TECHNICAL DOCUMENTATION

By: Hrid Chakraborty

Date: 5th February 2024

Project Name:

Infoview Generation using AI Prompts

Context:

Users manually set separate parameters for each tag to generate visualizations. This can get time consuming when there are over 100 parameters involved.

Objective:

To create an AI-based solution, which takes in a user prompt in natural language and generates visualizations. This can greatly boost user experience and drastically reduce time spent configuring visualization settings.

Current Status:

The project is mostly complete. Currently, only the most used and the most relevant “InfoViewSettings” object parameters are being generated by the AI. This is due to limitations in context size.

Main Tools Used:

1. [Python](https://www.python.org/) programming lanugage

2. [Flask](https://flask.palletsprojects.com/en/stable/) framework for backend API implementation

3. [Azure Power Automate](https://make.powerautomate.com/) (GPT-4o)

4. [Postman](https://www.postman.com/) for testing the API endpoint

Project Workflow:

1. The user clicks on the “Ai” button which is on the left sidebar in the “Infoview” page.

2. User enters a prompt and clicks on “Submit”. In the frontend, ivs.get() is called, which returns the current state of the infoView by returning an infoViewSettings object. (You can try this by typing ivs.get() in the browser console)

3. This object, along with the prompt is sent to the backend API server.

4. The backend server first makes an API call to the Azure server by sending only the prompt.

5. The Azure server returns data in structured natural language object(not yet usable).

6. The backend applies postprocessing by merging AI output with infoViewSettings object.

7. The backend returns a modified infoViewSettings object, which is then used in the frontend to set the visualizations (using ivs.set()).

API Usage:

API Endpoint: http://<host\_ip>/GetInfoViewFromPrompt  
Content-Type: application/json

Method: POST

API Body:

{

“userPrompt”: <user\_prompt>,

“infoViewSettings”: <the\_object\_returned\_from\_ivs.get()>

}

(NOTE: Be careful when copy-pasting the body from this document. The quotation marks are rich text and can cause issues.)

Config File Format:

Type: json

Body:

{

“API\_URL”: <azure\_url>,

“TAG\_DETAIL\_SEARCH\_API\_PREFIX”: <>,

“INSIS\_TOKEN”:<>,

“INSIS\_KEY”:<>,

“TAG\_SEARCH\_API”:<>

}

Main Project Usage Guide (requires Python 3.12.8):

1. Create a virtual environment named “myenv” in the project folder using “python -m venv myenv”

2. Activate the virtual environment using “./myenv/Scripts/activate”

3. Run “pip install -r requirements.txt” to install all the necessary packages and dependencies

4. Create a config.json file, if it does not exist. Follow the format given in the documentation.

5. Start the server by running “python main.py”. Make sure that the venv is activated while doing so.

6. The endpoint is localhost:5000/GetInfoViewFromPrompt. Test it with Postman.

Future Work:

1. Improving the prompt logic, possibly making it scalable for future extensions.

2. Adding speech-to-infoview functionality by converting speech to text (prompt) and using that prompt for infoview generation.

The code will be available in the Azure Repo “inSisAskIT” under the “infoViewAI-backend” folder