COMP.SE.140 Project End Report

Allan Li

1 Instructions for the teaching assistant

1.1 Implemented optional features

Implemented static analysis step in the pipeline. The tools used include ESLint for JavaScript linting and Ruff for Python linting.

1.2 Instructions for examiner to test the system

The system can be initialized and tested locally by running `***docker compose up --build***` in the terminal.

2 Description of the CI/CD pipeline

2.1 Version management

This project can be found in the "project" branch of the repository. The branch utilizes a three stage CI/CD pipeline which builds, tests, and deploys the project services. The pipeline kicks in automatically when changes are pushed to the branch.

2.2 Build tools

The build step includes linting and code formatting. For JavaScript services, ESLint is used for linting and Prettier used for code formatting. For the Python service, Ruff was chosen for both linting and code formatting.

2.3 Testing: tools and test cases

Testing for the project mostly involved writing positive and negative tests for various API endpoints. The testing for the project was conducted using Vitest and Supertest for both

the API-gateway service and monitor service. Test-driven development practices were utilized for facilitating the implementation of the API-gateway functionalities.

2.3.1 API-gateway test cases

```
API-gateway listening on port 8083

 ✓ src/index.test.js (7)
   ✓ PUT /state (5)
     ✓ should return a 200 status code if state is INIT
     ✓ should return a 200 status code if state is RUNNING
     ✓ should return a 200 status code if state is PAUSED
     ✓ should return a 200 status code if state is SHUTDOWN
     ✓ should return a 400 status code if state is valid
   ✓ GET /state (1)
     ✓ should return a 200 status code with the current state
   ✓ GET /run-log (1)
     ✓ should return a 200 status code with the run logs

 Test Files  1 passed (1)
      Tests  7 passed (7)
   Start at  21:12:57
   Duration  485ms (transform 27ms, setup 0ms, collect 258ms, tests 31ms,
environment 0ms, prepare 75ms)
```

Figure 1 - API-gateway test cases

2.3.2 Monitor test cases

```
Monitor listening on port 8087

 ✓ src/index.test.js (3)
   ✓ GET / (3)
     ✓ should return a 200 status code
     ✓ send an empty response if no messages are received
     ✓ send a response with messages in new lines if they are received

 Test Files  1 passed (1)
      Tests  3 passed (3)
   Start at  21:24:32
   Duration  395ms (transform 18ms, setup 0ms, collect 192ms, tests 10ms,
 environment 0ms, prepare 55ms)
```

Figure 2 - Monitor test cases

3 Example runs of the pipeline

3.1 Failing tests

tests

Failed Started 1 day ago by allan_li

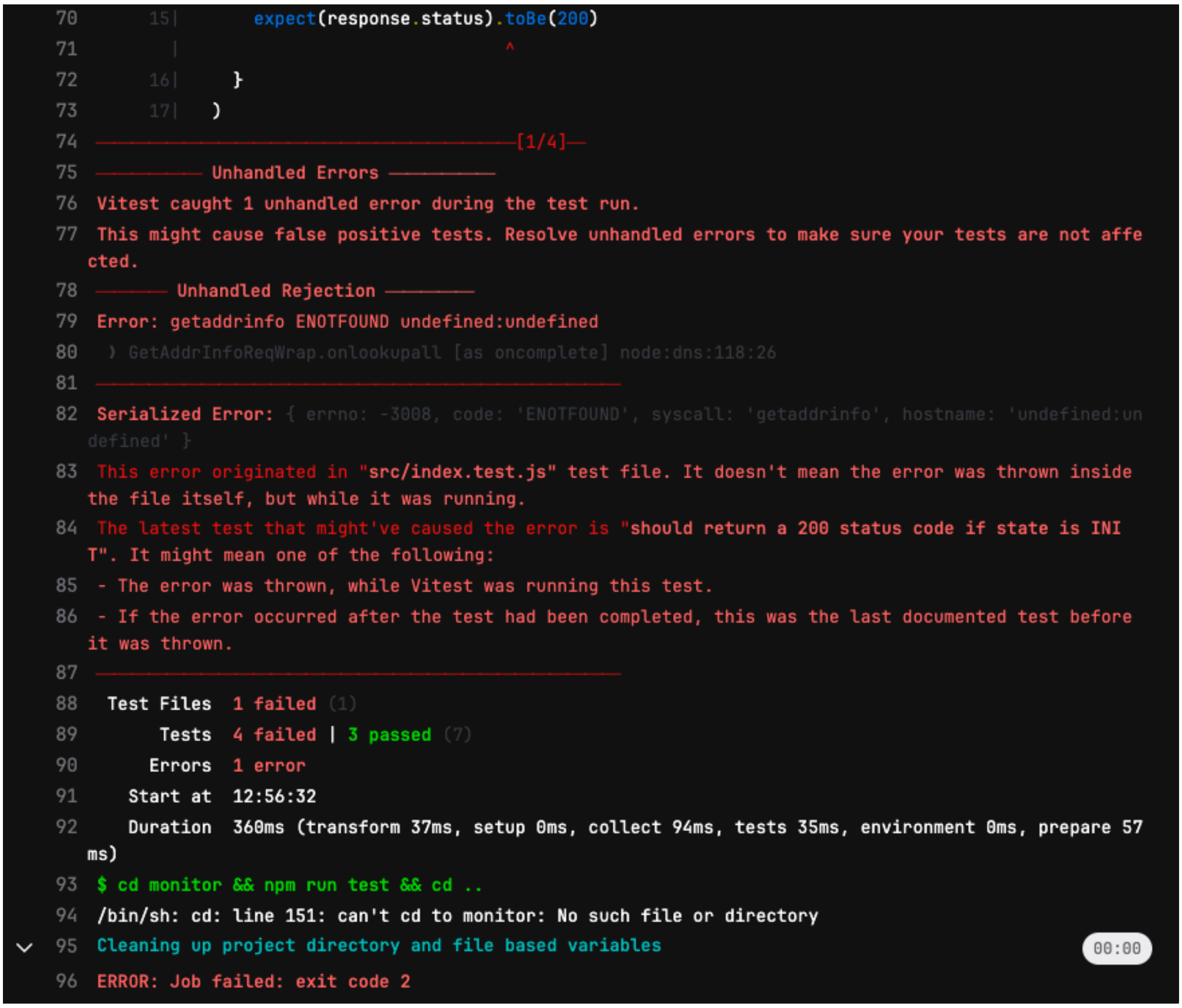
Search job log

```
1  Running with gitlab-runner 16.8.0 (c72a09b6)
2    on My Docker Runner gaGCjB21y, system ID: s_d73dc4e21c70
3  Preparing the "docker" executor                                    00:21
4  Using Docker executor with image node:alpine ...
5  Starting service docker:dind ...
6  Pulling docker image docker:dind ...
7  Using docker image sha256:646fba099175305554e28cbba10e8018c6d1b7029aaf84a4c07bc2ee754706c8 for docke
   r:dind with digest docker@sha256:081b45ad5278e6a5a6c80a66fc36cb5bccf852960e37c830ddea925bfeee7ea6 ...
8  Waiting for services to be up and running (timeout 30 seconds)...
9  Pulling docker image node:alpine ...
10 Using docker image sha256:6a28ec2adefbad5b790c6c86721c8eb39795c77efb3b5893919b21c092728f60 for node:
   alpine with digest node@sha256:a8beafd69068c05d09183e75b9aa679b520ba68f94b19c90d0da9f307f9f6565 ...
11 Preparing environment                                              00:00
12 Running on runner-gagcjb21y-project-146-concurrent-0 via Allans-MBP...
13 Getting source from Git repository                                 00:01
14 Fetching changes with git depth set to 20...
15 Reinitialized existing Git repository in /builds/allan.li/allan.li_private_project/.git/
16 Checking out 0a56e517 as detached HEAD (ref is project)...
17 Removing api-gateway/node_modules/
18 Removing monitor/node_modules/
19 Removing service1/.ruff_cache/
20 Removing service2/node_modules/
21 Skipping Git submodules setup
22 Executing "step_script" stage of the job script                    00:07
23 Using docker image sha256:6a28ec2adefbad5b790c6c86721c8eb39795c77efb3b5893919b21c092728f60 for node:
   alpine with digest node@sha256:a8beafd69068c05d09183e75b9aa679b520ba68f94b19c90d0da9f307f9f6565 ...
24 $ cd api-gateway && npm ci && cd ..
25 added 330 packages, and audited 331 packages in 5s
26 109 packages are looking for funding
27   run `npm fund` for details
28 found 0 vulnerabilities
29 $ cd monitor && npm ci && cd ..
30 added 320 packages, and audited 321 packages in 1s
31 107 packages are looking for funding
```

Figure 3 - Failing tests 1/3

```
32   run `npm fund` for details
33 found 0 vulnerabilities
34 $ cd api-gateway && npm run test && cd ..
35 > api-gateway@1.0.0 test
36 > vitest
37  RUN  v1.2.1 /builds/allan.li/allan.li_private_project/api-gateway
38 stdout | Server.<anonymous> (/builds/allan.li/allan.li_private_project/api-gateway/src/index.js:34:1
   1)
39 API-gateway listening on port 8083
40 stdout | src/index.test.js > PUT /state > should return a 200 status code if state is INIT
41 Monitor AMQP init:  Error: getaddrinfo ENOTFOUND undefined:undefined
42     at GetAddrInfoReqWrap.onlookupall [as oncomplete] (node:dns:118:26) {
43   errno: -3008,
44   code: 'ENOTFOUND',
45   syscall: 'getaddrinfo',
46   hostname: 'undefined:undefined'
47 }
48  ❯ src/index.test.js  (7 tests | 4 failed) 35ms
49    ❯ src/index.test.js > PUT /state > should return a 200 status code if state is INIT
50      → expected 400 to be 200 // Object.is equality
51    ❯ src/index.test.js > PUT /state > should return a 200 status code if state is RUNNING
52      → expected 400 to be 200 // Object.is equality
53    ❯ src/index.test.js > PUT /state > should return a 200 status code if state is PAUSED
54      → expected 400 to be 200 // Object.is equality
55    ❯ src/index.test.js > PUT /state > should return a 200 status code if state is SHUTDOWN
56      → expected 400 to be 200 // Object.is equality
57 ─────────── Failed Tests 4 ───────────
58  FAIL  src/index.test.js > PUT /state > should return a 200 status code if state is INIT
59  FAIL  src/index.test.js > PUT /state > should return a 200 status code if state is RUNNING
60  FAIL  src/index.test.js > PUT /state > should return a 200 status code if state is PAUSED
61  FAIL  src/index.test.js > PUT /state > should return a 200 status code if state is SHUTDOWN
62 AssertionError: expected 400 to be 200 // Object.is equality
63 - Expected
64 + Received
65 - 200
66 + 400
67  ❯ src/index.test.js:15:31
68      13|         .set('Content-Type', 'text/plain')
69      14|         .send(state)
```

Figure 4 - Failed tests 2/3



```
     15|        expect(response.status).toBe(200)
       |                                 ^
     16|     }
     17|   )
⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯[1/4]⎯
⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯ Unhandled Errors ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯
Vitest caught 1 unhandled error during the test run.
This might cause false positive tests. Resolve unhandled errors to make sure your tests are not affe
cted.
⎯⎯⎯⎯⎯⎯⎯ Unhandled Rejection ⎯⎯⎯⎯⎯⎯⎯⎯
Error: getaddrinfo ENOTFOUND undefined:undefined
 ❯ GetAddrInfoReqWrap.onlookupall [as oncomplete] node:dns:118:26
⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯
Serialized Error: { errno: -3008, code: 'ENOTFOUND', syscall: 'getaddrinfo', hostname: 'undefined:un
defined' }
This error originated in "src/index.test.js" test file. It doesn't mean the error was thrown inside
the file itself, but while it was running.
The latest test that might've caused the error is "should return a 200 status code if state is INI
T". It might mean one of the following:
- The error was thrown, while Vitest was running this test.
- If the error occurred after the test had been completed, this was the last documented test before
it was thrown.
⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯
 Test Files  1 failed (1)
      Tests  4 failed | 3 passed (7)
     Errors  1 error
   Start at  12:56:32
   Duration  360ms (transform 37ms, setup 0ms, collect 94ms, tests 35ms, environment 0ms, prepare 57
ms)
$ cd monitor && npm run test && cd ..
/bin/sh: cd: line 151: can't cd to monitor: No such file or directory
Cleaning up project directory and file based variables                                      00:00
ERROR: Job failed: exit code 2
```

Figure 5 - Failed tests 3/3

3.2 Passing tests

tests

Passed Started just now by allan_li

Search job log

```
Running with gitlab-runner 16.8.0 (c72a09b6)
  on My Docker Runner gaGCjB21y, system ID: s_d73dc4e21c70
Preparing the "docker" executor
Using Docker executor with image node:alpine ...
Starting service docker:dind ...
Pulling docker image docker:dind ...
Using docker image sha256:b9d1abe98c7a28077b118f96eb755d81ec5bc2308057c5344733057b92a62c2a for docker:dind with digest docker@sha256:91c7180444442d2596355a2859317d86c73025c63c67bceca3e16cd09a08e649 ...
Waiting for services to be up and running (timeout 30 seconds)...
Pulling docker image node:alpine ...
Using docker image sha256:6a28ec2adefbad5b790c6c86721c8eb39795c77efb3b5893919b21c092728f60 for node:alpine with digest node@sha256:a8beafd69068c05d09183e75b9aa679b528ba68f94b19c90d0da9f307f9f6565 ...
Preparing environment                                                                                    00:00
Running on runner-gagcjb21y-project-146-concurrent-0 via Allans-MBP...
Getting source from Git repository                                                                       00:01
Fetching changes with git depth set to 20...
Reinitialized existing Git repository in /builds/allan.li/allan.li_private_project/.git/
Checking out 45d5df12 as detached HEAD (ref is project)...
Removing api-gateway/node_modules/
Removing monitor/node_modules/
Removing service1/.ruff_cache/
Removing service2/node_modules/
Skipping Git submodules setup
Executing "step_script" stage of the job script                                                          00:08
Using docker image sha256:6a28ec2adefbad5b790c6c86721c8eb39795c77efb3b5893919b21c092728f60 for node:alpine with digest node@sha256:a8beafd69068c05d09183e75b9aa679b528ba68f94b19c90d0da9f307f9f6565 ...
$ cd api-gateway && npm ci && cd ..
added 330 packages, and audited 331 packages in 4s
109 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
$ cd monitor && npm ci && cd ..
added 339 packages, and audited 340 packages in 1s
109 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
$ cd api-gateway && npm run test && cd ..
> api-gateway@1.0.0 test
> vitest
 RUN  v1.2.1 /builds/allan.li/allan.li_private_project/api-gateway
stdout | Server.<anonymous> (/builds/allan.li/allan.li_private_project/api-gateway/src/index.js:48:11)
API-gateway listening on port 8083
 ✓ src/index.test.js  (7 tests) 32ms
 Test Files  1 passed (1)
      Tests  7 passed (7)
   Start at  19:28:59
   Duration  399ms (transform 52ms, setup 0ms, collect 108ms, tests 32ms, environment 0ms, prepare 80ms)
$ cd monitor && npm run test && cd ..
> monitor@1.0.0 test
> vitest
 RUN  v1.2.1 /builds/allan.li/allan.li_private_project/monitor
stdout | Server.<anonymous> (/builds/allan.li/allan.li_private_project/monitor/src/index.js:35:11)
Monitor listening on port 8087
 ✓ src/index.test.js  (3 tests) 11ms
 Test Files  1 passed (1)
      Tests  3 passed (3)
   Start at  19:29:00
   Duration  379ms (transform 27ms, setup 0ms, collect 98ms, tests 11ms, environment 0ms, prepare 71ms)
Cleaning up project directory and file based variables                                                   00:00
Job succeeded
```

Figure 6 - Passing tests

4 Reflections

4.1 Main learnings and worst difficulties

1. GitLab runner – the CI/CD pipeline was implemented and tested on the secondary course GitLab instance. While instructions were available for setting up the GitLab runner with a self-signed certificate, the process was still confusing and time-consuming.
2. Docker daemon socket on macOS – On macOS, Docker Desktop does not set up a `/var/run/docker.sock` file by default. This resulted in errors being thrown when running jobs on GitLab, as GitLab could not find the runner's Docker executor.
3. Python multithreading for non-blocking processes – Python threads were a novel concept for me prior to working on this project, and I spent a significant amount of time writing working code for handling state changes in the Python service. In the end, I figured out how to have two separate threads running in addition to the main thread. One was used to consume state change messages from RabbitMQ, and the other was used to process a message loop that reacts to the state changes.
4. Docker-in-docker – Docker-in-docker was also unknown territory for me, but thankfully there were many resources available for writing a proper CI config file that utilized docker-in-docker. In the end, I managed to make it work.
5. API testing – my experience with testing mostly consisted of frontend testing prior to this project. Now I have also gotten the chance to try out some backend/API testing, specifically with tools such as Supertest.

4.2 Amount effort (hours) used

I spent around 40 hours on this project.