**Concisys Software Project Report**

Concisys BOM Quoter

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description | Author(s) |
| 1.0 | 8/02/2023 | Initial document release | Monica Nguyen, Lan Do |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History 1](#_Toc141861182)

[1 Overview 3](#_Toc141861183)

[2 How It Works 4](#_Toc141861184)

[2.1 Project Overview 4](#_Toc141861185)

[2.2 Python Libraries 4](#_Toc141861186)

[2.3 Vendor APIs 5](#_Toc141861187)

[DigiKey API 5](#_Toc141861188)

[Mouser API 5](#_Toc141861189)

[TTI API 5](#_Toc141861190)

[2.4 Main Functionality 6](#_Toc141861191)

[3 Installation 8](#_Toc141861192)

[3.1 System requirements 8](#_Toc141861193)

[3.2 Setup 8](#_Toc141861194)

[4 User Guide 8](#_Toc141861195)

[4.1 Setting up input BOM 8](#_Toc141861196)

[4.2 Uploading BOM 11](#_Toc141861197)

[5 Issues and Challenges 13](#_Toc141861198)

[6 Future Development and Improvements 13](#_Toc141861199)

[Conclusion 13](#_Toc141861200)

# Overview

The **Concisys BOM Quoter** is a Python-based CLI program that is designed based off of the current Concisys ERP ‘Material Inquiry’ feature, in which users (e.g., program managers) upload a Bill of Materials (BOM) excel file to quote part items based on available vendors. It enables users to upload a BOM from their computer via the command line and returns the file with available part items and their attributes to determine accurate labor price quotes.

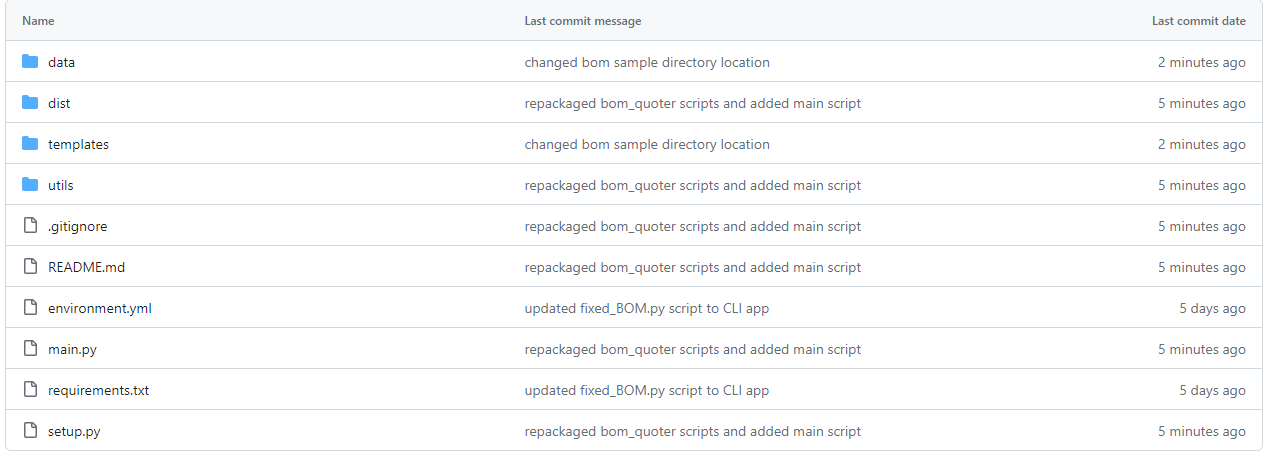
While the BOM Quoter has many similar functions as the ERP feature, this program also allows the user to compare pricing results based on the quantity order. For example, if a original quantity order is currently being quoted for 100 boards but a customer would also like to know the quote for a later time with 1000 boards of the same type of order, the quantity price will be determined and compared in the BOM Quoter..

# How It Works

## 2.1 Project Overview

Project Location: <https://github.com/concisys-dev0/bom_quoter>

The project folder hierarchy is as follows:



* **data** – contains corresponding JSON files to store API credentials and results
* **dist** – contains any scripts that are under development or testing and not yet released
* **templates** – contains blank BOM templates for ease of access; also includes BOM samples folder (**samples**) for testing and comparison
* **environmental.yml** – for Anaconda virtual environment
* **main.py** – Main functionality
* **requirements.txt** – required libraries to be set up via pip if desired
* **setup.py** – setup script to initiate `utils` library

## 2.2 Python Libraries

The Python language was preferred over other languages for building the BOM Quoter due to its versatility, simplicity, and widespread adoption. With its extensive ecosystem of libraries and frameworks, it provides a solid foundation for connecting to various APIs while handling complex data operations in addition to its support of data flow and compatibility with data formats, such as JSON, XML and CSV.

The code was written in functional program approach to handle data flow. It utilizes various Python libraries and modules. The main libraries used within this program include, but are not limited to:

* Excel file handing: *pandas, numpy, openpyxl, json*
* API Requests: *requests, requests-oauthlib, webbrowser*
* Web Scraping: *selenium, webdriver-manager, beautifulsoup4*
* Other libraries include: *time, sys, os, pathlib, operator, schedule, math, setuptools*

## 2.3 Vendor APIs

The BOM Quoter allows the user to briefly connect to three RESTful APIs from DigiKey, Mouser Electronics, and TTI and returns data in JSON format.

### DigiKey API

|  |  |
| --- | --- |
| Scripts | Description |
| dk\_keyword\_buy\_info | Get information from DigiKey API search |
| dk\_oauth2\_login | DigiKey login automation to change user credentials |
| dk\_oauth2\_token | DigiKey API authentication |
| dk\_RFQ\_BOM | Parse DigiKey results and save into an excel sheet called ‘DK\_Results’ |
| dk\_search\_info | API operations for DigiKey part search |

**Authorization URL:** `https:// api.digikey.com/v1/oauth2/authorize`

**Token URL:** `https://api.digikey.com/v1/oauth2/token`

**Authentication:** The API requires OAuth2 authentication to acquire the DigiKey API access token. To get client credentials for access, create a developer account on DigiKey developer website and follow the instructions to open an organization and receive a Client ID and Client Secret.

**Rate Limits:** DigiKey API is limited to 1000 requests per day per account. Additionally, after every 120 requests, API requires a 60 second cool-down time. Initiated tokens are refreshed every 30 minutes. This program provides up to 2000-3000 requests by default.

### Mouser API

|  |  |
| --- | --- |
| Scripts | Description |
| mouser\_apiKeys | Handles user information and change user’s API key as needed |
| mouser\_RFQ\_BOM | Parse Mouser results and save into an excel sheet called ‘Mouser\_Results’ |
| mouser\_search\_buy\_info | Get information from Mouser product search |
| mouser\_search\_v1 | API operations for Mouser API v1 |
| mouser\_search\_v2 | API operations for Mouser API v2 |

**Authorization URL:** `https://api.mouser.com/api/v2/search/ `, ` https://api.mouser.com/api/v1/search`

**Authentication:** The API requires authentication via the Mouser API key. Sign up for a developer account at Mouser developer website and follow the instructions to receive a Search API key.

**Rate Limits:** Mouser API is limited to 1000 requests per day per account. Additionally, after every 30 requests, API requires a 60 second cool-down time. This program provides up to 2000 requests by default.

### TTI API

|  |  |
| --- | --- |
| Scripts | Description |
| **tti\_RFQ\_BOM** | Parse TTI results and save into an excel sheet called ‘TTI\_Results’ |
| **tti\_search\_buy\_info** | Get information from TTI API response |
| **tti\_search** | API operations for TTI product search |

**Authorization URL:** ` https://api.tti.com/service/api/v1/search/`

**Authentication:** The API requires authentication via the TTI API key. Please refer to your Concisys supervisor to get this API key.

**Rate Limits:** TTI API is limited to 1000 requests per day per account. Additionally, after every 100 requests, API requires a 60 second cool-down time. This program does not currently have request limitations due to Concisys company account credentials.

## 2.4 Main Functionality

The back-end functionality and API operations in this project are packaged under the `utils` package. The listed scripts below correspond to the following categories in the `utils` package.

|  |  |  |
| --- | --- | --- |
| Function | Parameters | Description |
| utils.input\_BOM.setup\_BOM\_info | path: path to file | Gather the information required for each vendor API and pass the BOM to a dataframe |
| utils.mergeCompare\_pricing.compare\_options\_result | path: path to file | Asking the user for which price of quantity to compare between suppliers. |
| utils.mergeCompare\_pricing.get\_compare\_results | path: path to file  compare\_column: [str] name of column to compare | Get the result from API and prepare data like convert lead time to same unit as day, replace price to be maximum number if the item notes are “No Stock”, “Not Enough Stock”, “Please Check URL”. The data passes to compare price function, then copy the lowest price item row from result data frame to new data frame and return as result. |
| utils.fixed\_BOM.df\_result\_without\_scraping | path: path to file | Take the result from get\_compare\_results, and filling mounting type, package/case, and terminals base from description. If the terminals are empty, get terminals information from the package/case base on JEDEC and JEITA standard and return result to the data frame. |
| utils.fixed\_BOM.scrape\_saved | path: path to file | Take the result from get\_compare\_results, and filling mounting type, package/case, and terminals base from description. If the terminals are empty, get terminals information from the package/case base on JEDEC and JEITA standard. The mounting type, package/case, and terminals that is blank using web scraping to fill information as much as possible. |
| utils.fixed\_BOM.save\_RQF\_BOM | path: path to file  df\_r2: dataframe of total results output | Inserts the dataframe results into the file and stylizes the excel sheet for `Best\_Prices` sheet. |
| utils.summary.save\_summary | path: path to file | Returns the count for line items and part attributes (TH, SMT, etc.) and inserts the results in a new sheet called `Summary.` |

# Installation

## 3.1 System requirements

Required:

1. Python 3.10 or higher
   * Enter the command `python -v` or `python --version` to know which version of Python you have
2. [Anaconda IDE](https://docs.anaconda.com/free/anaconda/install/index.html) or PIP (automatically installed after Python install)

Optional:

1. chromedriver.exe from https://chromedriver.chromium.org/downloads
   * Latest stable version: 113, 114

## 3.2 Setup

1. Clone the [bom\_quoter Github](https://github.com/concisys-dev0/bom_quoter) repository or download the ZIP file into your local machine
2. Install the required dependencies into a virtual environment
   1. If using Anaconda: `conda env create -n my-env-name -f environment.yml`
   2. If using PIP:
      1. Create your virtual environment: `python -m venv my-env-name`
      2. Activate your virtual env: `my-env-name\Scripts\activate.bat` (Windows), `source .venv/bin/activate` (OS)
      3. Then install dependences: `pip install -r requirements.txt`
3. Open the bom\_quoter directory in your virtual environment (cmd.exe on Windows, terminal on Mac)
4. Enter the command `pip setup.py install` to setup packages. Otherwise, the BOM Quoter will not work.
5. To activate the program, cd to main directory and enter: `python main.py` in the command line.

# User Guide

## 4.1 Setting up input BOM

Required: You must have a customer BOM to copy the ***`BOM`*** and ***`Sheet1`*** sheets into.

1. In the project directory, open or obtain the **BOM\_input\_template.xlsx** file from the **templates** directory.

A screenshot of a computer

Description automatically generated

Figure : Found under 'templates' directory in project directory

Open the BOM input template. There are two sheets that are required for the BOM Quoter processing to be successful: `*BOM*` and `*Sheet1*` (see Figure 2 below). *Note*: *do NOT delete or change the name of these sheets in your BOM file upload as it will create syntax errors.*

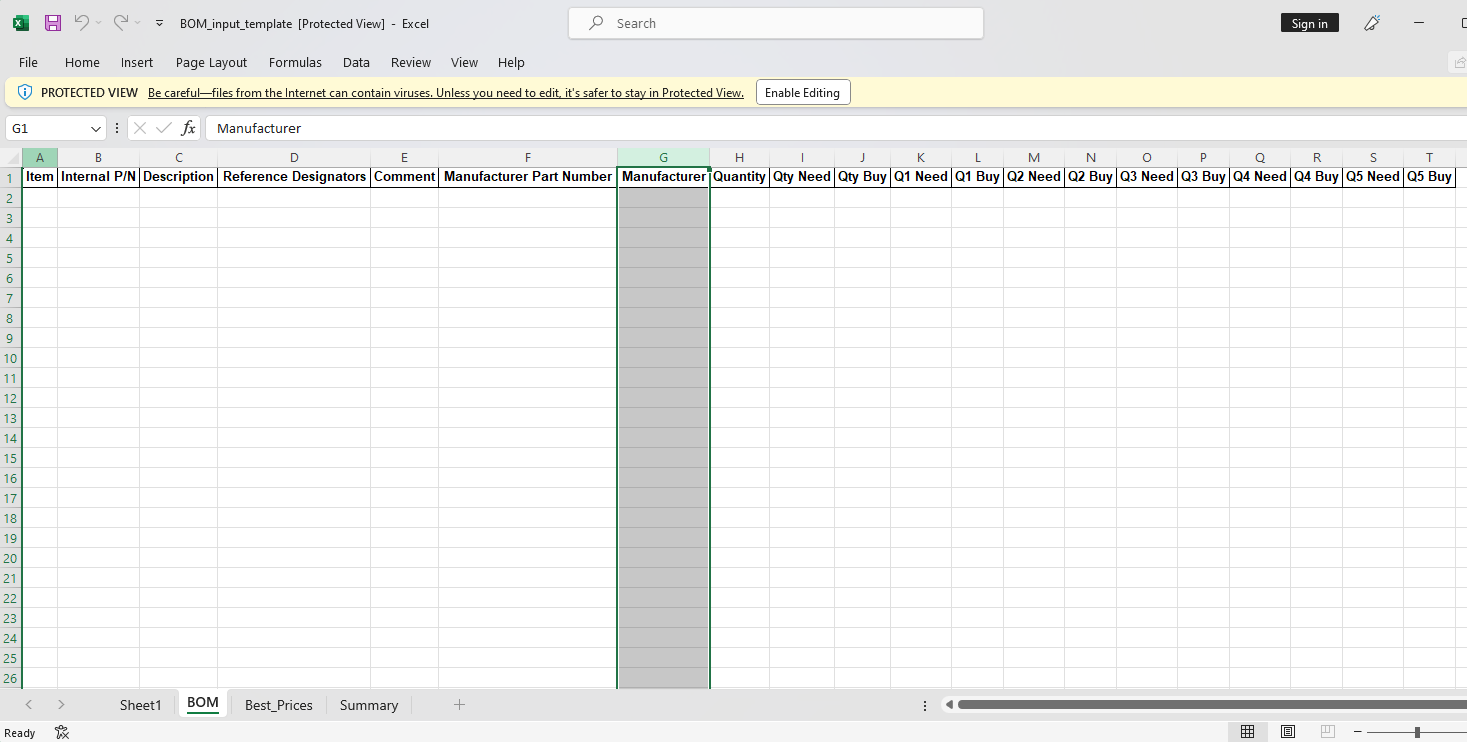


Figure : Template of the BOM sheet. Sheet1 and BOM required to be present when the BOM file is uploaded.

1. Instead, to avoid syntax errors, create a new copy of the **`*BOM*`** sheet and **`*Sheet1*`** sheet in your BOM file and copy/paste the columns from the respective sheet in the BOM template to your BOM file.
2. Then copy/paste the appropriate value data corresponding to the column from the original BOM to the ***`BOM`*** sheet in the same file. See figure 3.

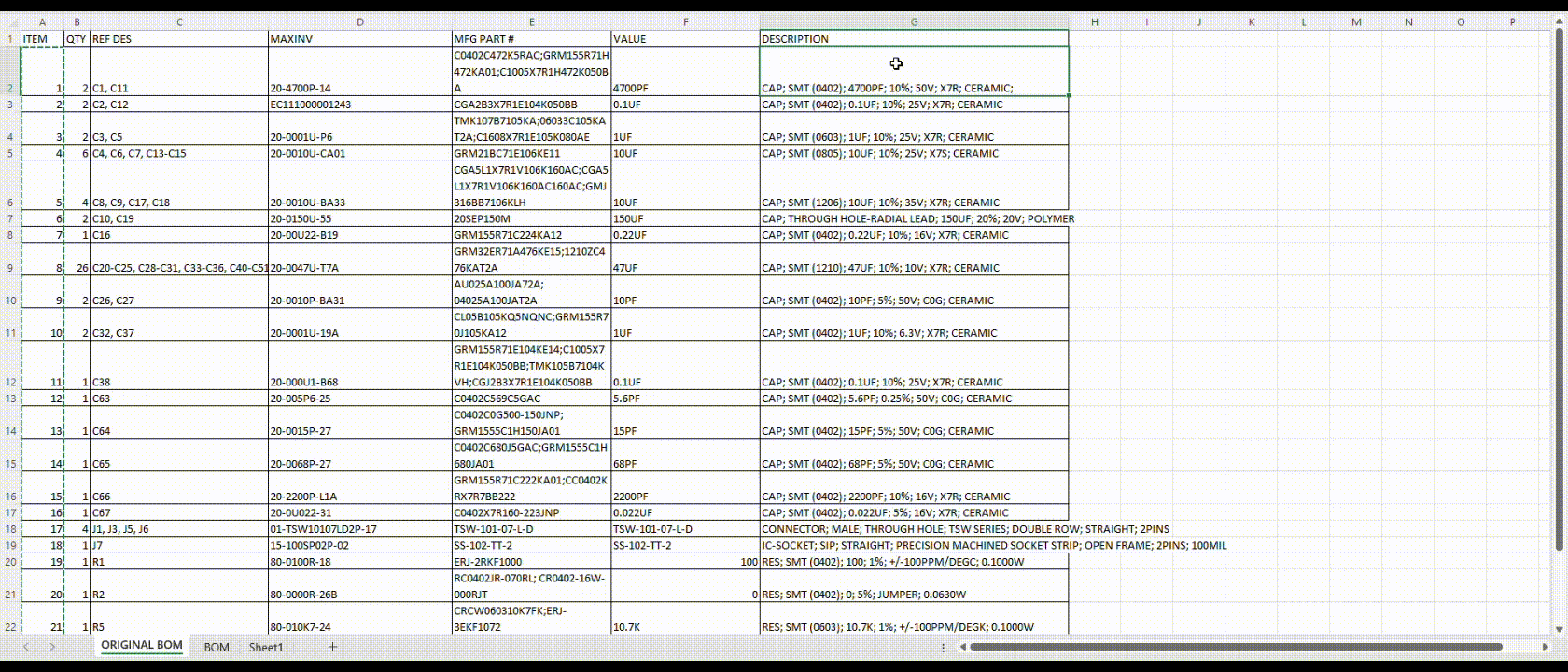


Figure : Copying original BOM data to BOM input sheet

1. All the cells in the following columns of the **`*BOM`*** sheet are required to be filled:
   1. *Item* – line item number
   2. *Reference Designators* – found in original BOM, location of the part on PCB when assemble
   3. *Description* – description of part
   4. *Manufacturer Part Number* – manufacturing part number
   5. *Quantity* – quantity of part for a single board
      1. If item is DNI/Do Not Install put in 0 if you need to keep it in the BOM
   6. *Qty Need* – the quantity needed to complete quantity order (more than 1 board)
   7. *Qty Buy* – the quantity to buy to complete quantity order (account for attrition)
2. Go to **`*Sheet1*`** that you copied into your BOM file, enter the *Qty Order* value to the *Qty* column. See figure 4.

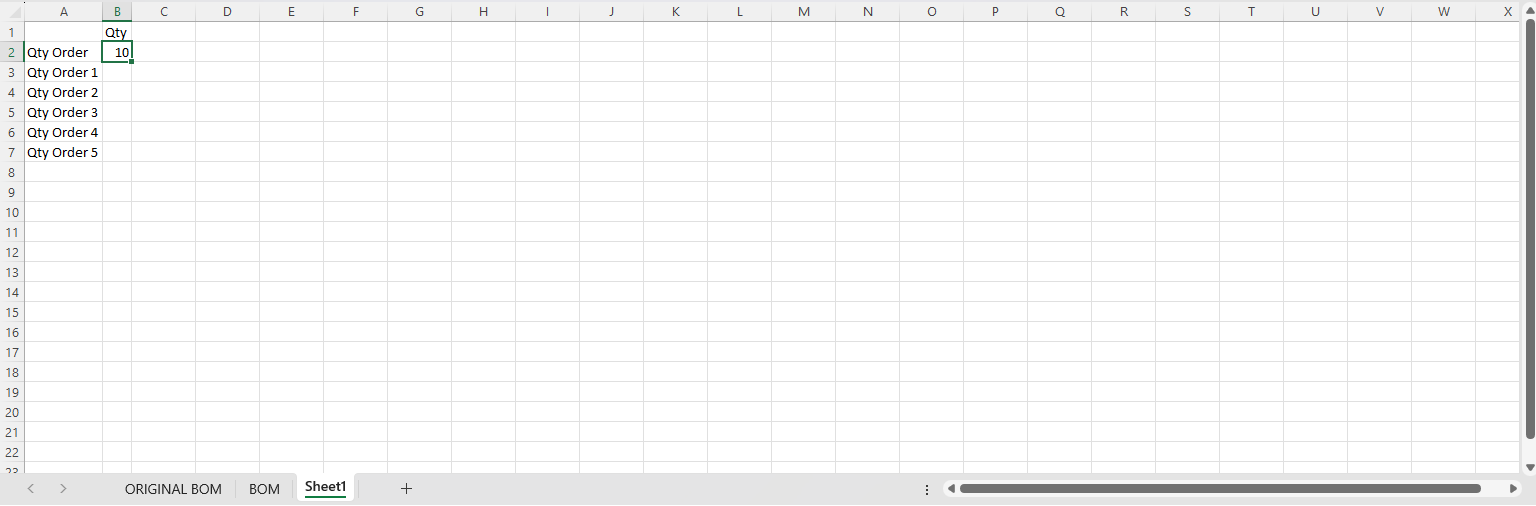


Figure : Sheet1 with Qty Order

1. If there is more than one value in the column, add the Q# Need and Q# Buy to ***‘BOM’*** where # is *Qty Order* number. See figures 5 and 6.

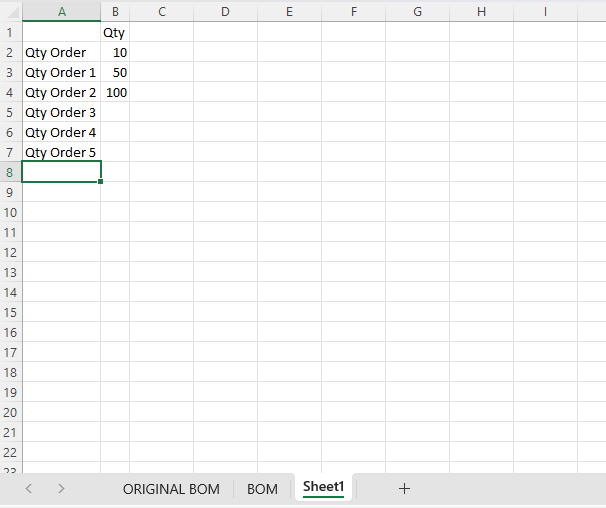


Figure : Adding additional qty orders in Sheet1

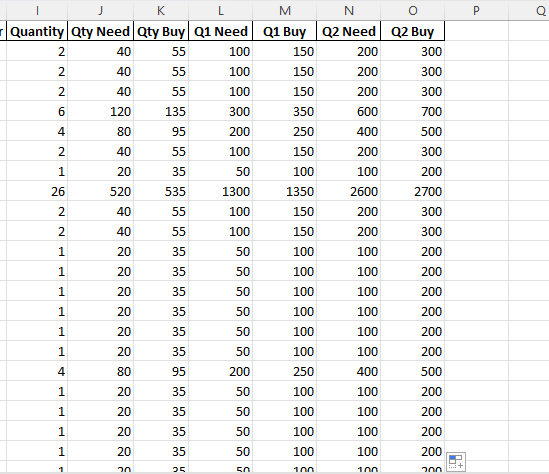


Figure : adding additional qty orders to BOM sheet

1. Save your BOM file with the changes from Steps 1-6. You are now ready to upload the BOM file to the BOM Quoter.

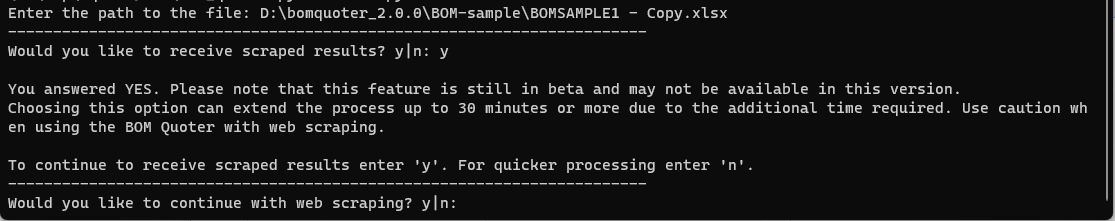
## 4.2 Uploading BOM

1. After updating your BOM file, open the command line (cmd.exe) and open the project directory on your local machine.
2. Go the directory where main.py is and run the program using the command: `python main.py`. Make sure you are running the program in a virtual environment.
3. After running the program, you will be prompted to enter the BOM file path. Copy the path and paste it into the prompt.

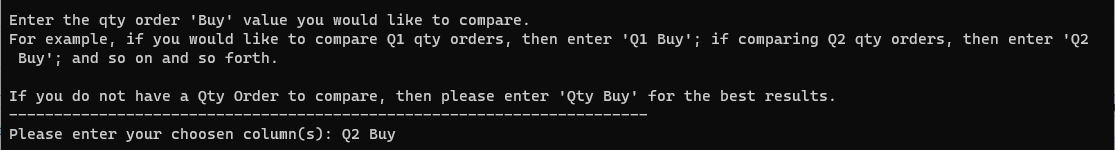


1. Follow the prompts as follows.

For web scraped results:



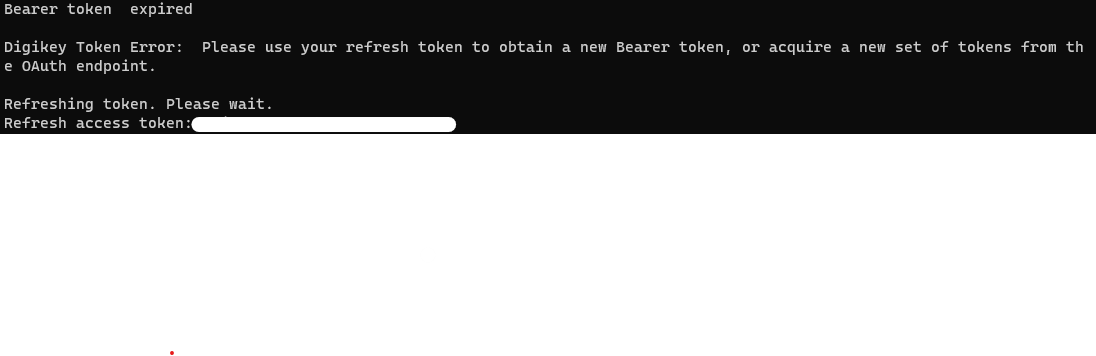
Qty Order prompt:



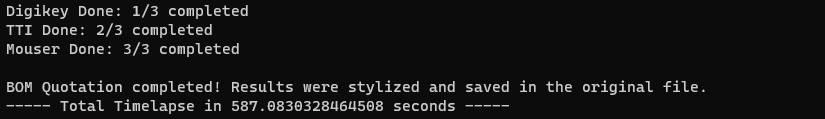
BOM processing:



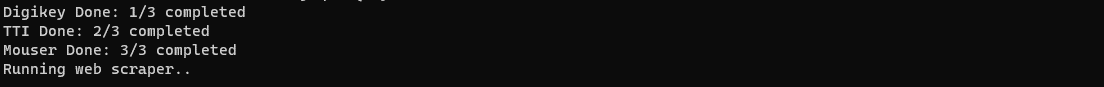
If Digikey token is expired, the program will refresh and replace the token:

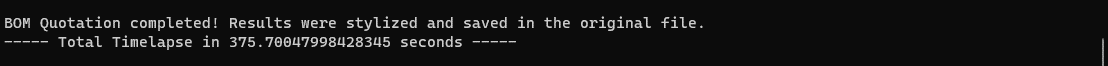


1. BOM Quoter success (without scraping):



BOM Quoter success (with scraping):





# Issues and Challenges

The following are known issues and their temporary solutions:

|  |  |  |
| --- | --- | --- |
| Date | Issue Description | Temporary Fix (if applicable) |
| 8/2/23 | Daily Rate Limit Per Minute Exceed error with code 403 from Mouser when two or more users are using the program |  |
| 8/2/23 | Latest Chrome version 115 is not stable. | Using chrome version < 114 to test the program. |
| 8/2/23 | Inconsistent errors with ElementNotInteractableException: Message: element not interactable | The error can be reduced by adding more wait time when doing web scraping. |
| 8/2/23 | ConnectionError: ('Connection aborted.', ConnectionResetError(10054, 'An existing connection was forcibly closed by the remote host', None, 10054, None)) |  |
| 8/2/23 | ConnectionError: HTTPSConnectionPool(host = 'api.github.com', port=443): Max retries exceeded | Add ‘verify=False’ in the parameter of request operation to reduce the HTTPSConnectionPool related error. |
| 8/2/23 | ReadTimeout: HTTPSConnectionPool(host='api.mouser.com', port=443): Read timed out. (read timeout=None) | The error can be fix by adding number of second to timeout variable in the API request, but this will make TTI API requests more prone to connection errors. |

# Future Development and Improvements

The BOM Quoter team has identified several potential areas for future feature development and improvements. These include:

* Optimize fixed\_BOM code using data structure to manage mounting type, package/case and terminals data flows to improve its latency
* More testing to identify, resolve bugs and errors for better throughput
* Improve the GUI to use as desktop or web app
* Optimize the use of scheduling and parallel programing to decrease the wait time of web scraping
* Setup internal database to save data collected from suppliers API, for faster access and compare when need to find substitute part or check on our own stocks
* Automate total materials and labor cost calculation

# Conclusion

The Concisys BOM Quoter prototype, which is currently undergoing rigorous testing, represents a significant milestone in our journey to create a reliable, robust, and lightweight backup solution to our existing ERP web application. Throughout development, we have strived to address the critical need for a dependable contingency plan in order to ensure the continuity of our core operations

Our next step is to have rigorous testing on the user-end to identify real-world user pain points when using either this program or the ERP. Moving forward, the insights gained from the prototyping phase will play a pivotal role in further refining and optimizing the backup solution to the Concisys ERP. The feedback and observations gathered from testers and end-users will enable us to tailor the prototype to meet the specific requirements of our organization.

The development of this backup solution aligns with our commitment to delivering a seamless and reliable experience to our users. With this innovative system in place, we can confidently mitigate potential risks and bolster our software application's resilience against any unexpected disruptions.

In the coming phases, we will focus on addressing the valuable insights gathered during testing, continuing to enhance the user experience, and ensuring that the backup system seamlessly integrates with our primary software application. Together, these efforts will culminate in a complete and comprehensive backup solution that reinforces the stability and continuity of our critical operations.