End Report

1 CONTENTS

1 INSTRUCTIONS

1.1 OPTIONAL FEATURES

- Statical analysis is implemented through linting and formatting for Node.js-based services, including tests, service2 and api-gateway. Linting is implemented with eslint and formatting with Prettier.
- Tests and statical analysis are run in the CI/CD pipeline.

1.2 INSTRUCTIONS FOR TESTING

1.2.1 Instructions to start the application

- Ensure Docker and Docker Compose are installed on your system.
- Clone the repository and navigate to the root folder of the project.
- Build the services using the following command:
  - docker-compose build --no-cache
- Start services using:
  - docker-compose up

1.2.2 Instructions to test the application

- Using curl-commands:
  - Log in to the application:
    - curl http://localhost:8197 -u testuser:testpassword -H "Accept: text/plain"
  - Example of GET request:
    - curl http://localhost:8197/state -H "Accept: text/plain"
  - Example of PUT request:
    - curl http://localhost:8197/state -X PUT -d "PAUSED" -H "Content-Type: text/plain" -H "Accept: text/plain"
- Run tests cases:
  - Restart the container to clear application data:
    - docker-compose down && docker-compose up
  - Build and run tests in a container:
    - docker build -t tests-image ./tests --no-cache
    - docker run --rm tests-image
  - Alternatively navigate to the tests folder and execute tests with:
    - npm run test
- Use of browser:
  - After logging in via curl, access the website in a browser. Buttons are provided to make requests: GET /request and PUT /state to shutdown services.

1.2.3 Development platform information

- Hardware
  - CPU: AMD Ryzen 5 4600H with Radeon Graphics (3.00 GHz, 6 cores, 12 processors)
  - Memory: 24.0 GB DDR4 RAM, 3200 MHz
  - Storage: 476 GB SSD (SKHynix_HFM512GDHTNI-87A0B)

    - GPU: NVIDIA GeForce GTX 1660 Ti, AMD Radeon™ Graphics
- Software
    - CPU Architecture: x64-based PC
    - Operating System: Windows 11 Home 23H2
    - Docker Version: 27.3.1
    - Docker Compose Version: 2.29.7
    - Node Version (local testing, linting and formatting): 20.18.0

2 CI/CD PIPELINE

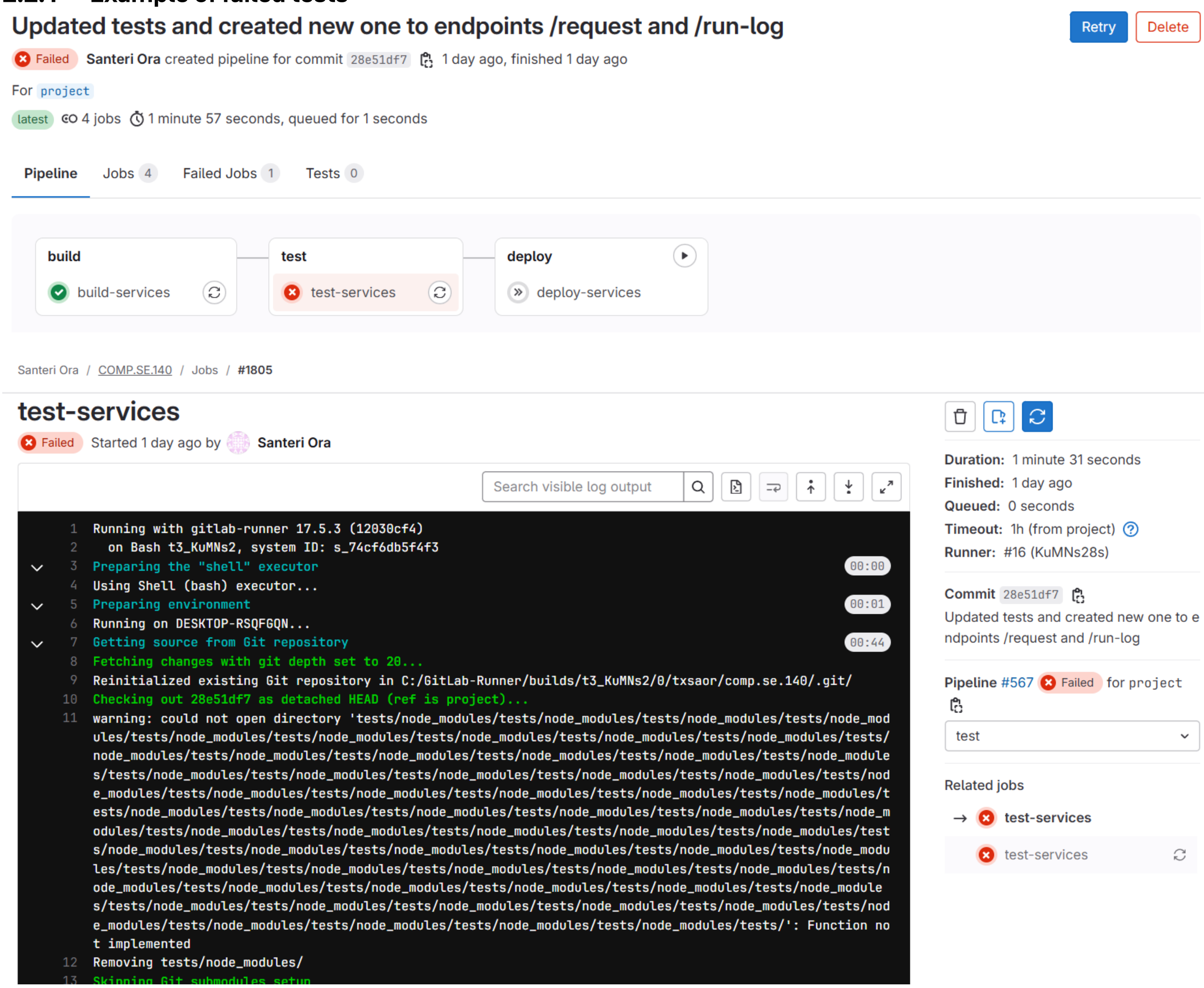
2.1 DESCRIPTION

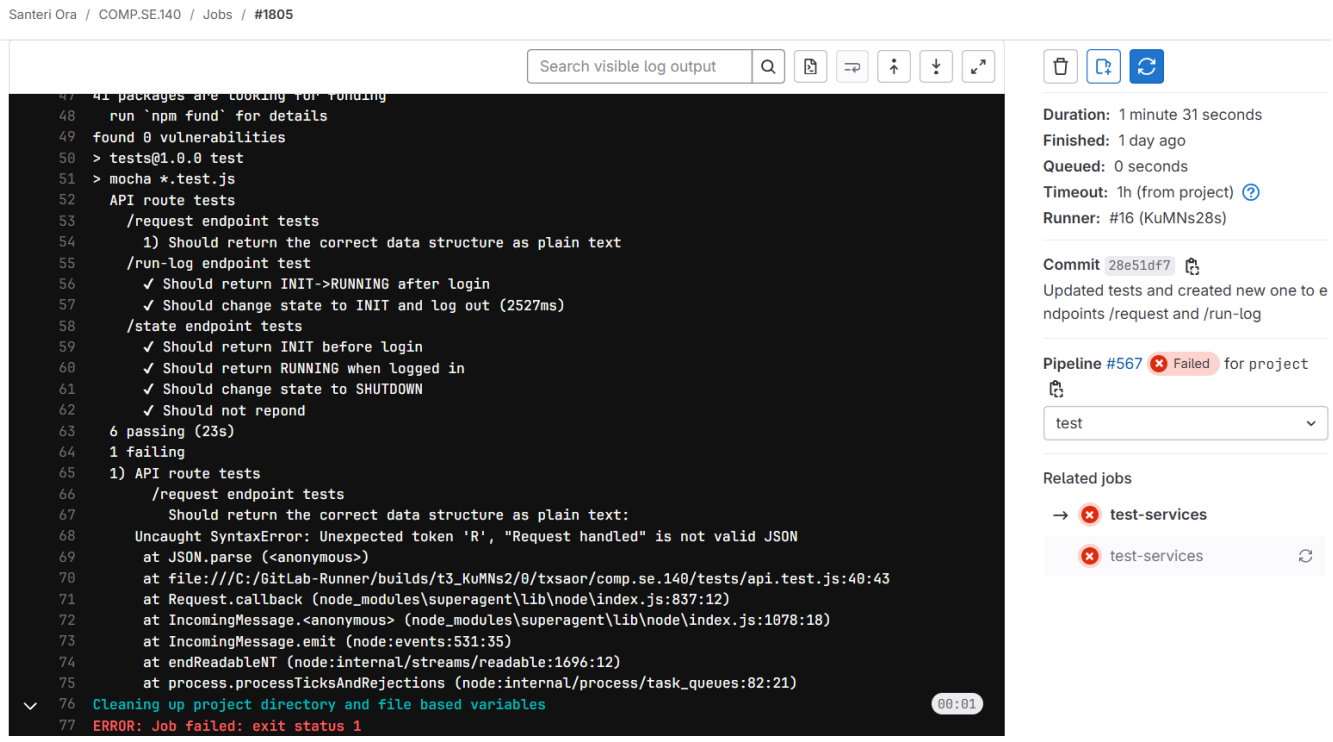
- Version management
    - Git is used for version control.
    - The “project” branch is dedicated to development and final deployment.
    - The pipeline triggers on every commit, with deployment restricted to commits made to the “project” branch.
    - Pipeline is implemented to GitLab CI/CD pipeline and run by GitLab runners.
- Static analysis
    - The pipeline starts with linting and formatting checks for Node based services tests, service2 and api-gateway with Eslint and Prettier.
    - This step ensures consistent code style and detects syntax errors.
- Build tools
    - Docker is used for containerizing services
    - Docker Compose orchestrates the multi-container setup.
- Testing
    - Frameworks: Mocha, Chai and Chai-http for unit testing.
    - Tests are executed in the pipeline in independent docker container outside of other services.
    - Test cases are divided into three sections based on the tested endpoint.
    - Test cases cover login, GET and PUT requests to /state, GET requests to /request and /runlog.
    - Tests are chained and all of them are run in one go. Because services state changes and logs and auth status are saved, If one of the tests fails, it can follow that some of the next tests can also fail even though they would work correctly with the correct service state.
    - The last test of testing SHUTDOWN will shutdown all of the services.
- Packing
    - Services are packed using:
        - docker-compose build --no-cache
- Deployment
    - Automated deployment in “project” branch via Docker Compose using:
        - docker-compose up -d
- Monitoring

- Test logs are accessible in the GitLab pipeline. No additional monitoring tools are implemented.

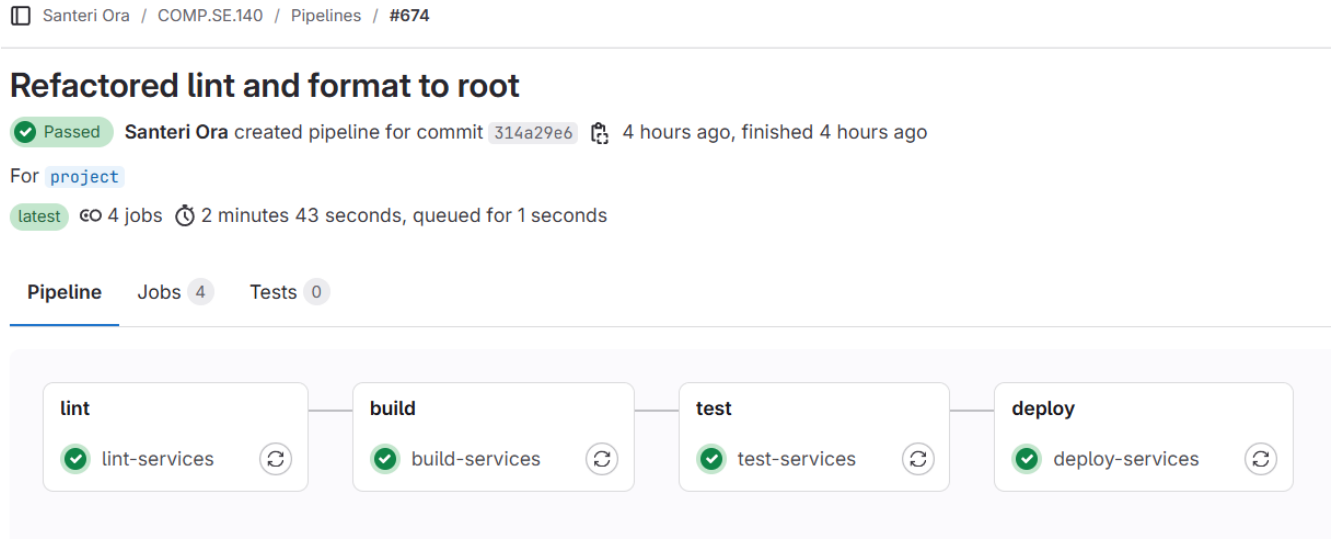
2.2 Example runs of the pipeline

2.2.1 Example of failed tests





2.2.2 Example of passed tests



test-services

Passed Started 4 hours ago by **Santeri Ora**

Search visible log output

```
Running with gitlab-runner 17.5.3 (12030cf4)
  on Bash t3_KuMNs2, system ID: s_74cf6db5f4f3
Preparing the "shell" executor                                    00:00
Using Shell (bash) executor...
Preparing environment                                             00:01
Running on DESKTOP-RSQF6QN...
Getting source from Git repository                                00:03
Fetching changes with git depth set to 20...
Reinitialized existing Git repository in C:/GitLab-Runner/builds/t3_KuMNs2/0/txsaor/comp.se.140/.git/
Checking out 314a29e6 as detached HEAD (ref is project)...
Skipping Git submodules setup
Executing "step_script" stage of the job script                   00:55
$ docker-compose up -d
 Container compse140-nginx-1  Stopping
 Container compse140-service1-2  Stopping
 Container compse140-service1-3  Stopping
 Container compse140-service1-1  Stopping
 Container compse140-service1-1  Stopped
 Container compse140-nginx-1  Stopped
 Container compse140-api-gateway-1  Recreate
 Container compse140-service1-2  Stopped
 Container compse140-service1-3  Stopped
 Container compse140-service2-1  Recreate
 Container compse140-service2-1  Recreated
 Container compse140-nginx-1  Stopping
```

Duration: 1 minute 1 second
Finished: 4 hours ago
Queued: 2 seconds
Timeout: 1h (from project)
Runner: #16 (KuMNs28s)

Commit 314a29e6
Refactored lint and format to root

Pipeline #674 Passed for project

test

Related jobs

→ test-services

Santeri Ora / COMP.SE.140 / Jobs / #2141

Search visible log output

```
 Container compse140-nginx-1  Stopped
 Container compse140-service1-3  Recreate
 Container compse140-nginx-1  Stopping
 Container compse140-nginx-1  Stopped
 Container compse140-nginx-1  Stopping
 Container compse140-service1-1  Recreate
 Container compse140-nginx-1  Stopped
 Container compse140-service1-2  Recreate
 Container compse140-api-gateway-1  Recreated
 Container compse140-service1-2  Recreated
 Container compse140-service1-3  Recreated
 Container compse140-service1-1  Recreated
 Container compse140-nginx-1  Recreate
 Container compse140-nginx-1  Recreated
 Container compse140-service2-1  Starting
 Container compse140-api-gateway-1  Starting
 Container compse140-service2-1  Started
 Container compse140-api-gateway-1  Started
 Container compse140-service1-2  Starting
 Container compse140-service1-2  Started
 Container compse140-service1-1  Starting
 Container compse140-service1-1  Started
 Container compse140-service1-3  Starting
 Container compse140-service1-3  Started
 Container compse140-nginx-1  Starting
 Container compse140-nginx-1  Started
$ docker build -t tests-image ./tests --no-cache
#0 building with "desktop-linux" instance using docker driver
#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 187B done
```

Duration: 1 minute 1 second
Finished: 4 hours ago
Queued: 2 seconds
Timeout: 1h (from project)
Runner: #16 (KuMNs28s)

Commit 314a29e6
Refactored lint and format to root

Pipeline #674 Passed for project

test

Related jobs

→ test-services

Search visible log output

```
#1 DONE 0.0s
#2 [internal] load metadata for docker.io/library/node:20.18.0
#2 DONE 0.5s
#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s
#4 [1/4] FROM docker.io/library/node:20.18.0@sha256:a7a3b7ec6de4b11bb2d673b31de9d28c6da09c557ee65453672c8e4f754c23fc
#4 DONE 0.0s
#5 [2/4] WORKDIR /tests
#5 CACHED
#6 [internal] load build context
#6 transferring context: 60.44kB 0.5s done
#6 DONE 0.6s
#7 [3/4] COPY . .
#7 DONE 0.6s
#8 [4/4] RUN npm install
#8 6.684 npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you wan
t a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
#8 6.882 npm warn deprecated glob@8.1.0: Glob versions prior to v9 are no longer supported
#8 8.062
#8 8.062 added 119 packages, and audited 120 packages in 7s
#8 8.062
#8 8.062 41 packages are looking for funding
#8 8.062   run `npm fund` for details
#8 8.068
#8 8.068 found 0 vulnerabilities
#8 8.069 npm notice
#8 8.069 npm notice New minor version of npm available! 10.8.2 -> 10.9.1
#8 8.069 npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.9.1
#8 8.069 npm notice To update run: npm install -g npm@10.9.1
```

Duration: 1 minute 1 second
Finished: 4 hours ago
Queued: 2 seconds
Timeout: 1h (from project)
Runner: #16 (KuMNs28s)

Commit 314a29e6
Refactored lint and format to root

Pipeline #674 Passed for project

test

Related jobs

→ test-services

Santeri Ora / COMP.SE.140 / Jobs / #2141

Search visible log output

```
#8 8.069 npm notice
#8 DONE 8.3s
#9 exporting to image
#9 exporting layers
#9 exporting layers 0.5s done
#9 writing image sha256:dc300564134707882O9a7497e968012e67ab082dfafd9ae1586e096f51d61037 done
#9 naming to docker.io/library/tests-image 0.0s done
#9 DONE 0.5s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/tqblylna05zfd26nyn4vd7w5q
$ sleep 10
$ docker run --rm tests-image
> tests@1.0.0 test
> mocha *.test.js
  API route tests
    /request endpoint tests
      ✓ Should return the correct data structure as plain text (2752ms)
    /run-log endpoint test
      ✓ Should return INIT->RUNNING after login
      ✓ Should change state to INIT and log out (2520ms)
    /state endpoint tests
      ✓ Should return INIT before login
      ✓ Should return RUNNING when logged in
      ✓ Should change state to SHUTDOWN
      ✓ Should not respond
  7 passing (23s)
$ docker-compose down
 Container compse140-nginx-1  Stopping
 Container compse140-nginx-1  Stopped
 Container compse140-nginx-1  Removing
 Container compse140-nginx-1  Removed
```

Duration: 1 minute 1 second
Finished: 4 hours ago
Queued: 2 seconds
Timeout: 1h (from project)
Runner: #16 (KuMNs28s)

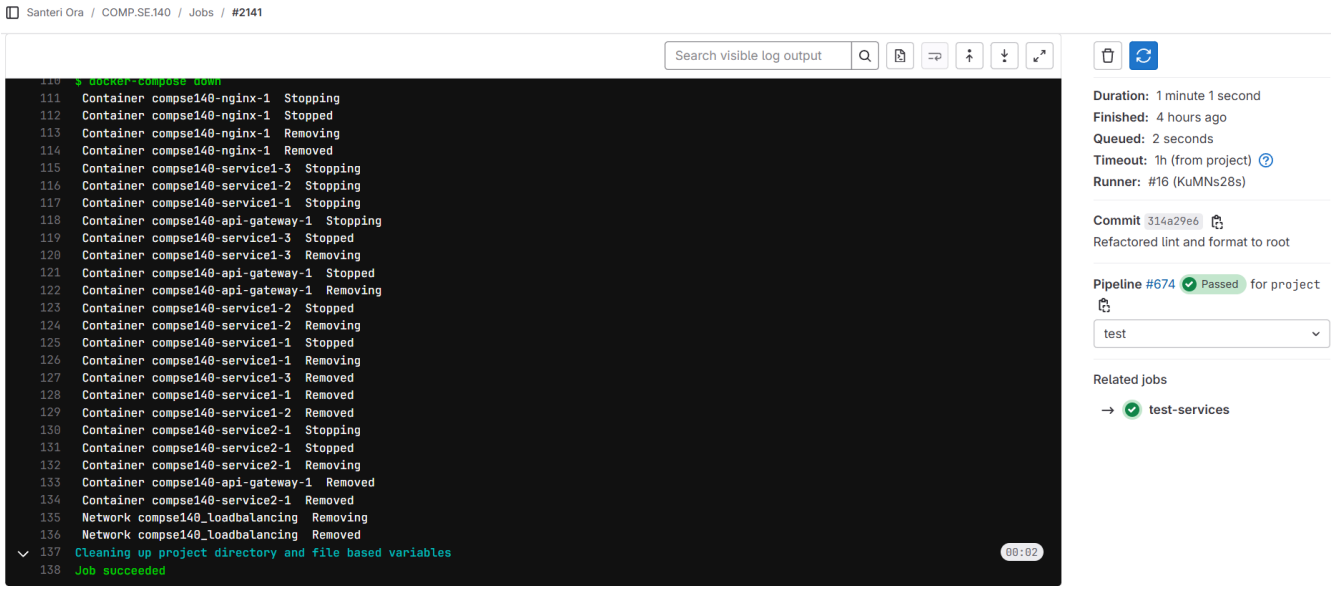
Commit 314a29e6
Refactored lint and format to root

Pipeline #674 Passed for project

test

Related jobs

→ test-services



3 REFLECTION

3.1 MAIN LEARNINGS AND WORST DIFFICULTIES

I have had experience with Docker and using multiple services earlier. Although the final project and earlier exercises provided some new experiences with Docker, especially with more advanced usage like using entrypoints and creating replicas of services. The learnings that I gained were more about the new technologies used in the project, such as NGINX and the GitLab CI/CD pipeline with runners. Most of the challenges also came from these areas. By learning from the internet and experimenting, I created authentication, rate limiting, and state handling in NGINX. In earlier exercises, the NGINX implementation was very simple, but for the project, I had to rethink the whole structure and learn even more about NGINX and its capabilities.

Most of the challenges came from small details and implementing new things that I had never done before. There weren't any major issues, but I encountered some blockers where I got stuck and spent most of the project time. The first issue was shutting down the services inside Docker. I managed to handle that with entrypoints, and it worked well on my computer and in the GitLab pipeline when running tests. I received feedback from the NGINX exercise that the usage of the whole service required removing the entrypoint, and after that, shutting down the services didn't work. For this final project, I couldn't find the reason for that because the implementation works without any problems in Docker on my computer and also in GitLab runners. I also containerized the tests to get detached results.

Especially for this final project, most of the difficulties came from saving the authentication state and the services state to NGINX. I had to redo the authentication logic and switch the whole NGINX image to allow the use of Lua blocks. After all, I got NGINX working as intended, but I am still not very satisfied with the implementation. The logic is pretty messy, and due to time constraints, I couldn't improve it to the best of my ability.

I also encountered some issues running GitLab runners on my computer, but I solved that after one day. Overall, these worst difficulties that I mentioned took up 80% of the whole time I used for the project. All other tasks were very simple or I was already familiar with them.

3.2 THINGS I WOULD DO DIFFERENTLY

Things that I would do differently would be testing and error handling. I would perform more comprehensive testing to ensure there are no bugs and unwanted behaviors in NGINX because it was a very new technology for me, and as I already mentioned, I am not fully satisfied with the implementation. The other thing I would do differently is more advanced error handling. For the earlier exercises, I tried to handle errors well and show correct messages to users. With the final project I realized that proper and more detailed error handling would take too long, so I just focused more on positive usage and correct responses of the services.

3.3 AMOUNT EFFORT USED

50 hours

3.4 USAGE OF AI

- GitHub Copilot (ChatGPT) was used for syntax fixes. I also attempted to solve advanced errors and seek assistance, but most of the responses were incorrect and ended up being a waste of time.
- ChatGPT was used to review the End Report's English language and provide corrections.