Ayyan Mumtaz

This project, sponsored by Boehringer Ingelheim, was directed towards developing a web application, by the name of ‘MeSH mining’, capable of querying and retrieving gene-related data from extensive literature. The primary function of our application is to enable users to perform searches by gene using GeneID, by medical subject headings (MeSH terms), or by multiple genes, where gene IDs are separated by commas, to observe the relationships between gene and MeSH terms. MeSH Mining presents search results in a well-organized, paginated table. The columns displayed from left to right are: Gene ID, Mesh, pVal, Enrichment, and Show References, which include links to the respective PubMed articles referenced. The design of our table reflects critical aspects of statistical analysis and biological relevance: the pVal column indicates the statistical significance of the findings, where a lower p-value suggests a stronger significance of the observed difference, and the enrichment column, calculated via Fisher’s exact test, compares the p-value to the standard deviation, helping highlight the strength of gene-term associations.

There are two main components of my contribution. I am going to start with the first contribution, my transformation of our project from a rudimentary script that allows you to return disorganized plaintext, to a full stack web application, which I accomplished in a single Github commit. These contributions fundamentally transformed the project from a basic script delivering unstructured plaintext into a comprehensive, user-friendly, and interactive full-stack

web application. There were many components to accomplishing this, it was basically a retooling of our entire codebase, and I will try to go over the important parts.

To start, the integration between the frontend and the backend required a lot of work. I had to establish clear API endpoints for each function (e.g., `searchByGene`, `searchByMesh`) which allows the frontend to make HTTP requests to these endpoints. On the HTML front, in our frontpage file. I included input fields and search buttons within dropdowns, facilitating direct interaction with backend APIs for searching by Gene ID, MeSH Term, or multiple Gene IDs. This is a shift towards a more interactive, user-initiated search process.

A ton of logic had to be figured out on the backend as well. Our old search functions returned all results at once. I added the additional parameters `page` and `per_page`, and returned results for the specified page. I used SQL's `LIMIT` and `OFFSET` clauses, which improves visibility by displaying our disorganized results in a paginated format, but doubles as crucial for handling large datasets without overwhelming the user or the system. Since we wanted to organize the text, the functions could not just return results as tuples like it did before. Now each function has a dictionary containing not just the results but also metadata like total records, current page, items per page, and total pages. . This structured format makes it easier to handle in the frontend for displaying data in tables or other UI elements.

Making our website dynamic and interactive informed my decision to strongly reconfigure our Javascript functions. JavaScript is essential for delivering a dynamic user experience, enabling on-page elements to update in real-time based on user actions without requiring a full page refresh. For instance, when using the search function, it's necessary to show

a dropdown menu that reacts to user input. Additionally, JavaScript leverages its event-driven architecture to handle user interactions like button clicks and form submissions. This is exemplified by event listeners attached to search buttons, which initiate AJAX calls to retrieve data based on user inputs. The data retrieved is then processed and seamlessly added to the web page.

Moreover, JavaScript is vital for managing the application's state and navigation. Regarding pagination, JavaScript can adjust URL parameters to preserve state across various user actions, handle user events, and dynamically refresh content, making it indispensable for the Gene-to-MeSH project. This capability allows the application to be responsive, ensuring a smooth experience for users querying genetic data. For example, our results page, which had to be completely retooled, has previous and next button clicks controlled by JavaScript, which listens to them, fetches the corresponding page data, and updates the table content and current page number accordingly.

In summary, the last state of the project before my contribution had the basic search functionalities, but it was far from being a web application, requiring manual database querying via curl commands. I revamped the entire codebase, pushing it towards our goal of a more dynamic and user-centric application.

My other contribution, an idea that stemmed from the early ‘completeness’ of our project, was a bit extracurricular.We fully completed the directive assigned to us from Boehringer Ingelheim, so when considering what else was left to do, I decided it would be interesting to see if we could create a visualizer for the search results. Our database is so large, and I figured perhaps it could help an end user interpret information if they could see it before them.

I created a network graph that effectively presents the search results. At its center is a node representing either the gene ID or MeSH term being searched. Surrounding it are peripheral nodes representing MeSH terms associated with the gene, if on the MeSH term results page, or gene IDs associated with the MeSH term, if on the gene ID results page. Edges connect the central node to each of these peripheral nodes, and their weight (thickness) is inversely proportional to the p-value. A lower p-value, indicating a stronger association, results in a thicker edge, giving users a visual cue about the strength of the association.

The graph utilizes a force-directed layout, naturally positioning nodes with stronger associations closer to the central node. In this way, higher enrichment translates to a shorter distance between nodes, while lower enrichment spreads them farther apart. This makes sense because the enrichment highlights the strength of gene-term associations, and it is naturally intuitive that the farther the gene and term are, the weaker that association. Additionally, hovering over nodes displays their names, providing quick identification of the nodes and enhancing the user experience.

This contribution ended up being a lot of work, it might have been less code than the other things I have done, but it took a lot more careful consideration. Both pVal and enrichment have great variance in our data, so what I did was make them conform to a standard normal distribution relative to each gene and MeSH term search. So all the values are relativized to the specific search that their data belongs to. Thus, the network graph would look coherent, and

though it’s not indicative of the absolute values it is visually interpretable and appealing.

Tailored to the sponsor's workflow and research needs, the use case for this software allows users to swiftly identify articles related to genes, MeSH keywords, or multiple genes, based on data gathered from PubMed. It's particularly valuable for verifying prior research, preventing redundancy, and expanding on existing experiments. Researchers in drug development can use the tool to uncover gene-disease associations through Gene-MeSH pairs, providing critical insights for early drug discovery. It helps them evaluate genetic anomalies or side effects with statistical measures like p-value and enrichment, enabling data-driven prioritization of research. Additionally, this software can help medical professionals analyze hereditary diseases linked to specific genes and their statistical relevance, thus helping healthcare providers advise patients proactively.

The limitations of our current implementations are plentiful. The database's lack of automatic updates significantly hinders the utility of this application, as outdated information diminishes the credibility of research outcomes, particularly in rapidly evolving fields like medicine. Manual updates are cumbersome, prone to error, and require substantial maintenance, which limits their practical application in the long run. The scalability challenges of managing increasing data volumes exacerbate these issues. Current search tools, such as dropdown lists for Gene IDs and MeSH terms, are difficult to navigate and become unmanageable as the database grows. Without accurate and timely data, researchers risk following outdated information that may no longer be relevant, potentially compromising the direction of their research. To maintain trustworthiness and effectively support scientific endeavors, a user-friendly design with

improved filtering and search capabilities is essential to make sense of the growing data and keep the information current. Understanding these flaws will help our sponsor create a better tool in the future.