**Runbook / Setup / Debugging**

**Tools Used:**

|  |  |  |
| --- | --- | --- |
| **Tools** | **Usage** | **Version** |
| Visual Studio Code | IDE | -- |
| MongoDB | Helps in executing the code in a separate environment. | Suggested 4.4 |
| Postman | Used to call/request the API | Updated Version |
| Robo 3T | Data in the database are visualized | Updated Version |
| Anaconda | Executing the code by creating a separate environment. | Updated Version |
| WinSCP | Updating the code in the server. | Updated Version |
| GIT | Backup and collaborating | Updated Version |

**Libraries:**

**NOTE :** All the Libraries are given in “requirements.txt” file. Only the Sensitive one are mentioned over here.

|  |  |
| --- | --- |
| **Library** | **Version** |
| SpaCy | 2.3.5 |
| typing\_extensions | 4.1.1 |
| Spicy | 1.8.0 |
| en-core-web-sm | <https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.3.1/en_core_web_sm-2.3.1.tar.gz> |
| Python | 3.9.7 (Suggested) |

**Steps to execute:**

* Create a environment in Anaconda and install all the requirements.
* If running for first time.
  + Ensure all the libraries in the “requirements.txt” file is installed.
  + Being an Flask application this takes the main file as “app.py”
  + Before running the “app.py” just for testing purposes run the below mentioned files individually just to test all the packages are installed properly.
    - chart\_generator.py
    - entity\_extractor.py
    - data\_generator.py
  + Create an error log folder in the same folder where you are running the code.
  + Error log folder name should be mentioned in the “app.py” file.
  + Create a folder called “db” in C – Drive of the desktop to save the data in the database.
  + Once after successful installation of mongodb, add it to the system path.
  + Though we are running the application locally, ensure to start the “mongodb”.  
    Methods to start the “mongodb” is mentioned below.
    - Open an command prompt and run the command “mongod”
    - Parallelly open another command prompt and run the command “mongo”
    - This will start the mongodb instance locally. If this throws a error, kindly check the version of mogoshell and db. Also check whether it starts properly in the command prompt.
* In the app.py, ensure the path is changed to local storage path and port mentioned in the constructor is free to host the application.
* Once on a successful execution of the above steps, run the “app.py” file. Probably it should not throw any errors, If so, the errors will be mostly related to packages check with version and packages installed.
* To check its running successfully initially check in the environment and then check in the local browser with route – <http://localhost:7000/test>

**Execution in the Server:**

* Setting up the server.
  + Login as sudo user using the command – “sudo su”
  + Create a new environment in the server command prompt and Install all the requirements mentioned as given in the above “Library” topic.
  + Activate the Environment and run the source files initially. Once successfully running the source file run the main file i.e app.py
  + The execution is tested by running the route ‘http://SERVER\_IP:7000/test’ and terminate the execution.
  + Being this is an server, the application should run continuously and on failure/bugs also it should run again continuously. So, lets create a service to do this.
  + Create a Gunicorn Service in the server for visual workbench backend.  
    NOTE: The service code is attached in the folder as “vaas\_app.service”.
  + Start the service by executing the given code in the cmd “sudo systemctl start vaas\_app”  
    NOTE: “vaas\_app” is the service name
  + If the server is down or not up to its limit just restart the server by using the command  
    “sudo systemctl restart vaas\_app”
  + If you wanted to stop the instance, use the below command.   
    “sudo systemctl stop vaas\_app”
* To push the code to the server, Use the WinSCP tool.   
  Login into the server using WinSCP and navigate to the required folder. By dragging and dropping across the server folder from local system the files can be uploaded to the server.   
  NOTE: Ensure the dragging and dropping is done exactly.
* Start the service by executing the given code in the cmd “sudo systemctl start vaas\_app”  
  NOTE: “vaas\_app” is the service name
* If the server is down or not up to its limit just restart the server by using the command  
  “sudo systemctl restart vaas\_app”. If it doesn’t works out then stop the instance and run the file manually in the environment and debug the errors.
* If you wanted to stop the instance, use the below command.   
  “sudo systemctl stop vaas\_app”