To use the FastAPI application with the endpoints you've created, follow this workflow:

1. Activate the virtual environment: source FastAPI/myenv/bin/activate
2. Navigate to FastAPI folder: cd FastAPI
3. Run the app: uvicorn run:app --reload --log-level debug
4. Access the swaggerUI url: <http://127.0.0.1:8000/docs>
5. Data parsing

Example dataset:

A screenshot of a computer

Description automatically generated

Unigene is a database maintained by the National Center for Biotechnology Information for experimental research, which provided a set of non-redundant sequences of genes or expressed sequence tages (ESTs)

The Unigene column in the dataset contain a list of identifiers, seperated by the pip charactor "|", which would split into individual genes to create a set.

Each identifier in the Unigene column represents a unique cluster of gene expression or sequence data:

each ID corresponds to a particular set of gene expression or sequence data that is considered unique within the database.

Data are grouped based on sequence similarity, which implies that they may represent that same gene or similar genes.

The IDs point to a collection of sequences from mRNA and ESTs that appear to come from the same transcription locus(gene or genomic region)

Researchers and bioinformaticians use these identifiers to access data, perform comparative analyses, and elucidate the functions and relationships of genes across different species.

To perform Boolean Algebra analysis on the uploaded dataset using the Boolean Algebra tool:

1. Parsing the data: parse the data from the text file into a format that can be processed by the Boolean Algebra tool.
2. Reading the file, extracting the relevant columns, and converting them into sets of geneset
3. Converting the set datatype to json in order to be stored in the SQLite database
4. Boolean Algebra analysis will be performed on the Unigene column data, as the set of genes for each GeneWeaver ID.
5. Prepare "BooleanAlgebraInput" instances for each Boolean algebra operation will be performed.
6. This will include specifying the operation type (Union, intersection, difference) and provide the list of gene sets to process
7. Running the analysis: instance the Boolean algebra tool, perform the analysis and return the result.
8. a .db file will be created automatically after run the app, which can be opened in the SQLite database.
9. Run the tests: python -m unittest test\_crud.py

A screenshot of a computer

Description automatically generated

We are creating three tables that stored in the database:

Genesets table: store the input dataset in the database

A screenshot of a computer

Description automatically generatedAnalysisRun table: store the analysis run ID, run status,start time, end time

A screenshot of a computer

Description automatically generated

The AnalysisResult table, used to store the analysis result for each run

A screenshot of a computer

Description automatically generated

From Genesets to AnalysisRun:

When a new analysis task is initiated, the application will reference the Genesets table to retrieve the necessary data. It will then create a new entry in the AnalysisRun table to track the analysis process. The new entry will include a unique run ID and the status will be set to 'pending' or 'running'.

Running the Analysis:

The analysis task will use the data from the Genesets table to perform the required computations or processing. Throughout this task, updates on progress can be recorded in the AnalysisRun table by updating the run status.

Storing Results:

Once the analysis is complete, the results are stored in the AnalysisResult table. Each entry in this table is linked to a specific run ID from the AnalysisRun table, ensuring that results can be traced back to their respective analysis tasks.

End of Run Updates: The AnalysisRun table is updated with the 'end time' of the run, and the run status is changed to 'completed'. If there were any issues, the status could reflect an 'error' state, and additional error information could be stored if necessary.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Get geneset by geneweaver\_id:

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Why you were interested in this project

Science background integrated with CS application

This project democratizes access to powerful gene set analysis

This API service will encourage collaboration among researchers by simplifying the sharing of gene sets and analysis configurations, thereby fostering a community of innovation.

The ability to get real-time status updates and results as soon as they are available makes the analysis process more transparent and user-friendly.

Running on local machines ensures that users have control over their computational resources

Researchers and bioinformaticians use these identifiers to access data, perform comparative analyses, and study the functions and relationships of genes across different species.

From Genesets to AnalysisRun:

When a new analysis task is initiated, the application will reference the Genesets table to retrieve the necessary data.

It will then create a new entry in the AnalysisRun table to track the analysis process.

The new entry will include a unique run ID and the status will be set to 'pending' or 'running'.

Running the Analysis:

The analysis task will use the data from the Genesets table to perform the required computations or processing.

Throughout this task, updates on progress can be recorded in the AnalysisRun table by updating the run status.

Storing Results:

Once the analysis is complete, the results are stored in the AnalysisResult table. Each entry in this table is linked to a specific run ID from the AnalysisRun table, ensuring that results can be traced back to their respective analysis tasks.

End of Run Updates:

The AnalysisRun table is updated with the 'end time' of the run, and the run status is changed to 'completed'. If there were any issues, the status could reflect an 'error' state, and additional error information could be stored if necessary.