SOFTWARE REQUIREMENTS SPECIFICATION

for

S.M.A.R.T

Sprint Mapping and Annotation Recommendation Tool

Version 0.1.4

Prepared by : Alexander Nemirovskiy

October 9, 2020

1 Requirements

1.1 Information

The current software version 0.1.4 is a web application that consists of two docker containers activated through a docker-compose file provided with the installation package.

A backend API container is created exposing a process running on the host machine and listening on port 8080 by default. This process represents the core of the application and it is not necessary to expose the port it is using as it is meant for internal communications only unless the goal is to create alternative UI to what is provided in the package.

An additional NginX container is orchestrating incoming http requests and acting as a reverse proxy towards the internal API process. It expects port 80 to be exposed on the host machine by default. This container is secondary to the application usage and it is meant only to facilitate the installation process and can be replaced with a custom service to act as a web server for the application's UI.

1.2 Technical requirements

The application was tested on the following configurations:

- CentOS/RHEL based Amazon Linux AMI 2018.03 x64
- 8 vCPUs 2.3 GHz, Intel Broadwell E5-2686v4
- 32 GiB memory
- 32GiB

Although not necessary to replicate, these configurations allowed the application to be run asynchronously and in a multi-threaded environment which inherently augmented the throughput.

The minimal set of requirements are defined as follows:

- An operative system capable of running docker and compose version 3 or above
- a multi-core CPU with at least 2GHz frequency
- at least 5GB of available storage on the disk
- at least 8GB of RAM

2 Configuration

2.1 Host machine setup

The application can be delivered in two different ways and both expects the host machine to have at least docker to be installed and running on it.

Minimal setup The distributed package for the minimal solution contains the following items:

- a docker image for the main API
- a dist folder containing the compiled UI frontend of the application to be exposed through a web server
- a model binary file and a jar file used by the backend services

This solution is meant for the case in which the host machine will not be relying on the configured Nginx service to run its web service.

2.2 Configuring the application

Standard configuration

1. Create a folder on the host machine and put the contents of the application package in it. This will be the working directory of the application.
2. Navigate to the `reverse_proxy` folder and open the `proxy.conf` file with a standard text editor.
3. The setting inside `upstream` block represents the internal process port the backend API will be running on and to which the NginX will forward http requests. This is not meant to be exposed to an external user necessarily and it is for internal use only. Change the parameters to reflect the host configurations.
4. Navigate back to the root directory and start the docker-compose process using the `docker-compose.yml`
5. Once the process is finished put the `jaxb_impl-0.1.3.jar` and the `model.bin`files into the input folder
6. Now start the docker containers using docker-compose once more.

Minimal setup: custom web-service configuration

1. Create a folder on the host machine and put the contents of the application package in it. This will be the working directory of the application.
2. In the root folder use the existing Dockerfile to build the image for the backend API.
3. As an optional step, one can change the port number the API application will be listening on to a custom one in the last line of the Dockerfile before building the image
4. Using a web server of one's choice, expose the contents of the dist folder to make the user interface accessible via a web browser

Additional information If it is required to change the internal API service port, the Dockerfile in the root folder needs to be modified according to the desired configuration: change the port number the last line of the file. If the NginX service is used as well the aforementioned change shall be reflected in the `upstream` block of the `proxy.conf` file. Both files shall contain the same port number for the application to work unless another custom service is to be used instead of NginX otherwise the API won't work.