MeSH Mining

University of Connecticut x Boehringer Ingelheim

John Matura, Isaiah Haynes, Megan Sidmore, Ayyan Mumtaz,
Joan Tejera, Nandan Sureshkumar

Table of Contents

Purpose of the Software

Boehringer Ingelheim (BI), a pharmaceutical company with an emphasis on targeted disease treatment, heavily relies on specific gene research to improve treatment and prevention therapies for patients. In a continuously expanding scientific library with rapidly growing and updating gene research, the present challenge is developing a solution toward streamlining the ability to locate research on a specific gene. With a pre-existing database of important genes, the present challenge is parsing through the large database to obtain published information for any requested gene through a set of scientific literature. The library of literature is obtained through Medical Subject Headings (MeSH), a controlled ontology that annotates published PubMed and Medline articles. The goal of the MeSH Mining software is to allow end users to perform queries on a significant dataset of Gene IDs and MeSH terms. In doing so, searches will redirect end users to a shortlisted selection of Gene IDs, MeSH terms, and relevant references linked to PubMed articles based on the query criteria. When using the software, end users can search using one of three criteria:

- Searching the database by gene (via GeneID)
- Searching by term (via MeSH term)
- Searching by multiple genes (via combined pValues)

The MeSH Mining software is intended for employees at BI to locate information about genes and MeSH terms by streamlining the search in an existing database which originally had no such functionality. This both optimizes gene search methodologies for end users and promotes the necessary efficiency toward finding targeted solutions for existing healthcare concerns.

Design of the Software

The MeSH Mining website was built with technologies such as Flask, Streamlit, and Postgres to seamlessly integrate with existing systems at Boehringer Ingelheim. The database is hosted on an AWS server for efficient communication with the front end aspects of the website.

Other points:

- Flask app
- Results and home page
- Backend functions
- Front end development & functions
- Design and end-user considerations

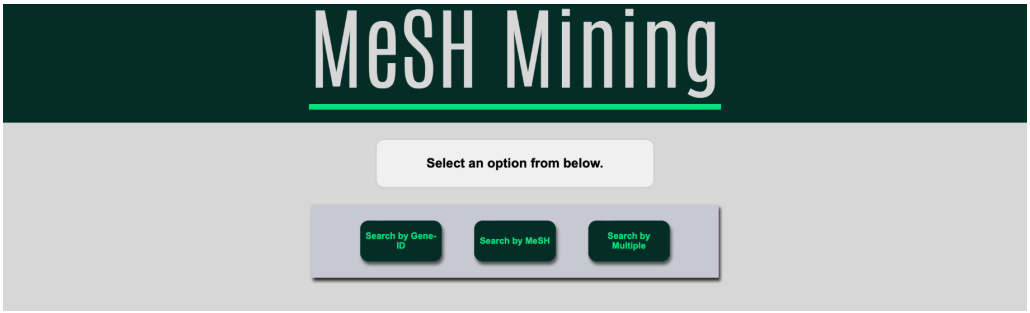
Frontend

The user interface of the MeSH Mining application is built with HTML and CSS in conjunction with JavaScript to assist in overall functionality of frontend objects. The UI consists of two main pages: the home page and the results page, which is shown to users after a search has been executed and successfully returns results. Each page has its own HTML, CSS, and JavaScript files to define the design and functionality of the page, which can be found in the ‘templates’ and ‘static’ folders.

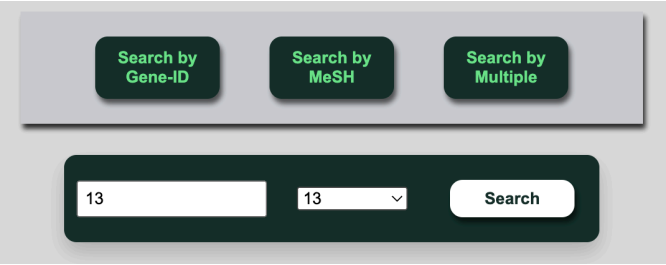
The main page consists of three buttons, one for each of the search criteria. Clicking on any of these buttons will open up a dropdown menu with a search box, a dropdown list with all possible searches, as well as a search button. Users can type directly into the search box to look for a specific gene ID or MeSH term, or they can simply use the dropdown list to see what

searches are available to them and pick from that. Clicking on the search button will initiate the backend queries and functions, and will direct the user to the results page, which will be automatically populated with results from the query.

The frontend was designed such that the color scheme aligns with Boehringer Ingelheim's updated, official colors and is simple enough that any employee who has never used the application before would be able to successfully navigate the interface on the first try.



Main Page



Example Dropdown menu

ID	MeSH Term	pVal	Enrichment	References
13	Rifampin	1.25e-4	123.53	Show articles
13	Alleles	1.33e-4	30.40	Show articles
13	Acetylation	1.92e-4	99.35	Show articles
13	Polymorphism, Single Nucleotide	1.95e-4	26.63	Show articles
13	COS Cells	2.66e-4	84.40	Show articles
13	Genetic Predisposition to Disease	2.91e-4	23.22	Show articles
13	Sequence Deletion	3.66e-4	71.79	Show articles
13	Activation, Metabolic	3.91e-4	2565.01	Show articles

Example Results Page

Functionality of the frontend is largely a combination of HTML, CSS, and javascript working in conjunction with one another, as well as with the backend Python functions, to create a seamless user experience. For example, the buttons on the main page were created using HTML and CSS for their layout and design, but JavaScript functions are controlling the dropdown menus and what happens when a user clicks on the button.

While the front end is the most visible part of the website, there is much more to be said about what is happening behind the scenes of the MeSH Mining application.

Backend

The cornerstone of the backend lies in the establishment of a secure connection to the PostgreSQL database. The code snippet below exemplifies the connection setup:

```
def openConnection() -> type(tuple()):
    try:
        # Establish a connection to the database
        db_config = {
        'host': '*****',
        'database': '*****',
        'user': '*****',
        'password': '*****'
        }

        #Connect to the database with config
        connectionInstance = psycopg2.connect(**db_config)

        # Create a cursor object to interact with the database
        cursor = connectionInstance.cursor()

        # print("Connected to the database!")
        if connectionInstance.status == psycopg2.extensions.STATUS_READY:
            return connectionInstance, cursor

    except Exception as e:
        print(f"Error: Unable to connect to the database - {e}")
        raise
```

Database connection instance function definition

Most Python 3 Libraries follow a similar format for connecting to a database and use a cursor to emulate a user in the execution of queries. See Psycopg 2 docs.

With an established connection, the core functionality of the backend begins as follows. The following code snippets showcase key functions for searching data by GeneID and MeSH:

```
def searchByGene(gene: str) -> tuple[list,str]:
    connection, cursor = openConnection()

    query = """SELECT * FROM \"GENE\" WHERE \"GeneID\" = %s ORDER BY \"p_Value\" ASC;"""

    cursor.execute(query, (gene,))

    output = cursor.fetchall()

    connection.close()
    return list(output), simplejson.dumps(output, indent=2)
```

Query data by "GeneID" function definition

The function which handles search by MeSH functionality is identical save for querying rows by "MESH" column value. These functions: establish a temporary database session, query based on user input. It fetches all query results, closing the database session and finally, returning the data in the form of a list to be used on the website, or in JSON format for exporting.

The backend also supports the retrieval of aggregated data, this is the third use case functionality, search by multiple Gene IDs:

Search by Multiple GeneIDs Query

```
query="""
    SELECT DISTINCT ARRAY_AGG("p_Value")AS pVals,"MeSH",COUNT(DISTINCT "GeneID")AS
    numGenes,ARRAY_AGG("GeneID" ORDER BY A."GeneID")AS listGenes
    FROM "GENE"AS A
    WHERE A."GeneID" IN %s
    GROUP BY A."MeSH"
    ORDER BY 3
    DESC LIMIT %s OFFSET %s;
      """
```

SQL Query for "rolling up" MeSH terms per expected data format

This is the query responsible for searching all rows of MeSH terms which have one or multiple of the given Gene IDs. From there if there are multiple rows with the same MeSH term then they are rolled up meaning array aggregated to be one row where the MeSH is now unique while preserving which Gene ID integer value these MeSH terms were associated with in the form of a new columns which is represented as “List Genes”. In this case this could range from one to all of the searched Gene IDs. From there the only step that needs to be taken into account aside from the other basic querying of genes or meshes is combining p-values which is done with the help of a helper function which implements Fisher’s method for combining p-values, a standard way of combining p-values which is made accessible in Python by Scipy’s Stats module. That new rolled up data can then be returned for use by the website and can be exported in JSON format.

With the core functionality of the backend out of the way there were some more required functions for our implementation of the website such as a function to return all Genes and another for all MeSH terms. These functions are used in our search box in the form of the dropdown that the user sees, if they wish to click and search for a gene or MeSH by name.

```
def listAllMesh() -> list[str]:
    connection, cursor = openConnection()

    cursor.execute(
        """SELECT \"MeSH\" FROM \"GENE\" ORDER BY \"p_Value\" ASC;""")

    data = cursor.fetchall()

    connection.close()
    return [str(i[0]) for i in data]
```

Function to retrieve all rows with distinct MeSH terms, also mirrored for Gene IDs

Database

Creating the database on Amazon RDS marked the pivotal first step in the project, laying the foundation for subsequent development efforts. This crucial step facilitated seamless access to the database for the entire team, enabling collaborative efforts in implementing both the backend and front-end components of the web application. By centralizing the data on a scalable and reliable platform like Amazon RDS, team members could leverage the gene literature reference data to develop and integrate various functionalities into the application. Furthermore, hosting the database on Amazon RDS provided a robust foundation for the project by ensuring high availability. This feature ensured that the database would remain accessible and operational even in the event of hardware failures or maintenance activities, minimizing downtime and ensuring continuous access to critical data. Initially, deploying the PostgreSQL database on Amazon RDS presented a pivotal step, leveraging the platform's robust infrastructure and managed services. This process encompassed configuring the database instance with meticulous attention to detail, including selecting appropriate specifications such as instance type, storage type, and allocated storage capacity. Furthermore, network settings and security groups were meticulously established to govern access, ensuring that only authorized users could interact with the database. This deployment on Amazon RDS not only provided a reliable and scalable foundation for the project but also alleviated the burden of managing infrastructure, allowing the team to focus on developing and implementing the core functionalities of the database. The subsequent phase involved populating the database with gene literature reference data, which was provided by the Boehringer Ingelheim team through a CSV file.

Backend Connection to Frontend

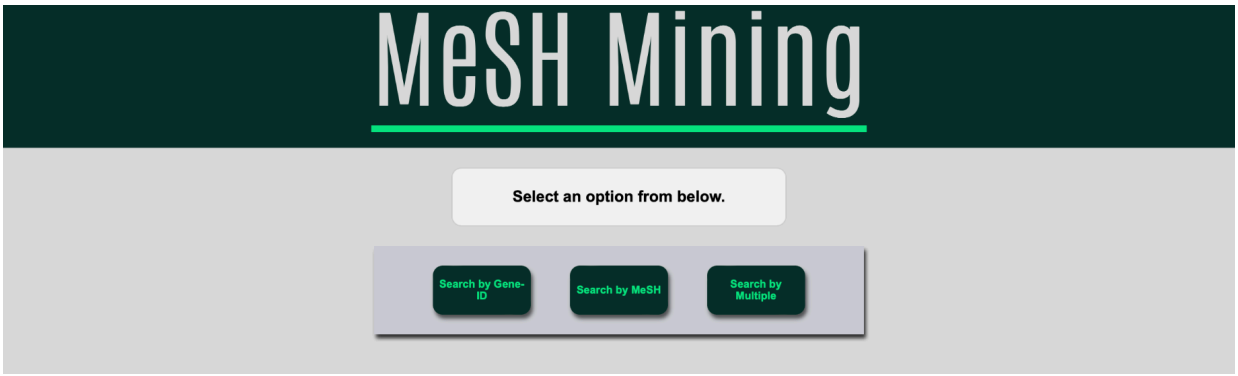
The backend functions are connected to the frontend using a simple application of Flask in python. Each of the main search options (searching by gene ID, MeSH, or multiple gene IDs) has its own API that exposes the backend functions via a url with query strings and headers. An application of this with Search by Gene-ID can be seen in the code snippet below.

```
@app.route('/searchGene', methods=['GET'])
def searchByGene_website():
   geneID = request.args.get('geneID')
   page = request.args.get('page', default=1, type=int)
   per_page = request.args.get('per_page', default=20, type=int)
   sort_by = request.args.get('sort_by', default='p_Value', type=str)
   if geneID is None:
       return jsonify({'error': 'No parameter in request'}), 400
   dbConnection = openConnection()
   response = searchByGene(geneID, dbConnection, page, per_page, sort_by)
   dbConnection.close()
   return jsonify(response)
```

When the search button on the homepage is clicked, it navigates to the results page which connects to the API using the following url (example values populated for query results): “/searchGene?geneID=34&page=1&per_page=20&sort_by=p_Value”. This sends all of the necessary data to the API which then sends that data through to the backend function. Finally, it returns the data from the backend functions to the results page in JSON format where it is parsed and placed into a table to be shown to the user. The remaining two connections function similarly with the searchByMesh function also accepting data sent via an API header instead of only query strings.

Functionalities

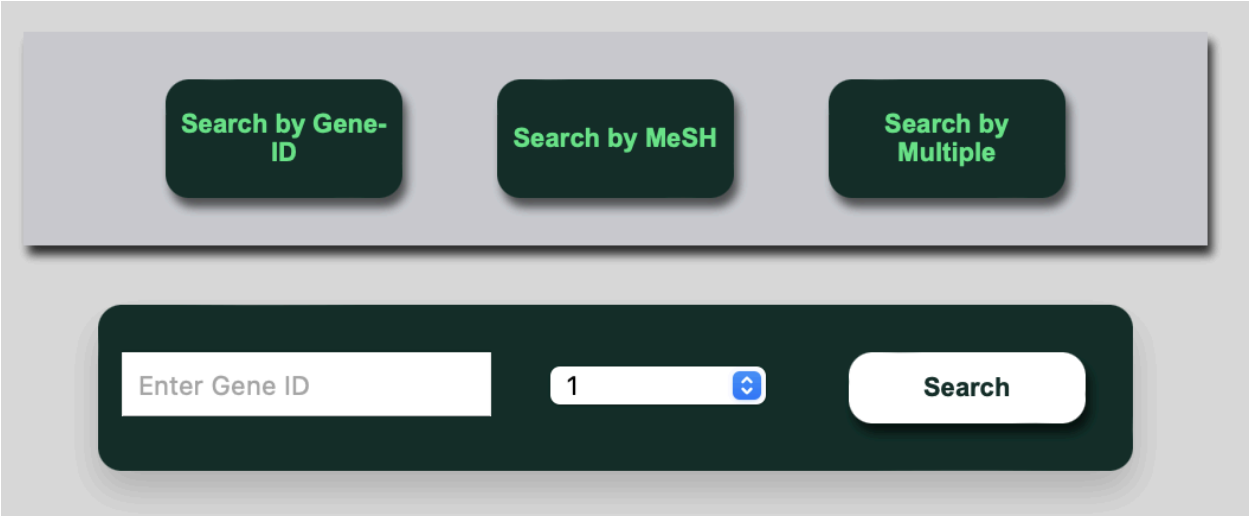
This program has three main functionalities: searching by a gene id, searching by a MeSH term, and searching by a list of genes. Each of these functionalities are easily accessible from the homepage by clicking on one of the three buttons shown in the figure below.



MeSH Mining Homepage

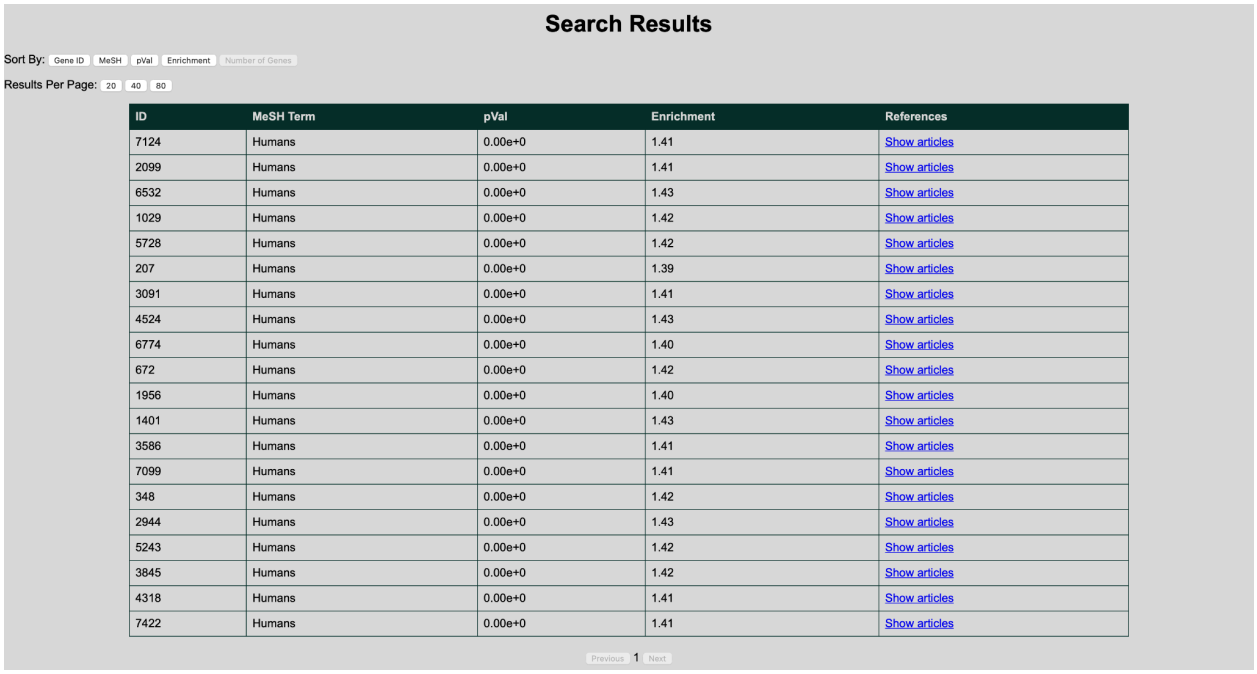
Clicking on the *Search by Gene-ID* button will show the user a search bar, a dropdown list, and a search button. A gene ID can either be inputted manually into the search bar or can be searched for using the dropdown list of all gene IDs in the database. Clicking on an ID from the dropdown list will automatically fill the search bar with that ID or replace any text already in the search bar with the ID that was chosen. Clicking on the button labeled *Search* then takes the user to the results page using the term from the search bar input. The *Search by MeSH* button also shows the user a search bar, a dropdown list, and a search button. However, instead of containing gene IDs in the dropdown list, this list contains all of the MeSH terms that are in the database. Clicking on the search button once again will take the user to the results page. The final main functionality is searching by multiple genes, accomplished by clicking on the *Search by Multiple* button. This option is most similar to the *Search by Gene-ID* option including a search bar,

search button, and the same dropdown list of all of the gene IDs. In order to search for multiple gene IDs, the IDs have to be formatted as a comma separated list (e.g. 2,9,14) in the search bar. Once again, the input can be typed either manually or through the dropdown list. Clicking on a gene ID from the dropdown list should automatically add each gene to the search bar in its proper format. Finally, clicking *Search*, will take the user to the results page.



Options for Search by Gene-ID

On the results page, data retrieved from the search term(s) are displayed to the user as a table. If the search term was a single gene ID or MeSH term, this table will have columns for gene ID, MeSH term, p-Value, Enrichment number, and references. If the search option was multiple gene IDs, the table will display columns for the combined p-values, the MeSH term, the number of genes for that specific row, and a list of gene IDs from the inputted list that correspond to that MeSH term. Images of these two tables can be seen below.



Search Results

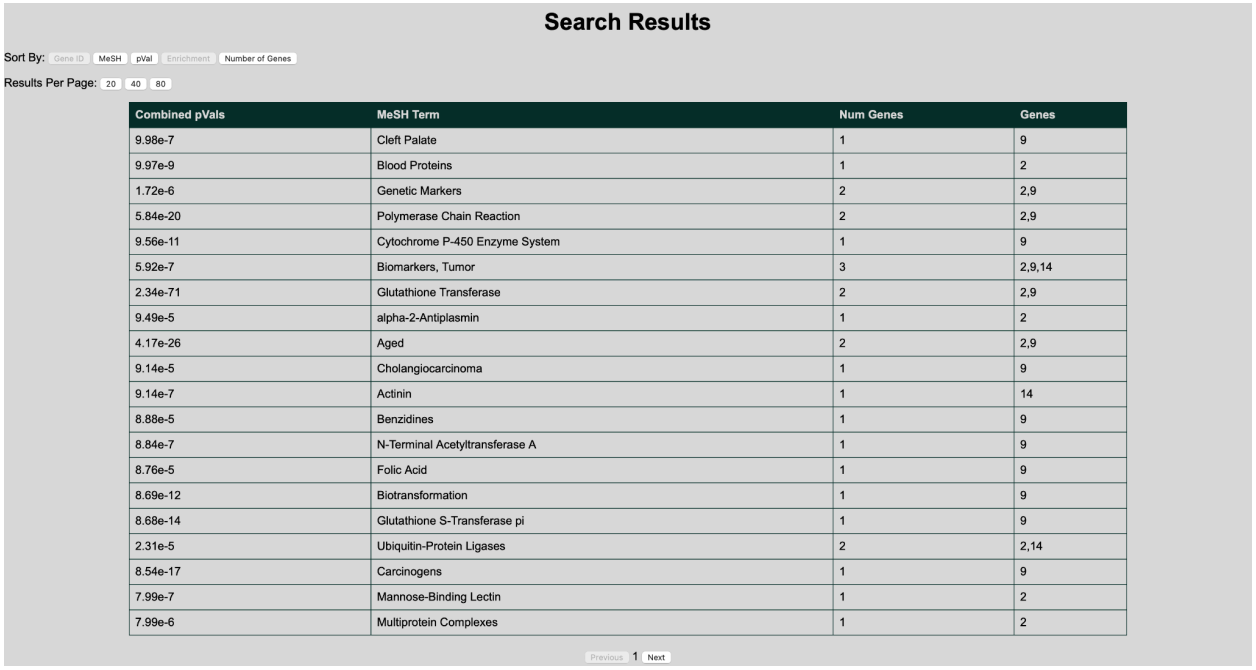
Sort By: Gene ID | MeSH | pVal | Enrichment | Number of Genes

Results Per Page: 20 | 40 | 80

ID	MeSH Term	pVal	Enrichment	References
7124	Humans	0.00e+0	1.41	Show articles
2099	Humans	0.00e+0	1.41	Show articles
6532	Humans	0.00e+0	1.43	Show articles
1029	Humans	0.00e+0	1.42	Show articles
5728	Humans	0.00e+0	1.42	Show articles
207	Humans	0.00e+0	1.39	Show articles
3091	Humans	0.00e+0	1.41	Show articles
4524	Humans	0.00e+0	1.43	Show articles
6774	Humans	0.00e+0	1.40	Show articles
672	Humans	0.00e+0	1.42	Show articles
1956	Humans	0.00e+0	1.40	Show articles
1401	Humans	0.00e+0	1.43	Show articles
3586	Humans	0.00e+0	1.41	Show articles
7099	Humans	0.00e+0	1.41	Show articles
348	Humans	0.00e+0	1.42	Show articles
2944	Humans	0.00e+0	1.43	Show articles
5243	Humans	0.00e+0	1.42	Show articles
3845	Humans	0.00e+0	1.42	Show articles
4318	Humans	0.00e+0	1.41	Show articles
7422	Humans	0.00e+0	1.41	Show articles

Previous 1 Next

Example Results page for Search by MeSH



Search Results

Sort By: Gene ID | MeSH | pVal | Enrichment | Number of Genes

Results Per Page: 20 | 40 | 80

Combined pVals	MeSH Term	Num Genes	Genes
9.98e-7	Cleft Palate	1	9
9.97e-9	Blood Proteins	1	2
1.72e-6	Genetic Markers	2	2,9
5.84e-20	Polymerase Chain Reaction	2	2,9
9.56e-11	Cytochrome P-450 Enzyme System	1	9
5.92e-7	Biomarkers, Tumor	3	2,9,14
2.34e-71	Glutathione Transferase	2	2,9
9.49e-5	alpha-2-Antiplasmin	1	2
4.17e-26	Aged	2	2,9
9.14e-5	Cholangiocarcinoma	1	9
9.14e-7	Actinin	1	14
8.88e-5	Benzidines	1	9
8.84e-7	N-Terminal Acetyltransferase A	1	9
8.76e-5	Folic Acid	1	9
8.69e-12	Biotransformation	1	9
8.68e-14	Glutathione S-Transferase pi	1	9
2.31e-5	Ubiquitin-Protein Ligases	2	2,14
8.54e-17	Carcinogens	1	9
7.99e-7	Mannose-Binding Lectin	1	2
7.99e-6	Multiprotein Complexes	1	2

Previous 1 Next

Example results page for Search by Multiple

As can be seen in the images above, the results page also includes options to sort the results and how many results to display per page. Depending on which search option was used, one or more of the sort options may be disabled. Clicking on any of the sort options should reload the current page with the new sorting option. The new sort option should also be retained

for each new page that the user goes to. It is important to note that the sort options from searching by multiple gene IDs are sorted in descending order, sorting by MeSH Term will start with '*Z*' and go down to '*A*'. The number of results per page defaults to 20, but 40 and 80 results per page can be chosen by using the options provided. Clicking on *show articles* in the references column shown when searching for a gene ID or MeSH term will open a new tab with the related PubMed article(s) found on the National Library of Medicine website. A "hidden" feature on the results page is that clicking on the *MeSH Mining* header will return the user to the homepage.

Use Case Scenarios

The use case scenarios of this program are highly tailored to the sponsors needs, workflow and research process. In no small part due to sponsor efforts compiling the PubMed data for this project, this software simply allows a user to quickly find articles associated with a gene, a mesh keyword or multiple genes. This arose out of a need to verify if a gene being researched had been done so in a relevant experiment. This is because not all research into a gene correlates to research interest of the end user, hence the importance of article titles (MeSH terms) to determine if specific avenues of research into a gene had been done before. That being said, while a useful tool xas to not repeat redundant research, the usefulness of this software extends to any case in which a user needs to find data or articles relating to one of the search criteria mentioned above. For example this could be instead of using it to verify that specific research hadn't been done, it could be used to expand upon existing experiments. It is in the same vein

that at this time there are no limitations of the software, we query the data and display it in the way that it was gathered from PubMed.

A specific use case could be for researchers at pharmaceutical companies engaged in drug development. It would help them identify genes and their association with diseases through MeSH terms. By understanding and searching Gene-Mesh pairs, his information can be crucial in understanding gene-disease relationships, and thus integral in the early stages of drug discovery. It could help to treat specific genetic anomalies or to mitigate side effects in populations with certain genetic profiles. With the inclusion of statistical measures like combined p-value and enrichment, which they can sort by, the researchers involved can make value-based judgements on these gene-mesh pairings and find what they deem to be statistically significant or perhaps anomalous, which could help them prioritize research and drug development efforts.

Another specific use case could be detailed genetic risk profiling. The application enables you to query specific genes, and specifically look for hereditary diseases indicated by MeSH terms, and also understand the statistical significance of these associations. This could perhaps allow detailed risk assessments to be created based on genetic profiles, which could help predict likelihood of developing certain genetic conditions. It could be very useful in helping healthcare providers advise patients on proactive approaches, perhaps even as far as allowing prospective parents to be informed on the genetic risks of inherited conditions, which could guide decision making in family planning.

Limitations of the Current Implementation

Limitations of the current implementation include both scalability in terms of untested performance against heavy user traffic scenarios. While the application may function adequately under typical usage conditions there has not been testing and validation for handling multiple users simultaneously. By conducting thorough load testing, developers can identify performance bottlenecks as well as optimize resource utilization, and implement relevant scalability measures. Additionally, implementing monitoring and alerting mechanisms can help detect and mitigate performance issues in real-time, ensuring that the application remains responsive and available even during periods of high user traffic. The current database structure presents several limitations that hinder data organization and retrieval efficiency. With only one table named "gene" and a single primary key counting each record, the database lacks the necessary normalization to effectively manage its contents. This structure, comprising five columns—gene ID, MeSH, p-value, enrichment, and PMIDs—doesn’t adequately represent the relationships between genes and their associated MeSH terms. Furthermore, the presence of duplicate gene IDs with differing MeSH values causes issues of data redundancy and compromises data integrity. Currently, all data is consolidated within a single table, which not only introduces redundancy but also makes the data retrieval processes complicated due to unnecessary duplication. Moreover, addressing these limitations to the database's functionality therefore a restructuring process is imperative. Introducing normalization principles will help the organization of data into multiple tables, with each table representing distinct entities and relationships. Specifically, the gene and MeSH data should be separated into separate tables, allowing for more efficient data management and retrieval processes. The application of

normalization rules, including the first, second, and third normal forms, facilitates the elimination of redundant data and ensures data integrity. Specifically, the first normal form (1NF) guarantees attribute atomicity and removes repeating groups, while the second normal form (2NF) addresses partial dependencies by breaking down tables into more granular entities. Lastly, the third normal form (3NF) eliminates transitive dependencies by further decomposing tables and establishing direct attribute dependencies on the primary key. Implementing these changes will not only enhance data integrity and organization but also optimize data retrieval processes and support more advanced querying capabilities.

Another critical limitation is the absence of automatic updates in the database, which does not synchronize autonomously with the data sources. That means we have to deal with presenting outdated information, which can significantly impact the utility and reliability of research outcomes. Relying on manual updates, not only increases the potential for error, but also demands substantial maintenance efforts. Considering that our search functionality is designed to be a tool in scientific pursuit, in fields such as medicine, it is especially significant that the data is reflective of the latest scientific advancements. Medical research is a fast-evolving field, and there may be all sorts of new discoveries and updates. If a researcher uses this application as a tool to develop hypotheses or design experiments on outdated information, they could end up with a suboptimal approach. They could end up with a misguided research direction, as newer studies may have made the insights from the old data redundant. Research is all about innovation, building upon other data to make informed decisions. With outdated data, this application as a research tool lacks credibility and trust.

Though you could use the tool in an up-to-date manner, if you use the manual approach of populating the table that we used to create our results, it is limited in realistic use-cases. You

would have to use scripts and repopulate the postgresql database every time you wanted accurate results. Additionally, there would be an increased error potential. Accounting for human error, you could potentially collect the wrong data by messing up on your script. Or make a mistake in populating the table, and then have inconsistencies in your table structure. Accounting for scalability, this becomes an even bigger problem as multiple researchers manually handling the data increases the potential for error in it being accurately represented. As you scale up, it becomes a bigger problem. With a big team of researchers, who knows how many errors can be made. It causes the application to not be credible or trustworthy, in the empirical world of research, where data has to be up-to-date and accurate for innovation.

On a related note of scalability, there are important scaling limitations when it comes to our database size, as it relates to some of our frontend components. For example, we have a dropdown list for Gene ID and MeSH term searches, which is already a bit hard to navigate because of the number of both, and will become increasingly so, as the database scales. Another scalability limitation is the utility of our searches themselves. We did implement helpful features such as sorting, but the end-user experience of interpreting and utilizing results is going to become increasingly difficult with larger datasets. A more user-centric design with advanced filtering and search capabilities, with new relevant parameters would help make it manageable as it scales.

Possible Future Improvements

In improving current efforts for the MeSH Mining software, as well as looking forward toward further implementation and redesign, there is much that can still be done to greater streamline the process and obtain more targeted information that caters more directly to what the end user is looking to find. There were a few features requested from the sponsor that we were not able to implement for various reasons, either due to their complexity, lack of time in the academic year, or bandwidth from the project team. These features could be added in the future, including filtering MeSH terms by major categories, using Human MeSH matches only, or searching by other species.

In terms of features that were not requested or previously considered, the current software allows users to search through the entire database, but does not have filtering capabilities in either the search page or results page aside from the given sorting features. For both efficiency and user-friendly purposes, it seems reasonable that a future development would allow for more specific search queries or better filtering through the redirected results. This is especially beneficial for search cases that return a lot of data and require multiple pages for results, as manually searching through this content can get both tiring and time-consuming. One implementation would be to specify and enhance the three query criteria. For example, if searching for research on a gene with a Gene ID 17, implementing a “keyword” optional search box may allow for end users to enter other terms, such as the name of a chronic condition, and therefore specify that they are specifically searching for results with a gene ID of 17 and how it is affected by a certain chronic disease. On the results page, results can be filtered in innumerable ways. However, one way the results could be filtered is using the associated PubMed articles,

prioritizing results by either date of release (newly or oldest added) or by more popular and trending articles (determined through the PubMed website). As optimization and efficiency has been a primary goal throughout the MeSH Mining development, this feature could be considered a high priority item in future development.

Beyond search functionality, a future improvement could involve enhancing the user interface Perhaps bookmarks for frequently accessed searches, customizable views, and interactive data visualization would make it easier to navigate the application. Bookmarks would streamline access to common queries, customizable views would allow users to adjust how information is presented based on their preferences, and interactive visualizations would aid in the easier comprehension of complex data. These features collectively improve navigation and engagement, making the research process more efficient and enjoyable.

A really critical future improvement is keeping the application current with the latest data. Especially in fields like medicine where new discoveries are made rapidly. Implementing automatic updates to our database from trusted sources such as Gene, PubMed, and MeSH would ensure that the application reflects the most current scientific data, which is important for its credibility and reliability as a research tool. In fields such as medicine, where our tool could be applicable, it could be a big hindrance. A researcher could make a development of hypotheses or give their attention towards certain research topics by looking at gene-mesh pairings, and it could ultimately end up being a misdirected pursuit because the observation used outdated data. It would also greatly reduce the potential of error if these databases were being updated automatically, without manual population.

Implementing the feature to be able to search by a different species could actually prove very beneficial, both to MeSH mining and organization goals. Part of BI's research and

innovative solutions include those catered to animal health products and veterinary medicines and solutions, so expanding the criteria to include content for species beside humans can be optimal for those working in that division of research. MeSH terms and Gene IDs are not always just associated with humans, and researchers or BI employees may want to be able to see results related to a specific species they are trying to gather information on. Searching per different species would definitely involve changes in both front and backend functions to be able to filter out the database around species. This improvement is near the same capabilities of filtering, and could likely even be implemented alongside the filtering capability previously mentioned by having a filter for species alongside filtering by MeSH term categories.

Furthermore, given the great size of the database and query results, a potential consideration may be to include a feature that tracks the user interactions with the MeSH mining software, ultimately saving and storing recent searches and previously browsed search results. These results can be stored for a certain amount of time (e.g., per search session or up to 30 days), or for a certain number of results (e.g., saving the most recent 15 queries or browsed information). A common issue in research and in searching a large database of results is keeping track of information without browsing content that has already been reviewed. Developing this additional feature would allow end users to keep track of their research and prevent the oftentimes time consuming and repetitive behavior of revisiting literature for needed information or data.