                                                                |           |                  | SHOW INFO PANEL | LEARN |
|----------------------------------------------------------------------|----------------|----------------------------------------------------------------|-----------|------------------|-----------------|-------|
| LIST METRICS                                                         |                |                                                                |           |                  |                 |       |
| Filter Filter topics                                                 |                |                                                                |           |                  |                 |       |
| <input type="checkbox"/> Topic ID ↑                                  | Encryption key | Topic name                                                     | Retention | Ingestion source |                 |       |
| <input type="checkbox"/> <a href="#">change-event-capacity-topic</a> | Google-managed | projects/cloud-app-dev-amen/topics/change-event-capacity-topic | —         | —                | ⋮               |       |
| <input type="checkbox"/> <a href="#">notification-topic</a>          | Google-managed | projects/cloud-app-dev-amen/topics/notification-topic          | —         | —                | ⋮               |       |
| <input type="checkbox"/> <a href="#">user-notifications-topic</a>    | Google-managed | projects/cloud-app-dev-amen/topics/user-notifications-topic    | —         | —                | ⋮               |       |
| <input type="checkbox"/> <a href="#">users-registration-topic</a>    | Google-managed | projects/cloud-app-dev-amen/topics/users-registration-topic    | —         | —                | ⋮               |       |

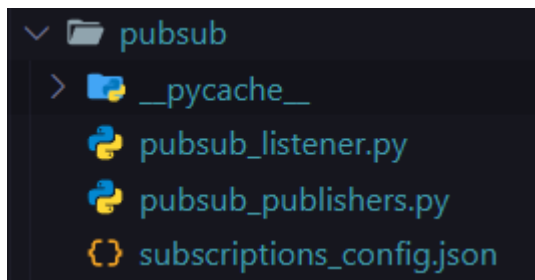
3. And the subscriptions for all topics was generated automatically:

| Subscriptions                       |                                                 |               |                            |              |           |                  |                       |            | SHOW INFO PANEL | LEARN |
|-------------------------------------|-------------------------------------------------|---------------|----------------------------|--------------|-----------|------------------|-----------------------|------------|-----------------|-------|
| LIST METRICS                        |                                                 |               |                            |              |           |                  |                       |            |                 |       |
| Filter Filter subscriptions         |                                                 |               |                            |              |           |                  |                       |            |                 |       |
| <input type="checkbox"/> State      | Subscription ID ↑                               | Delivery type | Topic name                 | Ack deadline | Retention | Message ordering | Exactly once delivery | Expiration |                 |       |
| <input checked="" type="checkbox"/> | <a href="#">change-event-capacity-topic-sub</a> | Push          | projects/cloud-app-dev-... | 10 seconds   | 7 days    | Disabled         | Disabled              | 31 days    | ⋮               | ▼     |
| <input checked="" type="checkbox"/> | <a href="#">notification-topic-sub</a>          | Push          | projects/cloud-app-dev-... | 10 seconds   | 7 days    | Disabled         | Disabled              | 31 days    | ⋮               | ▼     |
| <input checked="" type="checkbox"/> | <a href="#">user-notifications-topic-sub</a>    | Push          | projects/cloud-app-dev-... | 10 seconds   | 7 days    | Disabled         | Disabled              | 31 days    | ⋮               | ▼     |
| <input checked="" type="checkbox"/> | <a href="#">users-registration-topic-sub</a>    | Push          | projects/cloud-app-dev-... | 10 seconds   | 7 days    | Disabled         | Disabled              | 31 days    | ⋮               | ▼     |

4. To ensure that subscribers will work in future I changed their **pushConfigs**. For example, for all did the following, but only names of Subs changed:

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A/Final Project/backend (master)
$ gcloud pubsub subscriptions update change-event-capacity-topic-sub --push-endpoint=https://cloud-app-dev-amen.uc.r.appspot.com/pubsub/change-event-capacity-topic-sub
Updated subscription [projects/cloud-app-dev-amen/subscriptions/change-event-capacity-topic-sub].
```

5. In the next step, I started to creating functions for publishers and listeners (Subscribers), I created folder for them:



6. And the next I wrote my publishers:

- **publish\_user\_data:** When user register, it publishes topic with User data
- **publish\_notifications:** At each user actions it publishes topic with Notification
- **publish\_user\_notifications:** When some users leave review or register to event, it sends Notification to the event owner
- **publish\_changing\_event\_capacity:** When user register to the event, the capacity value decreases to 1, and when user delete his/her registration its increases to 1

That is all about the Pubs in my system.

```
Final Project > backend > app > utils > pubsub > pubsub_publishers.py > publish_user_data
1 from google.cloud import pubsub_v1
2 import json
3
4 project_id = 'cloud-app-dev-amen'
5
6
7 # Registration publisher
8 publisher1 = pubsub_v1.PublisherClient()
9 topic_name1 = 'users-registration-topic'
10
11 def publish_user_data(user_data):
12 topic_path = publisher1.topic_path(project_id, topic_name1)
13 message = json.dumps(user_data).encode('utf-8')
14
15 future = publisher1.publish(topic_path, message)
16 future.result()
17
18
19 # System notifications
20 publisher2 = pubsub_v1.PublisherClient()
21 topic_name2 = 'notification-topic'
22
23 def publish_notification(notification_data):
24 topic_path = publisher2.topic_path(project_id, topic_name2)
25 message = json.dumps(notification_data).encode('utf-8')
26
27 future = publisher2.publish(topic_path, message)
28 future.result()
29
30
31 # User notifications
32 publisher3 = pubsub_v1.PublisherClient()
33 topic_name3 = 'user-notifications-topic'
34
35 def publish_user_notifications(notification_data):
36 topic_path = publisher3.topic_path(project_id, topic_name3)
37 message = json.dumps(notification_data).encode('utf-8')
38
39 future = publisher3.publish(topic_path, message)
```

7. In this step I wrote listeners (Subscribers) that will handle published topics:

```
def process_message(subscription_name, data):
 try:
 if subscription_name == 'users-registration-topic-sub':
 db.collection('users').document(data['id']).set(data)
 print(f"User {data['username']} added to Firestore.")
 elif subscription_name == 'notification-topic-sub':
 requests.post(
 SEND_NOTIFICATION_URL,
 headers={'Content-Type': 'application/json'},
 json={"notification": data}
)
 print(f"Notification sent: {data['message']}")
 elif subscription_name == 'user-notifications-topic-sub':
 db.collection('notifications').document(data['id']).set(data)
 print("Notification successfully added.")
 elif subscription_name == 'change-event-capacity-topic-sub':
 event_ref = db.collection('events').document(data['event_id'])
 event = event_ref.get()
 event_data = event.to_dict()
 capacity = event_data['capacity']
 if data['changing'] == 'dec':
 updated_data = {"capacity": capacity - 1}
 event_ref.update(updated_data)
 else:
 updated_data = {"capacity": capacity + 1}
 event_ref.update(updated_data)
 print("Event capacity changed.")
 except Exception as e:
 print(f"Error processing message for {subscription_name}: {str(e)}")
```

Here I wrote one function that process all topics and do some actions based on the name of the published topic.

```
@pubsub_bp.route('/<subscription_name>', methods=['POST'])
def handle_pubsub_message(subscription_name):
 try:
 envelope = request.get_json()
 if not envelope or 'message' not in envelope:
 return jsonify({"error": "Invalid Pub/Sub message format"}), 400

 message = envelope['message']
 data = base64.b64decode(message['data']).decode('utf-8')
 data_json = json.loads(data)

 process_message(subscription_name, data_json)

 return jsonify({"status": "Message processed"}), 200
 except Exception as e:
 return jsonify({"error": str(e)}), 500
```

Here function handle all published topics and give them to the messages processing function.

Cloud functions for events handling is a good idea that can be realized. Cloud functions execute in answer to some actions. They give an opportunity to write and deploy small event driven functions that will executed as response to some actions. But I think that Pub/Sub will be better solution for this situation. Because them was developed to solve problems like that. And when we have ready solution with documentation, it will be faster and easier to solve some kind of problems.

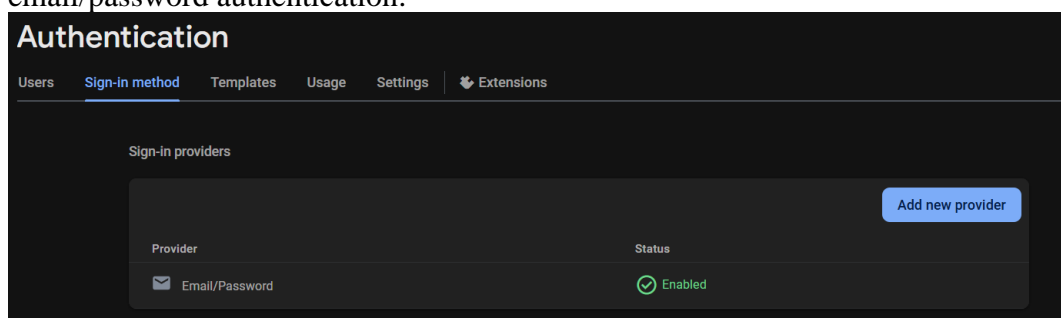
## 10. Machine Learning Integration

I would use AI to, for example, analyze user feedback on their opinions, or, for example, to make personalized recommendations of events based on their interests, or to represent the number of participants or some purpose of the chatbot for consultation. And I could improve the ideas that I have and make them effective. For input and output to my system, I would use Google Cloud AI Platform or Vertex AI.

## 11. Security Measures

I implemented security practices to my application. Here is explanation of them below:

1. Authentication provided by Firebase. It offers various methods of users authentication such as email/password, phone/password, google account and etc. For my app I selected email/password authentication:



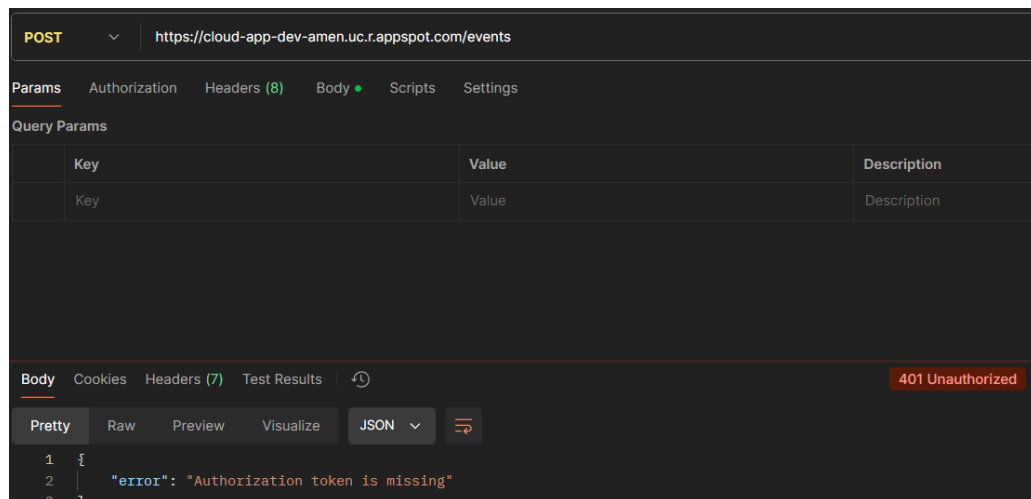
2. After login for user generates JWT token that is need in all server operations to enusere user authentication. And at each request to server in client side sends JWT token to approving:

```
const URL :string = 'https://cloud-app-dev-amen.uc.r.appspot.com'

export const fetchEvents = async () => { Show usages
 const token :string = getToken();
 const id :string = localStorage.getItem(key: "uid");

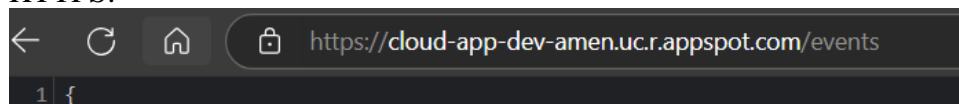
 try {
 const response :Response = await fetch(input: `${URL}/events`, init: {
 method: 'GET',
 headers: {
 'Content-Type': 'application/json',
 'uid': id,
 'Authorization': `Bearer ${token}`,
 },
 },
);
}
```

As an example, I will show example to enusre that my application secure. If you try access server endpoints without token it returns following:

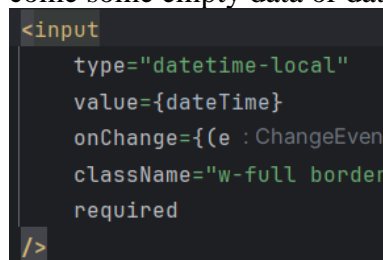


So, how you can it returns response with status **401 Unauthorized**. It means that it works well.

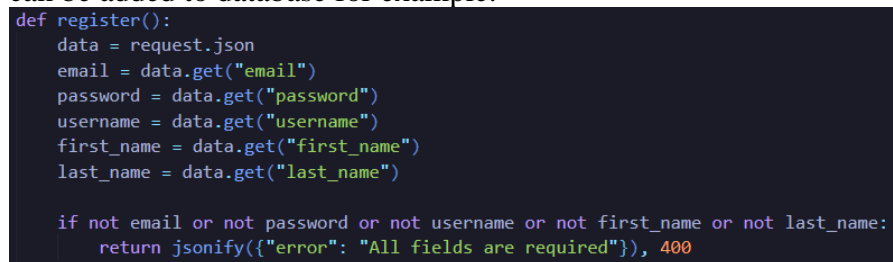
### 3. HTTPS:



- Also I added user data validation on client side. So, that ensure that to server never will come some empty data or data with errors.



But there is also some validations on the server. It servers as a last check to verify that data can be added to database for example:



- And finally, I guess that without monitoring tools any security procedures will be weak. So, I created Cloud Function that send notification to logs at each user action. It gives an opportunity to track user actions in the system and respond to any incidents in a timely manner.

| send_notification Cloud Run function (Deployed at Dec 13, 2024, 5:29:34 AM) URL: <a href="https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification">https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification</a> |                             |                                                                                                                                                                                         | View in Cloud Run |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING                                                                                                                                                                                      |                             |                                                                                                                                                                                         |                   |
| Logs Severity Default Filter Search all fields and values                                                                                                                                                                                              |                             |                                                                                                                                                                                         |                   |
| SEVERITY                                                                                                                                                                                                                                               | TIMESTAMP                   | SUMMARY                                                                                                                                                                                 |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 09:57:59.419 NPT | Notification ba81d9ac-4b96-4e16-aa9d-47164f35c67c. User w0pfrh6q70d72533h5d85xL43 leaved review to event: 27bed749-55c2-4285-bbbe-3afabef32d44 with rating: 4 at 2024-12-13 03:29:24    |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 09:59:28.277 NPT | POST 200 295 B 3 ms python-requests/2.32.3 https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification                                                                  |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 09:59:28.286 NPT | Notification a32842c7-5fb2-48f3-8148-f18e9ea99e62. User 0b1LxY5MAXF128n5nPWbeXbXv1 registered to event: ce4d2270-2bc7-4df4-b06e-190949ffa3c1 at 2024-12-13 04:14:27                     |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 10:02:21.246 NPT | POST 200 295 B 3 ms python-requests/2.32.3 https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification                                                                  |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 10:02:21.254 NPT | Notification 3ec12ec-4a96-424c-9ea7-cbdea9a9585f. User Q5mTr9rNaeSBA96D1nGLQhviY2J3 registered to event: 27bed749-55c2-4285-bbbe-3afabef32d44 at 2024-12-13 04:17:21                    |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 10:02:37.141 NPT | POST 200 313 B 3 ms python-requests/2.32.3 https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification                                                                  |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 10:02:37.145 NPT | Notification d537ae6f-ef86-48b2-b274-df3b86b77873. User Q5mTr9rNaeSBA96D1nGLQhviY2J3 leaved review to event: 27bed749-55c2-4285-bbbe-3afabef32d44 with rating: 5 at 2024-12-13 04:17:37 |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 12:37:04.798 NPT | POST 200 295 B 1.5 s python-requests/2.32.3 https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification                                                                 |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 12:37:06.459 NPT | Default STARTUP TCP probe succeeded after 1 attempt for container "worker" on port 8080.                                                                                                |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 12:37:06.541 NPT | Notification 0877a568-581e-4a45-b877-0596787867a2. User Q5mTr9rNaeSBA96D1nGLQhviY2J3 registered to event: 30a98a9b-a87b-4e6c-9649-236a61f8f50b at 2024-12-13 06:52:04                   |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 12:45:02.992 NPT | POST 200 294 B 1.1 s python-requests/2.32.3 https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification                                                                 |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 12:45:04.153 NPT | Default STARTUP TCP probe succeeded after 1 attempt for container "worker" on port 8080.                                                                                                |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 12:45:04.189 NPT | Notification c1e251be-1b88-4d74-ad4b-8912b8b3e83. User Q5mTr9rNaeSBA96D1nGLQhviY2J3 created event: e24f3c53-8c8c-4521-909d-df839316c6ab. New at 2024-12-13 07:00:02                     |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 19:26:17.272 NPT | POST 200 313 B 838 ms python-requests/2.32.3 https://us-central1-cloud-app-dev-amen.cloudfunctions.net/send_notification                                                                |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 19:26:18.272 NPT | Default STARTUP TCP probe succeeded after 1 attempt for container "worker" on port 8080.                                                                                                |                   |
| >                                                                                                                                                                                                                                                      | 2024-12-13 19:26:18.322 NPT | Notification 7274bc8b-48d2-48d6-bfbc-3a839152d347. User 0b1LxY5MAXF128n5nPWbeXbXv1 leaved review to event: 854be538-0313-464f-81ae-0f2b4b9ac73a with rating: 4 at 2024-12-13 13:41:16   |                   |

## 12. Scalability and Performance

**Scalability** is the ability of a system to handle more work or support more users when needed. It is important because it helps applications run smoothly even when there are more requests or higher demand.

What is **Horizontal Autoscaling**? It means adding more machines or instances to share the workload. It is good for applications that do not keep data on the instance (stateless), so any instance can handle any request.

### Horizontal Autoscaling Implementation:

To do this, we create an `hpa.yaml` file that contains the HorizontalPodAutoscaler manifest. HPA will receive the metrics of the processor used from the "metrics server", and if our condition is met, kubernetes will start adding the number of replicas of our module.

1. Create `hpa.yaml` file:

```
Final Project > backend > hpa.yaml
1 apiVersion: autoscaling/v2
2 kind: HorizontalPodAutoscaler
3 metadata:
4 name: event-app-hpa
5 spec:
6 scaleTargetRef:
7 apiVersion: apps/v1
8 kind: Deployment
9 name: event-app
10 minReplicas: 2
11 maxReplicas: 10
12 metrics:
13 - type: Resource
14 resource:
15 name: cpu
16 target:
17 type: Utilization
18 averageUtilization: 50
```

2. Next we apply our `hpa.yaml` to implement HPA.

```
azikkw@DESKTOP-T1GF8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Applicat
r)
$ kubectl apply -f hpa.yaml
horizontalpodautoscaler.autoscaling/event-app-hpa created
```

To test the work of the HPA, I will increase the artificial load on my To-do application, and we will observe the changes of horizontalPodAutoScaler live using `kubectl get hpa -[YOUR HPA] --watch` command:

```
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development I Serek A (master)
$ kubectl get hpa event-app-hpa --watch
```

| NAME          | REFERENCE            | TARGETS       | MINPODS | MAXPODS | REPLICAS | AGE |
|---------------|----------------------|---------------|---------|---------|----------|-----|
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 2        | 34m |
| event-app-hpa | Deployment/event-app | cpu: 2%/50%   | 2       | 10      | 2        | 38m |
| event-app-hpa | Deployment/event-app | cpu: 2%/50%   | 2       | 10      | 2        | 39m |
| event-app-hpa | Deployment/event-app | cpu: 31%/50%  | 2       | 10      | 2        | 40m |
| event-app-hpa | Deployment/event-app | cpu: 200%/50% | 2       | 10      | 2        | 41m |
| event-app-hpa | Deployment/event-app | cpu: 200%/50% | 2       | 10      | 4        | 41m |
| event-app-hpa | Deployment/event-app | cpu: 200%/50% | 2       | 10      | 8        | 41m |
| event-app-hpa | Deployment/event-app | cpu: 61%/50%  | 2       | 10      | 8        | 42m |
| event-app-hpa | Deployment/event-app | cpu: 24%/50%  | 2       | 10      | 8        | 42m |
| event-app-hpa | Deployment/event-app | cpu: 13%/50%  | 2       | 10      | 8        | 43m |
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 8        | 43m |
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 8        | 47m |
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 4        | 47m |
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 4        | 48m |
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 3        | 48m |
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 3        | 48m |
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 2        | 48m |
| event-app-hpa | Deployment/event-app | cpu: 1%/50%   | 2       | 10      | 2        | 49m |

So, this is HPA work image:

- When it just started to work and CPU is less than 50%, it uses minimum replicas number.
- When CPU is more than 50% and the load on application is increasing, the number of replicas is also starting to grow according to the load.
- When I lowered the load on application and it starts consuming less and less CPU, the number of replicas also starts to decrease accordingly.

**Load balancer** distributes incoming requests between pods to ensure an even distribution of all resources and high availability of the application.

### Load balancer implementaion:

1. First step to create Load Balancer is define service.yaml with service (LoadBalancer type):

```
Final Project > backend > service.yaml
```

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4 name: event-service
5 spec:
6 type: LoadBalancer
7 selector:
8 app: event-app
9 ports:
10 - port: 80
11 targetPort: 8080
```

2. And then, implement **service.yaml** using `kubectl apply -f service.yaml` command and check results using `kubectl get services`. And how you can see on image LoadBalancer successfully integrated to the kubernetes:



```

azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development
r)
$ kubectl apply -f service.yaml
service/event-service created

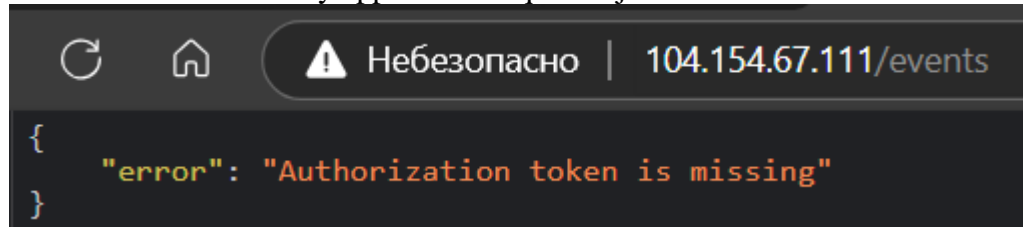
azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development
r)
$ kubectl get services
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
event-service LoadBalancer 34.118.227.105 <pending> 80:31126/TCP 33s
kubernetes ClusterIP 34.118.224.1 <none> 443/TCP 6h11m

azikkw@DESKTOP-T1GP8V3 MINGW64 /e/Study I KBTU/Semester VII/Cloud Application Development
r)
$ kubectl get services
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
event-service LoadBalancer 34.118.227.105 104.154.67.111 80:31126/TCP 96s
kubernetes ClusterIP 34.118.224.1 <none> 443/TCP 6h12m

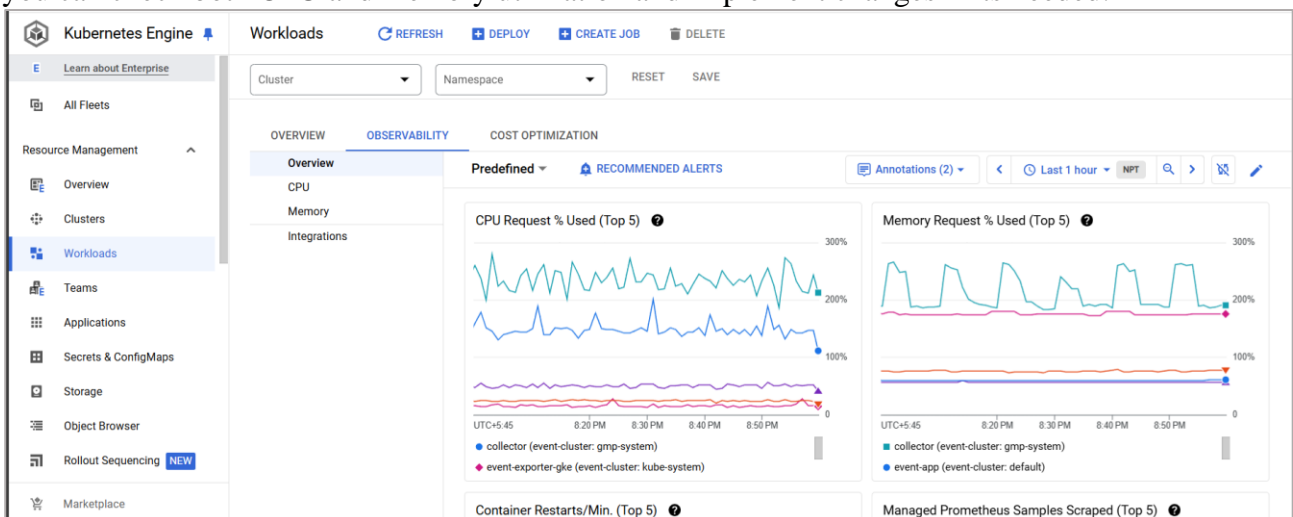
```

| Name ↑                        | Type          | Endpoints                         |
|-------------------------------|---------------|-----------------------------------|
| <a href="#">event-service</a> | Load balancer | <a href="#">104.154.67.111:80</a> |

- And finally how you can see the EXTERNAL-IP works, and can be access (On the image is unauthorized because my application requirese jwt token for authentication user).



About **performance** is that Kubernetes Engine enable Cloud Monitoring tools for containers. There you can check both CPU and memory utilization and implement changes if its needed:



## 13. Challenges and Solutions

The problems I have encountered:

- The first problem was to choose which database to use. Because each of them has its own benefits, it was hard to choose one. But finally I selected Fireabse, and think that it was good decision.
- Also I thought how to implement authentication, and when I found Firebase with its Authentication my problem solved momentarily.

3. It was also difficult to figure out how to organize the system so that it would be readable and scalable in the future.
4. Another problem was that it took me a long time to deploy my API to Google Cloud Endpoints, but the solution was very simple. It was necessary to read the documentation. As soon as I tried to implement something from there, it immediately worked properly. The problem was in the name of the host
5. And besides, the very implementation of an event-driven architecture was not an easy task at first. It was a little confusing to understand how everything was going on. But as soon as I started writing and trying it out in real practice, things immediately improved.
6. Implementing the client and server side at once was a bit difficult, but I liked it. There were some difficulties in coming up with the process of their interaction, but in the end everything turned out fine.

## 14. Conclusion

In conclusion, first of all, I would like to say that it was a very interesting experience. I was developing an event-driven cloud system for the first time. As a result, I have an event-driven cloud based application, where Users can create an account, create and manage events, as well as register for other people's events and leave feedback. And all this is accompanied by a pleasant user interface.

I have integrated cloud functions to send notifications to logs, which ensures constant monitoring of user activity. I also put my application in a container and deployed it in GKE. In addition, I created an API in Cloud Endpoints and added authentication via JWT to my application. I used the Firestore cloud database to store user data, and also used tools to ensure smooth operation and reliability.

During the development of the project, I gained knowledge in event-driven architecture for myself. I also realized the importance of being able to read and understand documentation.

In the future, I will probably improve and increase the functionality of the application itself. I have also added more tools to improve monitoring. I would increase the number of cloud functions and expand the event-driven system.

## 15. References

- [1] Google Cloud Documentation. (2024). *Python 3 Runtime Environment*. Retrieved from: <https://cloud.google.com/appengine/docs/standard/python3/runtime>
- [2] Google Cloud Documentation. (2024). *Create a Cloud Run function by using the Google Cloud CLI*. Retrieved from: <https://cloud.google.com/functions/docs/create-deploy-gcloud#functions-clone-sample-repository-python>
- [3] Google Cloud Documentation. (2024). *Google Kubernetes Engine (GKE) Documentation*. Retrieved from: <https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview>
- [4] Kubernetes Documentation. (2024). *Autoscaling Workloads*. Retrieved from: <https://kubernetes.io/docs/concepts/workloads/autoscaling/>
- [5] Kubernetes Documentation. (2024). *HorizontalPodAutoscaler Walkthrough*. Retrieved from: <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>

[6] Google Cloud Documentation. (2024). *Getting started with Cloud Endpoints for the App Engine flexible environment with ESP*. Retrieved from:

<https://cloud.google.com/endpoints/docs/openapi/get-started-app-engine>

[7] Google Cloud Documentation. (2024). *Pub/Sub Documentation*. Retrieved from:

<https://cloud.google.com/pubsub/docs>

## 16. Appendices

I have described my every step in great detail for each of the sections, so there is no need for this.

But I want to provide link to project code with all recourses and also link to diagrams:

- **GitHub:** <https://github.com/azikkw/Cloud-AppDevelopment-2024/tree/master/Final%20Project>
- **Database Diagram:** <https://dbdiagram.io/d/Event-Management-System-67568cbbe9daa85aca14f4ce>