FINAL REPORT- CIS 602 PROJECT

PROTEIN DICTIONARY

Why this Application?

PDB.org - The website or online bank for Protein Data's contains tons and tons of information about Proteins, its 3D shapes, what chains do they contains, their ligand detail, the molecular formula, and much more. It's almost like stars in the galaxy, infinite proteins! But one major problem, for a scientist or any inquisitive person is, these protein information cannot be accessed without an internet connection. Almost nothing can be done without internet. But, for those, who need to get detail in situations where there is unavailability of internet, this application could be little handy. Database is a big question here, but at least the most important, or widely used information about the protein can be handled by this application.

'*Protein Dictionary*' as the name speaks, is a place where information needed about a protein can be fetched, almost replicating the one in PDB, except for the advantage that this doesn't need an internet-connection, something that is a fuel to almost all the people on this earth.

The Desktop Application aims at giving as much information the user needs about a particular protein. Till date, the basic detail about a protein is displayed- Its ID, the Protein Title, its entity details- like number of chains, and what are the chain types- A or B or C, its release date and the resolution of the protein.

Why Python?

First of all, Python is simple and very easy to learn and implement. PyQt had such vast and diverse functionality that could be implemented so easily, that could not have been probably done with other language. Everything was comparatively easier in this, form XML parsing, to connecting the UI with Database, went on smooth with a very little glitches. Second, Python is one of the fastest growing programming language, and not just a language, it has so many libraries and functionalities, that it probably can do the job of 2 to 3 languages in one. Pandas is the python tool for Data Analysis, Bokeh is the D3 version of Python, and lot more. So why not use Python?

Problems, and how it was Solved

One of the initial problems faced were retrieving the details from the website. It was a herculean task downloading the PDB files one by one. There were tons of PDB and getting the XML format of these files was a chaos. Luckily this was solved by using the wget disposition comand. Yet again, every PDB had tons of information about each of its protein. Using XML parsing, the details that were essential, were converted into a CSV format. CSV saved the day.

The GUI of this application was done using PyQt- a Python Framework for User Interface. This was a cake-walk. The drop-down, called as a Combo-Box gets its item list from the CSV that contains structure Id, and Structure Name. This detail was retrieved using Pandas command - *pd.read_csv(file).* But then, back-end was a mess.

The MySQL Database functionality was used. The PDB-XML converted CSV's were then stored into the database. The database was then called in the GUI, using the PyQT command to retrieve the Database table. *Here it turned out messy*. There were Protein Id's which were not present anywhere in the XML nor in the PDB website, but strangely created in the Database. There were Id's like 1-Aug, 1-Dec, 1E+0.001 etc., When looked upon the CSV file, unfortunately these data's were present there too. Then the whole process of conversion had to be done- probably 3 times, until the most dim-witted **flaw** was discovered. For simplicity, after converting the XML to CSV, it was opened with Excel Sheet, to cross-check if all the data's were parsed correct. The Excel Sheet automatically

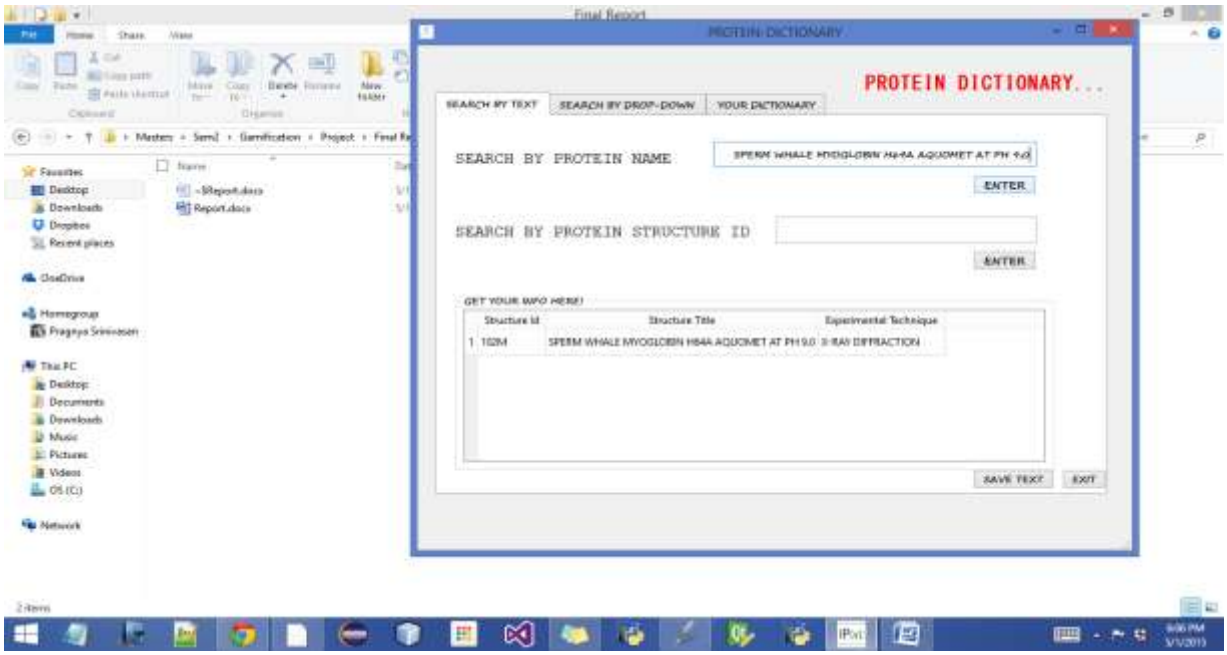
converted all the PDB Id's which were similar to Date- like 1AUG, 2DEC to 1-Aug and 2-Dec.All the E100, E400's were converted to excel exponential format. *Never open CSV's in Excel Sheet, one lesson learnt in this Project!*

After all these problems were resolved, the Front-End and Back-end gelled-in exactly as needed. The Combo box retrieves the ID and Title, and Displays the details, form the database, of the particular option chosen, in the table. Luckily, the barriers till stage, were crossed.

How does '*Protein Dictionary*' work?

The application is Developed using Python. The User Interface is created using PyQt. The application contains 2 tabs, the functionality of each tab mentioned as its name. The **first tab** is to search a protein by typing it in the text-box. The User is given an option to either search by ID or Title. The corresponding detail is displayed in the Table. The table content can be saved into a CSV format. Again, do not open the CSV in Excel Sheet!

TAB 1

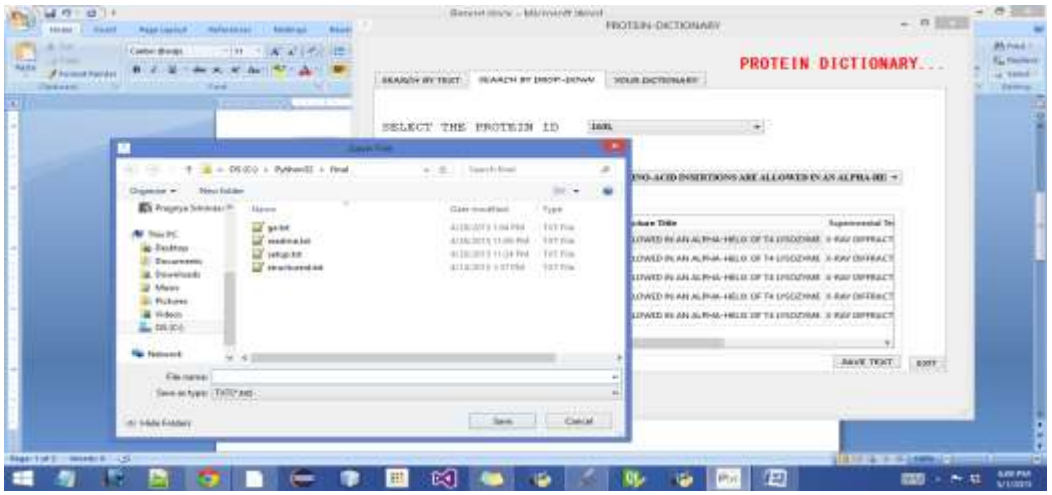


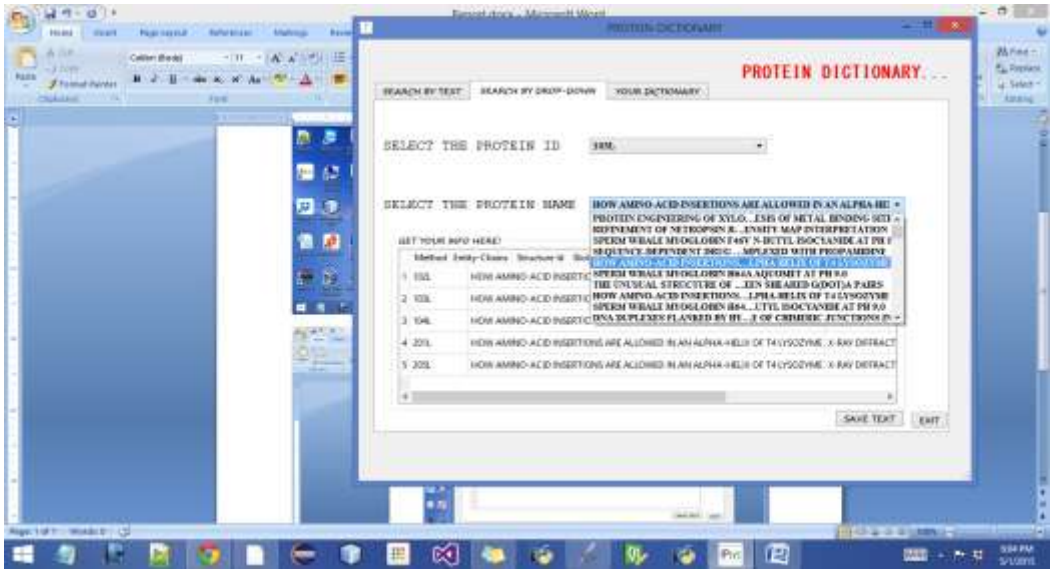
TAB 2 lets user to search with Combo-Box, that is a Drop-Down list. The Drop-down items contains all the Protein Id's in the PDB and another Drop down contains the Protein Title. The user can select the required Id or title, and the result will be displayed in the table space. This again can be saved into a CSV.

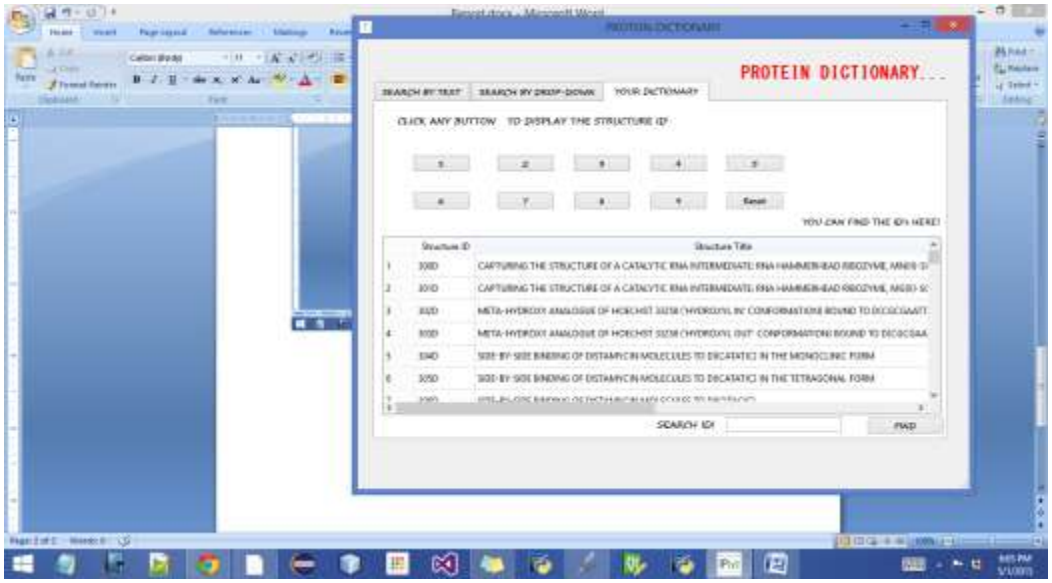
TAB 3 is the Dictionary mode. If a user needs to select a random Protein ID, starting from 1, and he doesn't know what to enter, He can click on the button marked '1'. It will display all the Protein Ids starting from 1 and its corresponding Title. Not much information can be obtained from the Protein name, but this Tab will definitely be useful for future implementation.

If a user needs to know just the Protein Title, from an ID, he can enter it in the tiny search area given. It is specially for people like me, who find it very difficult to remember the long jargon names these Proteins have.

The Reset Button, as it describes, resets the table if the user finds it loaded with too much data.


PROTEIN-DICTIONARY
PROTEIN DICTIONARY...
SEARCH BY TEXT
SEARCH BY DROP-DOWN
YOUR DICTIONARY
SELECT THE PROTEIN ID
SAVE TEXT
EXIT

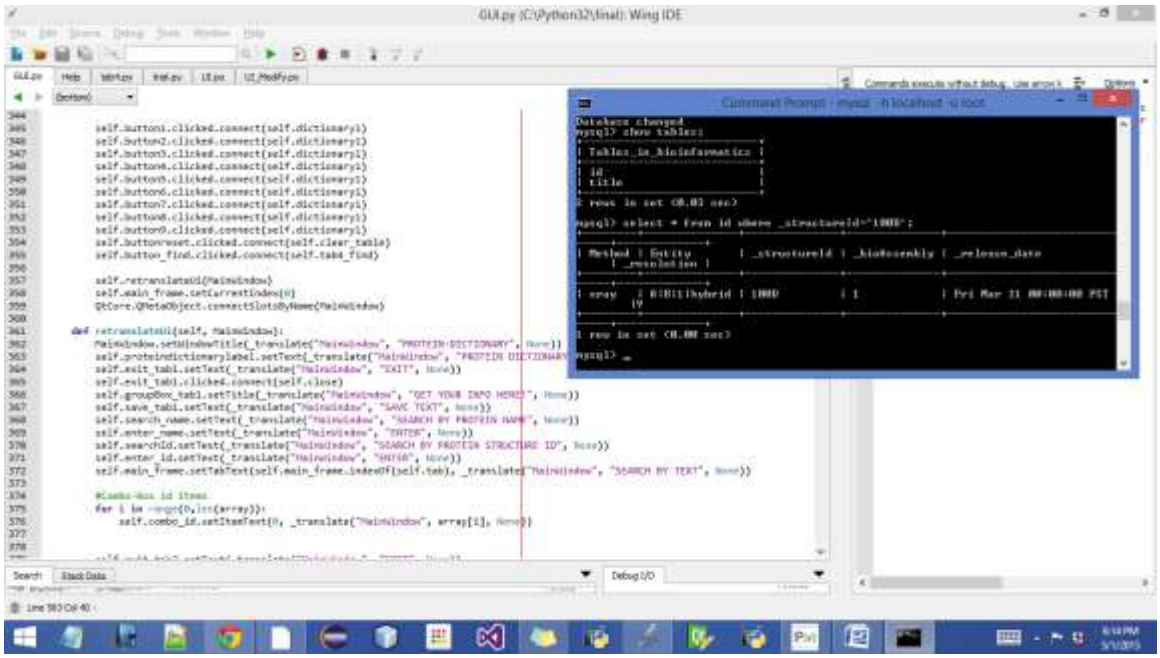

PROTEIN DICTIONARY...
SEARCH BY TEXT
SEARCH BY DROP-DOWN
YOUR DICTIONARY
SELECT THE PROTEIN ID
SELECT THE PROTEIN NAME
GET YOUR INFO HERE!
SAVE TEXT
EXIT


PROTEIN DICTIONARY...
SEARCH BY TEXT
SEARCH BY DROP-DOWN
YOUR DICTIONARY
CLICK ANY BUTTON TO DISPLAY THE STRUCTURE ID
YOU CAN FIND THE ID's HERE!
SEARCH ID!
FIND

The exit in the first 2 tabs, is an alternative for the tradition top right red-'X' button to exit the application. Tab 3 was already really crowded, guess that is why the extra Exit button wasn't required.

A screenshot of the Code with the backend is displayed here. The editor used is Wing IDE, an efficient one to handle Python GUI Scripts.

Code



What more can be done?

A lot more data about proteins has to be retrieved from the PDB website. an efficient way to store in the Database needs to be figured out, especially because of the over-flowing data. A lot more proteins are being added up in the website, there could be a way to dynamically load the new proteins added and store in the Database, so as to make the system efficient for changes.

The most important aspect to work on, which was proposed to be done, if time permits, but couldn't be unfortunately was retrieving these Protein Structure, and displaying them in a 3D manner. This would be really stressful on the Database, but an efficient technique to do this would solve all the problems. One more idea imposed upon was, to be able to create a 3-D molecular representation (with spheres and lines connecting) so that a person who doesn't understand Protein can at least be happy that he understood something from the application.

One part of the project which was pretty confusing was retrieving the information. The PDB's can be downloaded in two form, one is a PDB file, which can be opened using a text file, and another an XML format. It is *better to choose the XML*, because getting the required information from that is a lot easier.

Once the information is parsed and saved into a database, the rest of the journey is a cake-walk. Every detail that is present in this project (about the protein) is taken from the website pdb.org .