Start in virtual environment

Link to requirements.txt: <https://linuxhint.com/python-requirements-txt-file/>

<https://www.websense.com/content/support/library/data/v84/remediation_scripts/multi-platform_python.aspx>

Add ReadMe file on how to run and setup the code.

1. Explanation for the GUI application setup.
2. Explanation for the Web application communication setup.
3. Explanation for the CMD/Linux Terminal set up.

Provide Documentation of the function and classes used in the application.

1. Add diagrams showing the flow of your application
2. Provide function documentation of what they do, the parameters they expect and output they return
3. Function name and what it does
4. Parameter name | Type | Use case
5. Output value | Type | Meaning of the output value

Ask if a UI is needed or a bare command line will be used for this application system.

Instructions on how to run on Windows and Linux machine

Running python script from JavaScript:

<https://www.youtube.com/watch?v=lSAFVMaaH-w&ab_channel=ApesinCapes>

Solutions:

1. Using a GUI in python e.g Tkinter
2. Using the Web HTML/CSS/Django/Tailwind
3. Using the Command Line

Point to talk about in your report

1. REP-REQ being stateless
2. Why we needed an asynchronous approach
3. Using threading to simulate the sending and receiving of messages
4. “When you use REQ to talk to REP, you get a strictly synchronous request-reply dialog. The client sends a request. The service reads the request and sends a reply. The client then reads the reply. If either the client or the service try to do anything else (e.g., sending two requests in a row without waiting for a response), they will get an error.” – Important to add this to your report

Code Documentation:

Extended **Request-Reply pattern**

Diagram

Description automatically generated***The request-reply broker binds to two endpoints, one for clients to connect to (the frontend socket) and one for workers to connect to (the backend). To test this broker, you will want to change your workers so they connect to the backend socket. Here is a client that shows what I mean:***

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated**The Request-Reply can either use a poll or proxy.**

Graphical user interface, text, application

Description automatically generated

**Handling Ctrl-C signal**

Graphical user interface, text, application, email

Description automatically generated

**Why we need a ROUTER SOCKET INTERFACING OUT CLIENTS CONNECTING**

***The ROUTER socket, unlike other sockets, tracks every connection it has, and tells the caller about these. The way it tells the caller is to stick the connection*identity*in front of each message received. An identity, sometimes called an*address*, is just a binary string with no meaning except “this is a unique handle to the connection”. Then, when you send a message via a ROUTER socket, you first send an identity frame.***

**Why we need not to use a REP SOCKET while interfacing with the AppManager receiving connection**

*Incidentally the REP socket can only deal with one request-reply exchange at a time, which is why if you try to read multiple requests or send multiple replies without sticking to a strict recv-send cycle, it gives an error.*