Micro-Service Assessment Project

Assessee: Benedicta Baidoo

Assessor: Tory Clasen

Assessment Start Date: 15 Jan 2022 – 8am

Assessment End Date: 17 Jan 2022 – 8am

# Operations Requirements:

1. The system must be entirely dockerized where the entire system can be executed with a single command of `docker compose up` that pulls the pre-built images from github or docker hub.
2. The system must use automation to test, build, and deliver artifacts when merging into the trunk as published in https://cloud.google.com/architecture/devops/capabilities .
3. The system must have all deployable artifacts published to github releases, artifacts, or docker hub.
4. The system may publish language specific build artifacts to github releases, artifacts, or their language specific repositories (npm, pypi, ...).
5. When building the system docker images:
   1. all images should be running as non-root.
   2. all images should be as minimal in size as possible.
   3. all images must not contain any high or critical vulnerabilities as listed by https://github.com/aquasecurity/trivy .
6. When running the system, the web user interface (UI) shall be accessible on `http://localhost:8000`, assuming it is running on localhost.

# Development Requirements:

1. The UI must be built using a singular web component as defined in <https://developer.mozilla.org/en-US/docs/Web/Web_Components> that is used multiple times, one button for each back-end service.
2. Each button when pressed must make an AJAX call to one of the backend service API endpoints, retrieve a json object, and render it to page.
3. All backend services must conform to standards as published in https://dodcio.github.io/eads/ .
4. The backend services must be written in the following languages:
   1. Javascript
   2. Python
   3. Golang
   4. Rust
5. Each of the backend services must implement one of the following API endpoints (in no order):
   1. “/people” that returns a collection of names from a datastore.
   2. “/time” that returns the current timestamp.
   3. “/browser” that returns the user agent string of the requesting user.
   4. “/counter” that returns an integer value starting at 0 and increments by 1 each time it is called.

# Organizational Requirements:

1. Every commit in the trunk branch must conform to standards as published in https://www.conventionalcommits.org/en/v1.0.0/ .
2. Every commit in the trunk branch must pass all unit tests with 100% code coverage as published in https://trunkbaseddevelopment.com/ .
3. The source code repository must be tidy and easy to navigate.
4. The source code repository must include a README.md conforming to https://www.makeareadme.com/ .
5. The readme must include instructions for a local build of the system.
6. The readme must include instructions to test each of the services.
7. The readme must include instructions for running the project locally.
8. The readme must include a description of each API endpoint and what it returns.

# Assessment Grading Criteria:

1. Completion of functional requirements
2. Completion of non-functional requirements
3. Code quality (subjective)
4. Time management (subjective)
5. Skills displayed in <https://sijinjoseph.netlify.app/programmer-competency-matrix/>

# Assessment Grading Steps:

1. `git clone`
2. `docker compose up`
3. Navigate to <http://localhost:8000> in modern chrome
4. Click each button several times while watching the network tab in dev-tools
5. Inspection of source code in git
6. Inspection of delivered artifacts
7. Inspection of git commit history
8. Inspection of README
9. Discussion!