SOFTWARE REQUIREMENTS SPECIFICATION

for

S.M.A.R.T

Sprint Mapping and Annotation Recommendation Tool

Version 0.2.1

Prepared by : Alexander Nemirovskiy

October 15, 2020

0.1 Information

The current software version 0.2.1 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 that acts as a web server for the application's UI. In the latter case the minimal configuration installation should be followed.

0.2 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 GB RAM
- 32 GB internal dick storage
- docker and docker-compose installed
- npm v6.14.x and angular CLI v9.1.x installed

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

0.3 Host machine setup

The application is delivered in a package containing the necessary modules to build and run the program. It expects the host machine to have at least docker installed and available on it. The default installation process requires some ports to be open as well for either internal usage or external calls to the program.

This package contains the following items:

- an `app` folder containing the source code of the python backend.
- a `frontend/angularUI` folder containing the UI project to be compiled, built and later exposed through a web server.
- a `Dockerfile` and a `docker-compose` files to build and start the application.
- three folders, namely `uploads, input, output` that contain, or will contain, all the additional files used or generated by the application such as a model binary file and a jar file used by the backend services.

0.4 Configuring the application

Standard configuration This is the default configuration option. It will be using ports 80 and 8081 on the host machine to run the application. These settings can be changed at any time according to the host machine availability and setup. The following steps describe what needs to be done in order to launch the application without any change to the settings.

1. Extract the contents of the provided package into a folder. From now on this will be referred as the root folder of the application.
2. Navigate to the `frontend/angularUI` folder in order to change the IP address of the server that will be hosting the frontend application
   a) in `src/environments` folder open the `environment.prod.ts` file and change the `API_Endpoint` value to the host machine's actual IP address, leaving the `api/v1` suffix as is. The final value should look like this:

      `http(s)://your.hostname.or.domain/api/v1`

   b) leave everything else as is.
3. Navigate back to the root directory and start the docker-compose process using the `docker-compose.yml`. This will start the build process of the application for both frontend and backend parts.
4. (This step is optional if the package already contains the mentioned files. If those files have been downloaded separately this step is mandatory) Once the building process is finished put the `jaxb_impl-0.1.3.jar` and the `model.bin` files into the input folder.

5. Now start the docker containers using docker-compose once more.

Additional information If it is required to change the internal API service port, both the docker-compose file in the root folder and the nginx proxy configuration file need to be modified according to the desired configuration: change the port number in the command line of the docker-compose file that starts the uvicorn server for the API. 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. To summarize the needed steps:

1. In the root folder, open the docker-compose file and change the last parameter in line `command: [...]` that indicates the port on which the API will be running
2. Navigate to the `frontend/angularUI/service_conf` 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 be identical to the choice made in the previous step.