

**Name:** Dien Mach  
**UNIX Username:** dmach1

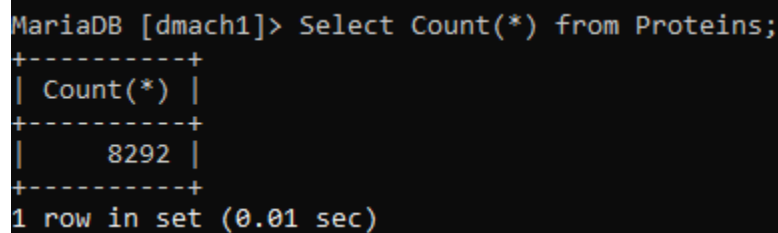
## Writeup for the Human Body Protein Search

### Tool Background

With an educational background in biochemistry and a passion for physical activities and exercise, I created a search engine for proteins and enzymes in the human body. When browsing the Human Metabolome Database website, I found the page layout for each protein to be outdated and cluttered. By creating this search engine, I reformed the web page to be more minimalistic, modern, and intuitive. As a result, the website is intended to help researchers, students, or any other user find information quicker about proteins or enzymes. Hopefully, the tool helps with biological research, education, and other use cases.

### About Tool and Implementation

The tool is a web-based data mining tool that allows users to easily search for information about proteins or enzymes found within the human body. The protein source data was obtained from the <https://hmdb.ca/> website in the form of an XML file [1]. There were 8292 protein entries in the file with related data for each protein. Using Python, the lxml library, and MySQL.connector module, I created a series of Python scripts found within the `./adv_pcc_final/final/database_scripts` directory to parse the source data and upload the protein entries and their related information onto MySQL. For more information about the relational database, under the 'schema' tab on the website, there is a schema diagram that displays the entity relationships between each table.



```
MariaDB [dmach1]> Select Count(*) from Proteins;
+-----+
| Count(*) |
+-----+
|      8292 |
+-----+
1 row in set (0.01 sec)
```

**Figure 1:** Screenshot displaying the total amount of protein entries.

**Name:** Dien Mach

**UNIX Username:** dmach1

```
MariaDB [dmach1]> show tables;
+-----+
| Tables_in_dmach1 |
+-----+
| Additional_Info   |
| Biological_Functions |
| Gene_Properties   |
| Go_Classifications |
| Literature_References |
| Metabolites       |
| Pathways          |
| Protein_Properties |
| Proteins          |
+-----+
9 rows in set (0.00 sec)
```

**Figure 2:** Screenshot displaying the 9 tables uploaded to MySQL.

After uploading the entries onto MySQL, I developed a user-friendly front end using a CSS/HTML/JavaScript/Python CGI stack. The HTML provides the structural elements for each page such as the front, about, schema, and writeup page. Meanwhile, the CSS provides styling guidelines for each element, which provides the aesthetics of the page. JavaScript modules were written to dynamically change HTML elements and CSS styling on the page based on user inputs such as clicking buttons, searching for proteins, hovering over elements, etc. Lastly, JavaScript would make AJAX calls to the Python CGI scripts which would make protein queries via the MySQL server and return the information in JSON format to the respective JavaScript module. As a result, a user-friendly web page was achieved and is described further below.

Users are greeted with a search bar and a header bar containing the 'about', 'schema', and 'writeup' sections for more information (Figure 3). Upon searching for a protein, an initial table appears containing all matching results and their corresponding metadata (Figure 4). Clicking on one of the protein entries brings you to the page shown in Figure 5. Here, users can click on one of the blue buttons which provides categorical information about the selected protein.

### Implementation Issues

Originally, I was going to parse the metabolite source data rather than the protein source data found on the HMDB website. However, I could not parse the file due to non-UTF-8 symbols found in the data. As the file was 6 gigabytes large, I could not find and delete the symbols nor could I debug it. As a result, I used the protein source data, which parsed just fine via the lxml library.

Most of the implementation issues and frustration came from the front-end stack. CSS/HTML/JavaScript are not my strongest languages as I have just learned them this semester. Figuring out how to debug the webpage was the most difficult part. It was not as straightforward to me as debugging issues in PyCharm through the PyCharm IDE. Through

**Name:** Dien Mach

**UNIX Username:** dmach1

YouTube tutorials, StackOverflow, official documentation, usage of the WebStorm IDE, web tools, many Google searches, and Google Chrome's built-in browser tools, I was able to debug many of the layout issues. Moreover, I learned to effectively use jQuery selectors, CSS, and Flexbox to achieve the desired page layout. Of note, another source of frustration was from debugging the Python CGI scripts when deploying them to the class server. The syntax was correct but there were internal server errors and the SQL queries were not executing properly when called. After many Google searches and checking the server error logs, I found that editing code locally on my Windows desktop and uploading the file to the class LINUX server caused bugs. Using the 'dos2unix' command on the UNIX command line, I was finally able to make the queries.

### Future Work

For this project, although I got the desired webpage and results I wanted, the JavaScript code became quite messy. I tried to implement object-oriented programming practices at the end to import/export modules for more modularity, readability, and reduced redundancy. However, I ran out of time and did not want to risk breaking the code. For the next project, I would want to create cleaner JavaScript code with clearer commenting, documentation, and following best practice guidelines, similar to how I generally write Python code.

With more time, I would implement hyperlinks leading directly to relevant external databases. For example, each protein has a GO, GenBank, UniProt ID, and associated metabolite data. I would update the JavaScript to append hyperlink tags to the respective EBI, GenBank, UniProt, and HMDB URLs to enhance convenience for the user. Moreover, I would fix minor issues such as hiding the tables completely when 0 results are found rather than showing only the table header rows.

Name: Dien Mach  
UNIX Username: dmach1

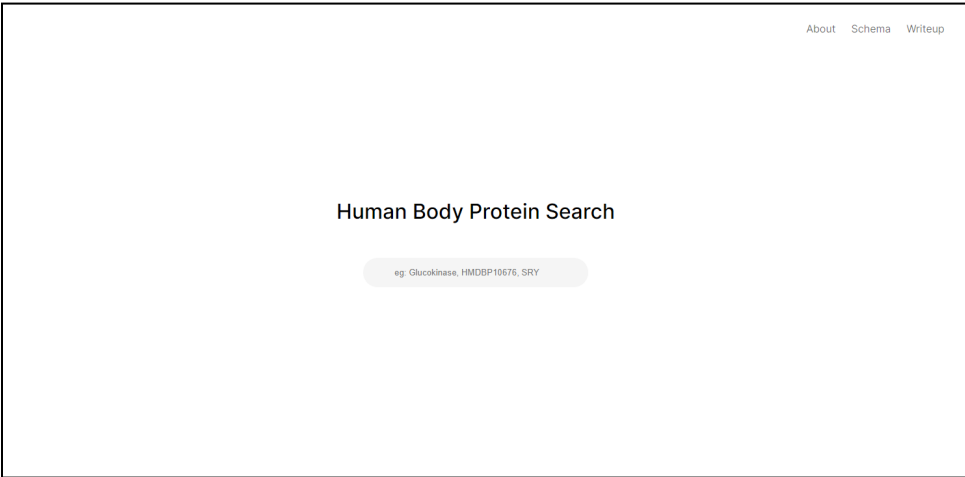


Figure 3: Front page of the website.

5 matches found.

Name	Gene Name	Type	Accession	Version	Creation Date	Last Modified
Creatine kinase S-type, mitochondrial	CKMT2	Enzyme	HMDBP00719	5.0	2013-01-16 01:51:24 UTC	2017-12-08 07:09:21 UTC
Creatine kinase B-type	CKB	Enzyme	HMDBP00720	5.0	2013-01-16 01:51:25 UTC	2017-12-20 20:28:57 UTC
Creatine kinase U-type, mitochondrial	CKMT1A	Enzyme	HMDBP00721	5.0	2013-01-16 01:51:25 UTC	2017-12-08 07:09:21 UTC
Creatine kinase M-type	CKM	Enzyme	HMDBP00722	5.0	2013-01-16 01:51:25 UTC	2017-12-08 07:09:21 UTC
Sodium- and chloride-dependent creatine transporter 1	SLC6A8	Unknown	HMDBP02873	5.0	2013-01-16 02:08:45 UTC	2017-12-08 07:09:39 UTC

Figure 4: Search results for 'creatine' input.

Biological Functions	Name	Creatine kinase B-type
Protein Properties	Gene Name	CKB
Gene Properties	Type	Enzyme
Pathways	Accession	HMDBP00720
Metabolites	Version	5.0
Go Classification	Creation Date	2013-01-16 01:51:25 UTC
Additional Info	Last Modified	2017-12-20 20:28:57 UTC
Literature		

Residue Number	Molecular Weight	Theoretical Isoelectric Point (pI)	Polypeptide Sequence
381	42643.9	5.592	>Creatine kinase B-type MFPSNSHNALKLRFFAEDEFPDLSAHNNHMAKVLTFELVAELRAKSTFSGFTLDDVIQTG VDNFGHPYIMTVGCVAGDEESYEVFKDLDFDPIIEDRHGGYKFSDEHRTDLNFDNLQGGDD LDFNYVLSSRVRTGRSIRGFCLPPHC SRGERRAIEKLAVEALSSLDGDLAGRYYALKSMT EAEQQQLIDHFLFDKFPVSFLLLASGMARDWPDARGIWHNDNKTFLVWVNEEDHLRVISM QKGGNMKEVFTRFCTGLTQIETLFRKSDYEFMWNPHLGYILTCPSNLGTGLRAGVHIKLP NLGKHEKFSEVLKRLRLQKRGTGGVDTAAVGGVDFVSNADRLGFSEVELVQMVDGVKLL IEMEQRLEQQQAIDDMFAQK

Figure 5: Protein properties for 'creatine kinase B-type'.

**Name:** Dien Mach

**UNIX Username:** dmach1

**Works cited:**

[1] <https://hmdb.ca/downloads>