Chirag Kasat
https://chiragkasat.com/
+91 9503376977

Auth flow for single sign-on using refresh and access tokens

OVERVIEW

There are two server instances and a PostgreSQL as well as a Redis database instance, all dockerized. The **auth-server** implements the authentication workflow. The other **test-server** consists of just a single route to test the authorization of a restricted route by another server other than the main auth-server. I have tried including two of the backend projects in this single project as they blended well. The authentication workflow is explained in the Auth-Flow pdf attached.

RUNNING THE PROJECT

The first would be having a .env file to have some environment variables set. I have attached a sample env-local file so we can use that. Then we fire up all the services using the docker-compose up command and we should have all things set up and running.

Commands:

1. **cp ./env-example .env**
2. **docker-compose up --build**

You can run the docker command in detached mode using a '-d' flag but in this way, you wouldn't be able to see all the logs.

**It may happen that PostgreSQL service hasn't booted up as it takes time to load for first time, so the auth-server might exit as it cannot connect to the database. In that case we can go to another terminal instance and restart the auth-server service using

```
docker restart <container_id>
```

The docker id can be found by running the following

docker ps -a

Then find the auth-server instance and copy the id and run the command above.

Now, the auth REST API should be available on port 3000 and the test API on port 4000.

TESTING

You may use any software for API testing (Postman, Insomnia, Thunder Client)

The base URL for auth-server will be - **http://localhost:3000/**

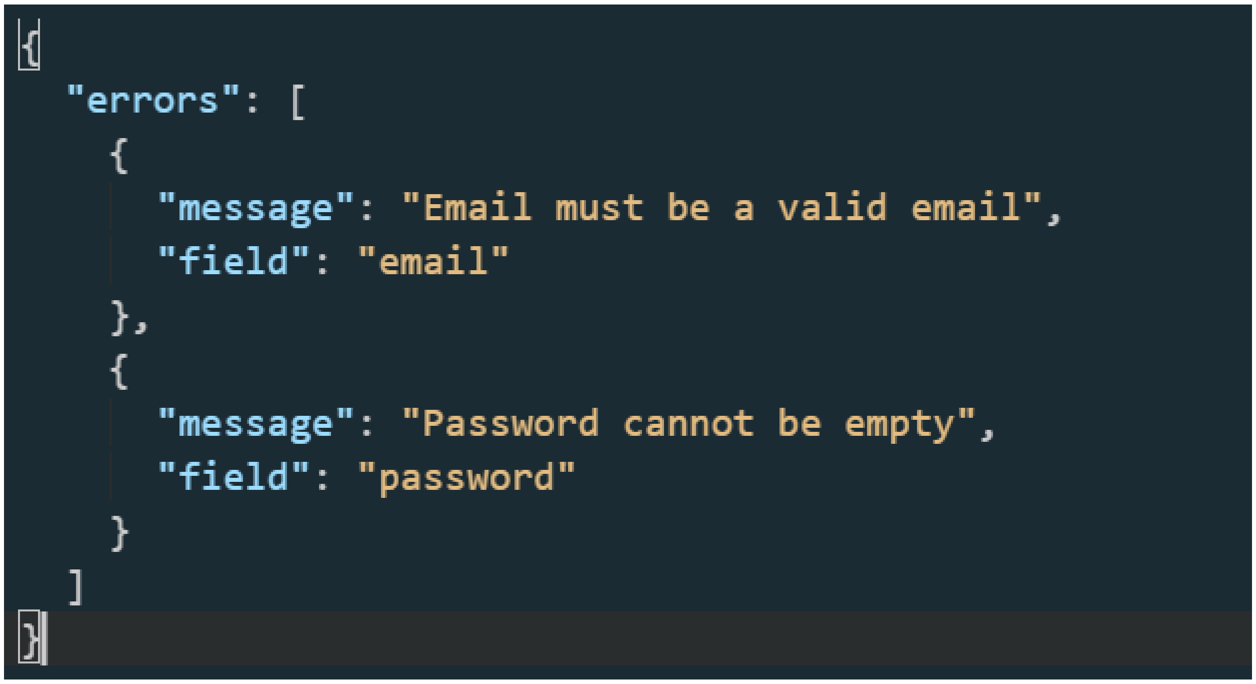
The base URL for test-server will be - **http://localhost:4000/**

The auth-server endpoints:

METHOD	ENDPOINT	USE
POST	/api/signup	User signup using email and password
POST	/api/signin	User sign-in
POST	/api/signout	Remove refresh token session, logout
POST	/api/refresh	Endpoint for asking for a new access token using the refresh token

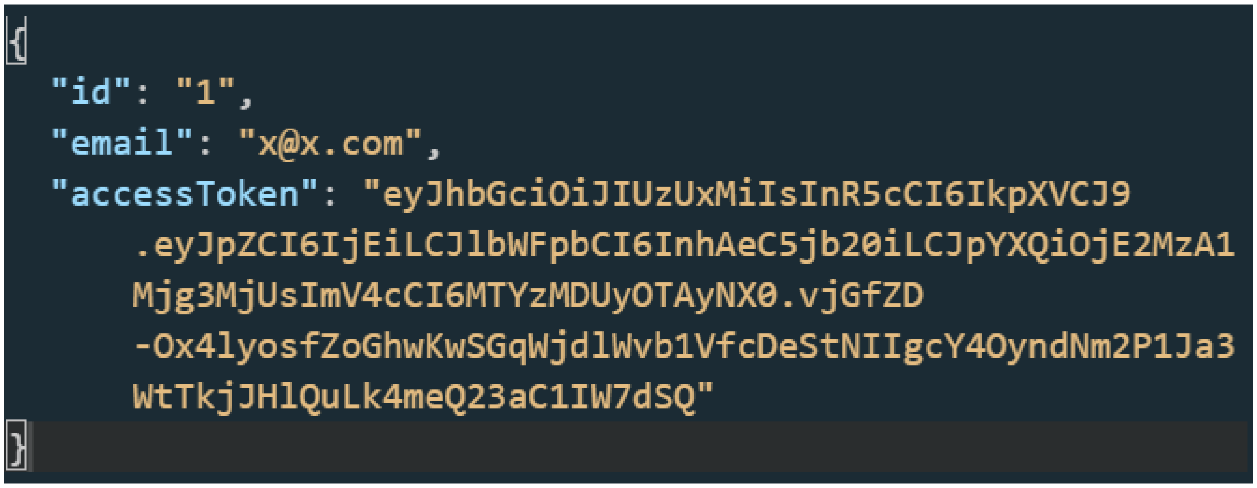
There's an additional /api/auth/user endpoint to check cookie-based access token.

- The first step will be registering the user.
  Using POST method **http://localhost:3000/api/signup**
  Include email and password in the body as JSON.
  This is the response when nothing is specified in body.
  The input is **validated** and **sanitized**.



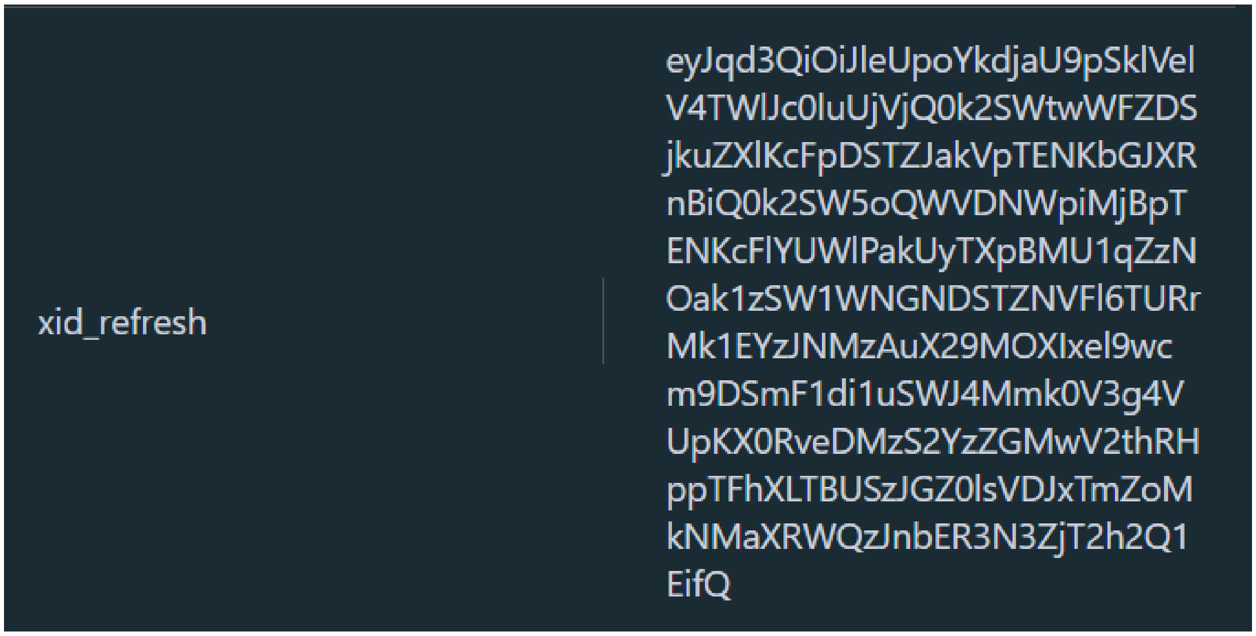
```
{
  "errors": [
    {
      "message": "Email must be a valid email",
      "field": "email"
    },
    {
      "message": "Password cannot be empty",
      "field": "password"
    }
  ]
}
```

- After providing the email and password there will be a response like this:



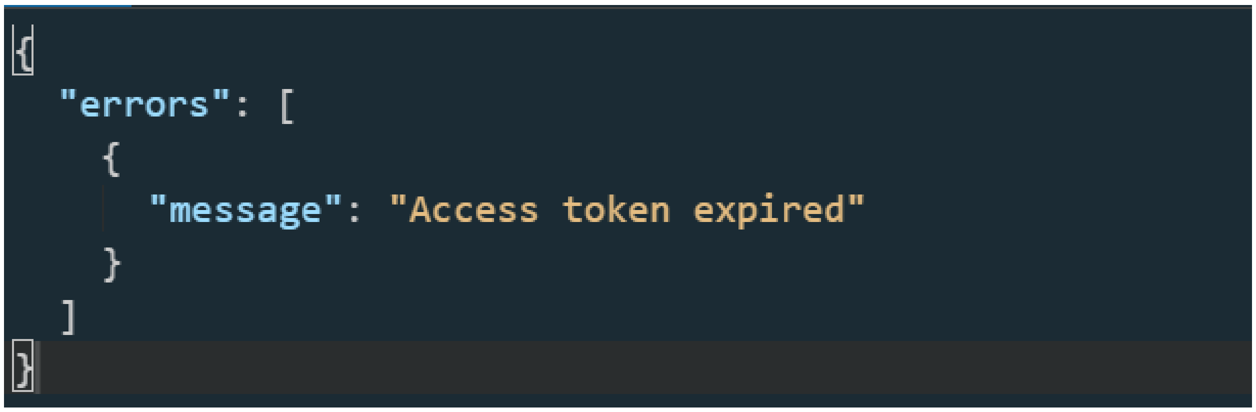
```
{
  "id": "1",
  "email": "x@x.com",
  "accessToken": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9
      .eyJpZCI6IjEiLCJlbWFpbCI6InhAeC5jb20iLCJpYXQiOjE2MzA1
      Mjg3MjUsImV4cCI6MTYzMDUyOTAyNX0.vjGfZD
      -Ox4lyosfZoGhwKwSGqWjdlWvb1VfcDeStNIIgcY4OyndNm2P1Ja3
      WtTkjJHlQuLk4meQ23aC1IW7dSQ"
}
```

- The refresh token is set as an **http-only cookie** named xid_refresh(mentioned in env file)



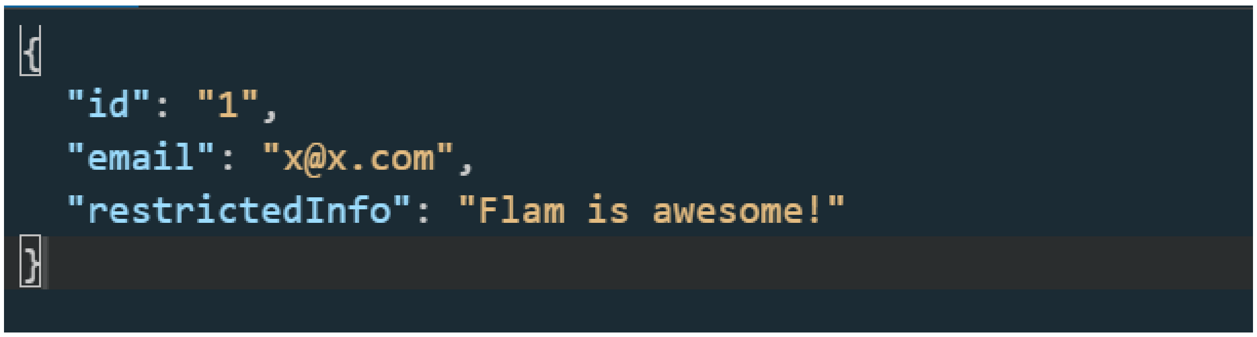
```
xid_refresh    eyJqd3QiOiJleUpoYkdjaU9pSklVel
               V4TWlJc0luUjVjQ0k2SWtwWFZDS
               jkuZXlKcFpDSTZJakVpTENKbGJXR
               nBiQ0k2SW5oQWVDNWpiMjBpT
               ENKcFlYUWlPakUyTXpBMU1qZzN
               Oak1zSW1WNGNDSTZNVFl6TURr
               Mk1EYzJNMzAuX29MOXlxel9wc
               m9DSmF1di1uSWJ4Mmk0V3g4V
               UpKX0RveDMzS2YzZGMwV2thRH
               ppTFhXLTBUSzJGZ0lsVDJxTmZoM
               kNMaXRWQzJnbER3N3ZjT2h2Q1
               EifQ
```

- Take the access token from above response and let's try to access the restricted endpoint of the test-server.
  Using GET method **http://localhost:4000/api/restricted**
- Include the access token in **Authorization header** in the format
  `Bearer <access_token>`
- If the token is not included or it has expired (expires after 5 mins, uses environment variable) the response will give a **401 status code**, and this response



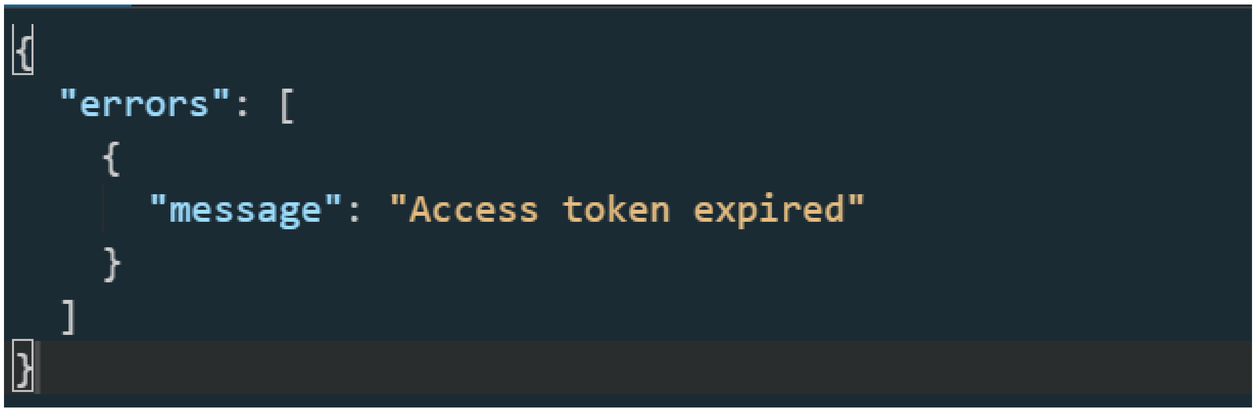
```
{
  "errors": [
    {
      "message": "Access token expired"
    }
  ]
}
```

- If the token is specified correctly, there should be this response.



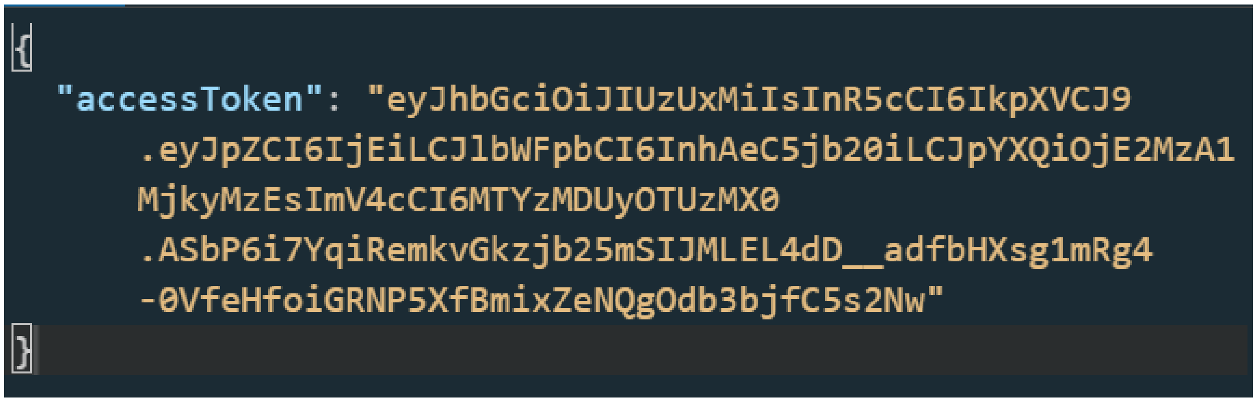
```
{
  "id": "1",
  "email": "x@x.com",
  "restrictedInfo": "Flam is awesome!"
}
```

- Once the token expires (5 mins), if you try to access this route again it will give the same error message with 401 status code.



```
{
  "errors": [
    {
      "message": "Access token expired"
    }
  ]
}
```

- This access token verification can be done by any service as required by just providing the secret key. After expiry though, we will again hit refresh endpoint of auth-server to get a new access token and use that for accessing test-server.
- Using POST method **http://localhost:3000/api/signup**
  If the refresh token hasn't expired(5 days) and it has a session in the redis database, itwill generate a new access token and send this as response, otherwise it will prompt to signin again.

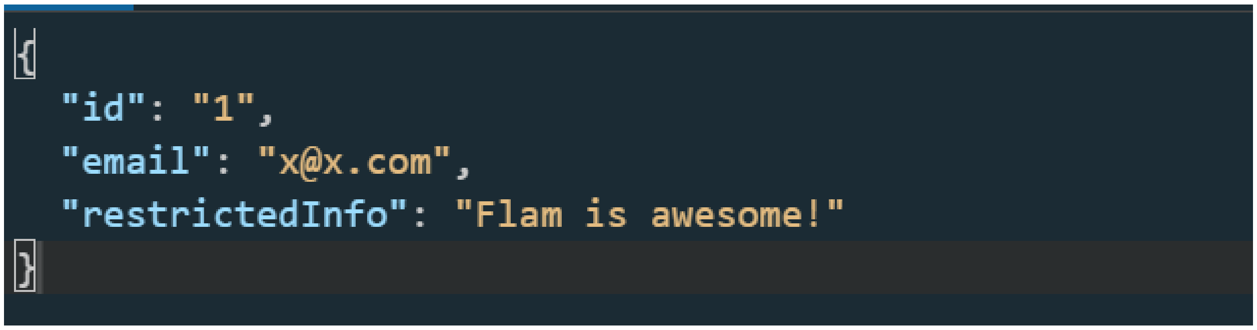


```
{
  "accessToken": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9
      .eyJpZCI6IjEiLCJlbWFpbCI6InhAeC5jb20iLCJpYXQiOjE2MzA1
      MjkyMzEsImV4cCI6MTYzMDUyOTUzMX0
      .ASbP6i7YqiRemkvGkzjb25mSIJMLEL4dD__adfbHXsg1mRg4
      -0VfeHfoiGRNP5XfBmixZeNQgOdb3bjfC5s2Nw"
}
```

- Now use this access token as Authorization header (Bearer <access_token>) in the restricted API of test-server

  Using GET method **http://localhost:4000/api/restricted**

  The response again would be



```
{
  "id": "1",
  "email": "x@x.com",
  "restrictedInfo": "Flam is awesome!"
}
```

CONCLUSION

This was an interesting project to work on and I thoroughly enjoyed learning about the best practices for authentication and authorization in a microservices based architecture. This flow manages to decrease some load on auth-server and still be secure enough. The other thing I wanted to do but didn't have enough time for was using Kafka to let the services to interact with each other using events and be separate from each other. I even applied for the frontend part but I don't think I would be able to complete it on time. This is my personal website and I have designed it myself, so you may have a look to judge my skills. I am excited for working with you and hopefully will be good enough for the position. Thank you.