

Interaction with Plant Genome database using SQL Alchemy

CS 223 - BIOINFORMATICS

Ermiyas Birara

Nishant Goel

Vigneshwari Chandrasekaran

Dec 15, 2015

FALL 2015

Table of Contents

**Abstract3**

**Introduction4**

Objectives4

SQLAlchemy5

Tools & Technologies6

**Approach7**

Problem Decomposition7

Data Set7

Requirement Analysis10

Problem Design10

Pseudo Code11

Testing12

**Methods14**

List of Modules14

**Results17**

**Conclusion23**

**Contributions24**

**References25**

List of Figures

SQLAlchemy Architecture**2**

Arabidopsis Thaliana Plant **8**

A\_Thaliana database table**9**

FASTA database table **9**

Screenshots**17**

Interactive Menu**17**

Creation of Database Tables**17**

Displaying the A\_Thaliana Table**18**

Displaying the FASTA Table**18**

Insertion of Gene information**19**

Deleting the Gene information**20**

Unsuccessful BLAST**21**

Successful BLAST **21**

BLAST output**22**

# Abstract

Integration of Python and SQLAlchemy will play a crucial role in creating, editing, accessing and interpreting genomic databases with regards to a specific query sequence. The vastness of genomic data makes the integration of python and SQLAlchemy very indispensable for accessing and filtering data readily. Genomic based query sequence and performing BLAST activities momentarily is the basic point in genomic research and studies, and the effectiveness of doing it requires the python programming language package bridging the gap so that query output will be very precise and effective. Moreover, the continuously updating genomic sequence data of various model organisms, such as **Arabidopsis Thaliana** demands the availability of data bases that is readily accessible to incorporate any kinds of changes in the genomic data bases.

Introduction

Objectives:

One of the objectives of this project is bridging the gap between genomic data base (NCBI) and python programming constructs concerning Arabidopsis thaliana genomic data bases. Genomic database, such NCBI and PGSB are very useful data bases in referencing genomic sequences of various model organisms. However, accessing those data bases is very cumbersome and demands the integration of python programming packages to ensure user friendlier and efficiency in accessibility. Besides, integrating python programming language helps to formulate data base management system designed specifically for individualized actions, such as editing data bases related with a specific plant or animal genome sequence.

Secondly, creating and editing gene and sequence related tables using SQL Alchemy is a center point of this project. Having efficient genomic data base management system ensures efficient accessibility of it by users. By taking this into account, we aim to create tables with two or more attributes that are related to sample of A. thaliana’s pre-tRNA genomic sequences. Populating the tables with sample genomic data through python constructs. Next to creating and populating the tables, features to insert new genomic data by user is accomplished using python environment. Users are also able to delete specific genome information from the tables using python codes.

Thirdly, managing nucleotide BLAST output using python and SQL Alchemy is a center point of this project. Nucleotide blasting is the activity performed with purpose of finding alignment sequence for specific query sequences. Having ten sample of Pre-tRNA genomic sequence of A. thaliana from PGSB (Plant Genome and System Biology) in hand, we set out to find the way to efficiently perform blasting of those sample genes against specific non-human and non-mouse genomic data bases. With this in mind, writing python codes and embedding it with the blasting action makes the query output very manageable and direct reference to the query input.

In conjunction with blasting activity, the feature to call up additional information related with the query output from the web is the sub-objective of the project. Once blast output with highest sequence alignment match is obtained, formulating a python code to choose that query output and pull Google imaging is performed as part of the overall project objectives.

What is SQLAlchemy?

SQLAlchemy is defined as the Python SQL toolkit which allows application programmers to utilize the full power and flexibility of SQL.

It is a suite of enterprise-level patterns, using which we can access the database in an effective and efficient manner.

There are basically two principles which SQLAlchemy aims to achieve, are as follows:

1. As the more size and performance start to matter, SQL databases behave less like objects.
2. As the more abstraction starts to matter, object collections behave less like relational tables.

For SQLAlchemy, the database acts like a relational algebra engine and not just a collection of rows and columns. You can perform statements like select, insert, delete on the data using the expression language of SQLAlchemy.

It also provides an Object Relational Mapping in between python constructs and the SQL databases. Here, classes are mapped to the database objects in multiple ways by bridging the gap between object model and database schema and thereby, enabling them to develop in a cleanly decoupled way.

Using SQL, you write a query like,

"SELECT \* FROM BookTable WHERE author = 'Linus'";

Whereas, using SQL Alchemy, we write query like,

book\_list = BookTable.query(author="Linus");

Even the user having no knowledge about the syntax of SQL Language, can easily utilize the functionalities of SQL databases using SQLAlchemy ORM.

The main goal of SQLAlchemy is:

*To change the way, you think about databases and SQL!*

Today, SQL Alchemy API is used by various companies such as Yelp, Reddit, Mozilla, Survey Monkey and many more.

Pictorial Representation of the architecture of SQLAlchemy.

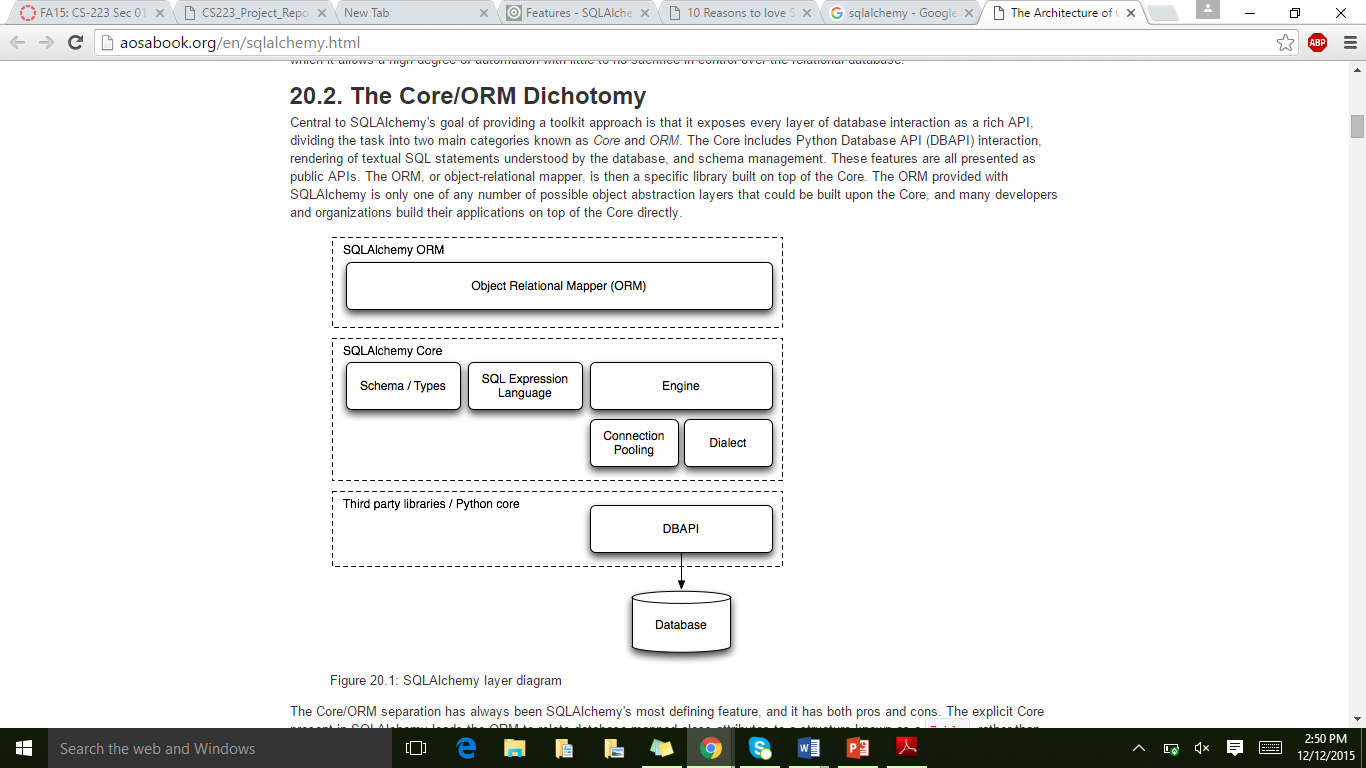


Figure 1: showing core components of SQLAlchemy

SQLAlchemy divides the components into two main categories known as *Core* and *ORM*. The Core consists of the Python Database API (DBAPI) to allow interaction and rendering of SQL statements by the database, and schema management. Also, Core has object relational mapper, built on top of it. The ORM combined with Core provides the abstraction layers to build applications directly.

Tools & Technologies

Language: Python 2.7

API: SQLAlchemy, PrettyTable, URLLib2, PIL, SimpleJson, etc.

Database: MySQL

Tools: Enthought Canopy Editor

Approach to develop the Computational Solution

Problem Decomposition:

* Creating a bridge connection between python and genomic data base.
* Creating tables using SQL Alchemy for a list of genomic information referring to *A. thaliana*.
* Inserting data about plant genes in the database tables.
* Displaying database tables containing all attributes of a particular gene.
* Running BLAST of a nucleotide sequence to find possible alignment sequence.
* Displaying query BLAST output for a gene.
* Extracting data from the web that is related to that query BLAST output.

Data Set **Arabidopsis Thaliana:**

Dataset Summary:

Arabidopsis Thaliana is a model organism in plant genomic sequence as H. sapiens and E. coli are in eukaryotic genome sequencing projects. By being a model organism, A. thaliana genome is fully sequenced is used as a base in blasting against other plant genomic sequences or the other way around. The reason overly is because of its relatively manageable genome size to work with. Performing BLAST (Basic Local Alignment Sequence Tool) to find the corresponding genomic sequence alignment plays a major role to understand the structure and functions of a specific query nucleotide sequence. However, accessing the vast genome data bases is not an easy task and requires continual updating of the data base. Beside, interpreting query output is another level of challenge that comes along with accessing most genome data bases.



Figure 2: Arabidopsis Thaliana Plant

File Format:

The format of the files that we used to create and populate the data tables is a CSV format.

Data Object and Types:

* **Gene Name**: It is unique for each respective gene and involves the 1st letter of the plant genes and species name followed by numbers.
* **Gene Type**: It is the gene that encodes for specific protein.
* **Description**: It is any additional information to describe the gene and its characteristics features.
* **Length**: It is the length of the entire nucleotide sequence representing the gene.
* **Strand**: These are two types of strand features: Forward and Reverse.
* **Start**: It is the order where the gene sequence starts among the entire genome sequence order.
* **Stop**: It is the order where the gene sequence stops among the entire genome sequence
* **FASTA**: It is the nucleotide sequences encoding the gene.

Sample Database Tables:

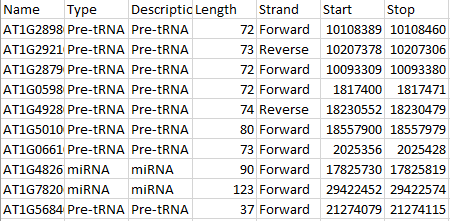


Figure 3: showing A\_Thaliana database table

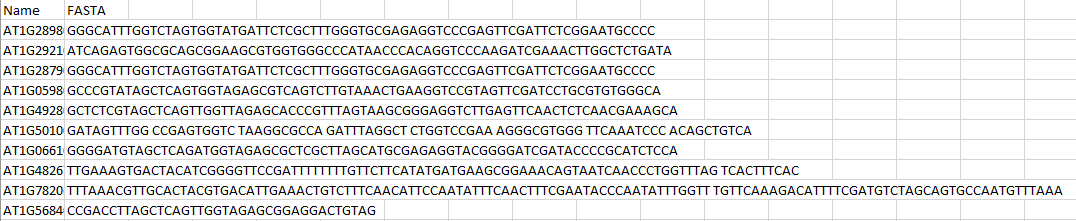


Figure 3: showing FASTA database table

Requirement Analysis:

It is all about “What needs to be done”. So, the features that needs to be incorporated are as follows:

* Getting the data from NCBI.
* Making it available to the user to perform various functions using our application.
* Implementing functionalities to insert, delete, and view data on the console.
* Implementing the functionality to analyze BLAST output so that the user is able to call additional information, such as imaging from the web.

Problem Design:

Depending on the analysis we did on the requirements, we planned to design the problem in the following way.

* Creating the database tables in MySQL database.
* Populating the data into the tables through SQL Alchemy.
* Providing the user an interactive menu to use the application.
* Asking the user for input, performing the validations, querying the database and displaying the output.
* Using URLLib2 API, extracting images of the analyzed BLAST output from the web.

Pseudo Code:

Here, we are mentioning pseudo codes of a few modules that we have implemented.

Insertion:

1. Connect with the MYSQL database using SQLAlchemy by creating an engine.
2. Call the ask() method to make user enter the data.
3. Ask the user for the information related to the gene such as name, type. Description, strand, length, start codon, stop codon, FASTA sequence.
4. Perform validations on the input entered by the user.
5. Execute a SQL query to check whether there is any gene present in the database with the same entered by the user.
6. If Yes, then ask the user to re-inter the information again.
7. Else, check whether the length, start and stop attribute are numeric or not. If not, ask the user to re-enter the details.
8. Else, check whether length is equal to the difference of the range of stop and start codon. If yes, ask the user to re-enter the data.
9. Else, perform the range validation in which check the value of start and stop codon is clashing with the ranges of start and stop codons of other genes. If yes, then, ask the user to re-enter the data.
10. Else, return all the values to the calling function.
11. Execute the query to insert data into the database.

Deletion:

1. Connect with the MYSQL database using SQLAlchemy by creating an engine.
2. Get the name of the gene to be deleted from the user.
3. Check if that gene is present in the database.
4. If yes, then execute the query to delete the gene from the database.
5. Else, throw an error to the user to enter the correct name of the gene.

BLAST:

1. Get the gene name from the user.
2. Search in database table for the FASTA sequence of that gene.
3. If no match found, inform user that the gene information is unavailable.
4. Else, call the BLAST code, by passing the FASTA sequence.
5. Look for matching BLAST result from the 5 reference databases.
6. If a BLAST match is found, call Google sub module to display the picture of matched plant.
7. Else print “Unsuccessful BLAST / No Hits Found”

Google Search:

Prerequisite: Active Internet Connection

1. Supporting module for BLAST.
2. Get input (search string) from BLAST module.
3. Embed this string to Google search URL.
4. Using URLLIB2, Simplejson packages, interpret the search result.
5. Save the image to local machine using PIL package.
6. Display the image to the user.

Testing:

Unit Testing: Built each module and performed Unit Testing

Regression Testing: Integrated all modules and performed regression testing to make sure the variables or other member functions doesn’t conflict each other.

Black Box Testing: Gave input for all the modules with all set of inputs that covered all validations. Input sets were:

1. Set 1 that passes all validations
2. Set 2 that fails at one or more validation points.

We marked it complete when we were able to get success /failure message as expected.

White Box Testing: Code was given input sets that touches all statements in branch, looping and method calls. For instance, In BLAST module, we perform BLAST in iteration with each iteration referring to reference databases acquired from NCBI.

1. Input 1 was given such that the match was found in 1st database.
2. Input 2 was given that had no match in 1st but had it in 2nd database.
3. Input 3, 4, 5 were given in the same pattern to check if the code reaches till the final iteration in pursuit of match.
4. Final input was given that had no match in all the 5 reference databases.

On getting the desired output on each case, it was possible to affirm that the loop iteration was proper as we expected.

Methods Used

The modules we have implemented and the how we have implemented those are mentioned below.

List of Modules:

* Interactive menu
* Creation of Genes tables
* Displaying the Gene information
* Insertion of Gene data
* Deleting the Gene information
* Option to do a Gene BLAST

Interactive menu:

* This is the entry point to the application.
* Designed it using the PrettyTable and TextTable APIs.
* This menu is kept active until the user decides to exit the application.
* Displays all the available options to the user and prompts him to enter the desired option.
* End of every iteration, user is prompted if he wishes to continue or not.

Create module:

* Created database tables from Python through SQL Alchemy.
* Two tables get created. One for holding the details of genes like name, type, description, length, start & stop codons.
* To ensure integrity, Name of the gene is made the primary key.
* The another table contains the FASTA sequence corresponding to the gene.
* These tables are related through foreign key referential integrity.
* The primary key in the former table is the foreign key in the latter one.
* This table is primarily used for Blast module.

Display module:

* This module extracts the complete data from database and displays that to the user on screen.
* Data from both the tables will be displayed.

Insertion module:

* This module allows user to update the database with new data
* It prompts user to enter data for every field
* Internal validations are handled in Python
  + Primary key validation
  + Synchronization of length with start and stop codon
  + Validation on the FASTA sequence content
* Once all the validations are passed, the whole set of data is added to the tables
* The parent table is updated first and then the FASTA table is updated with the sequence.

Deletion module:

* On user request, the complete gene information can be deleted from the plant genome database.
* Just like insert, to ensure consistency, the child table data is removed first, then the parent table is altered.
* This is because, child is dependent on parent, so for the data to exist in child, it needs parent data. Suppose, if the parent data is deleted prior to child data, the reference to child will be active and it will throw an error.

BLAST module:

* User can opt to BLAST a specific gene
* On entering the gene name, the corresponding FASTA sequence is retrieved (from child), the BLAST module is called with this FASTA.
* BLAST is done against 5 target NCBI plant databases
* Starts with one target, if a match is not found, iterates through the other 4 databases, until a match is found.
* At the end, if no match is found. It displays no match found
* On finding a match with the target, the target with which the highest match is achieved will be retrieved.
* Then the image of the target will be googled through code and it is displayed to the user in the front end.

Functionalities we have coded & not imported:

1. Range Validations in Insertion Module

We have written the code from scratch for validating the range of the start and stop codon that user has entered. We check the ranges of start and stop codon for each of the genes existing in the database. The start and stop codon of a new gene must not clash with the range of the stat and stop codon of existing genes.

1. Iterative BLAST

Code has been written to do BLAST the desired gene with the target. If no match is found, the control navigates to do BLAST with the next target, by modifying the batch file, iteratively until all the databases are searched. At the end, if there are no more databases to navigate, and no match had been found, the user is informed that “No HITS found”.

1. Image search and processing

Once the BLAST is done, the output is processed to pick the best matched alignment among all the possible BLAST output reported. The matched name will be searched in web by appending the string to Google search URL through Python code. The search result is a string of characters that represent the image. It is converted to jpeg file and displayed to the user.

Results (Screenshots)

Interactive Menu:

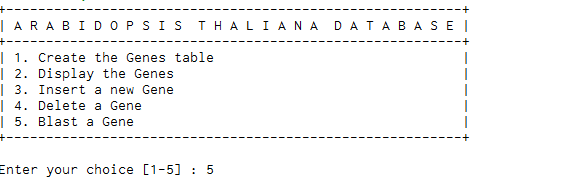


Figure 5: showing an interactive Menu

Creation of Database Tables using SQL Alchemy:

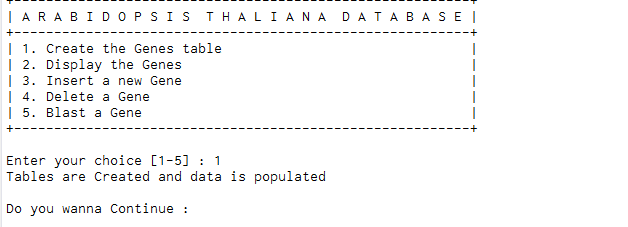


Figure 6: showing creation of database tables

Displaying the Gene Information:

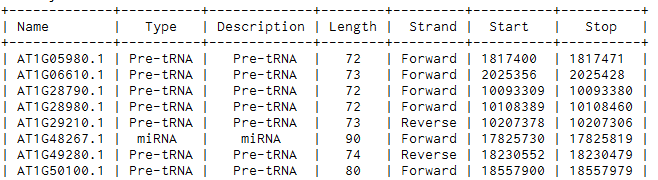


Figure 7: showing A\_Thaliana database tables with all the Genes information

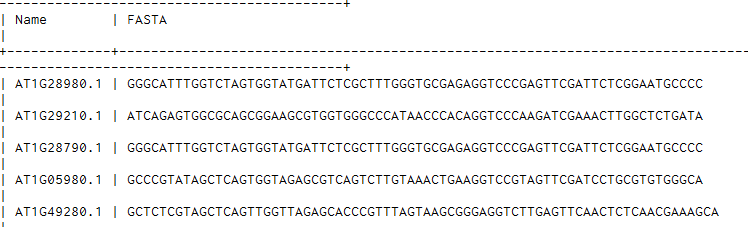


Figure 8: showing FASTA database table

Insertion of information of a particular Gene:

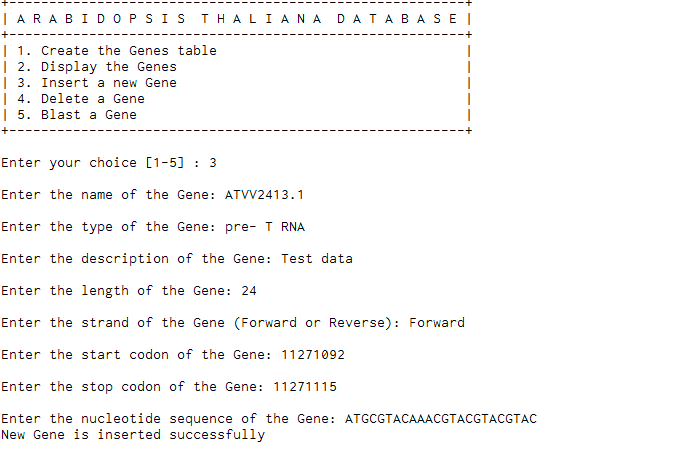


Figure 9: showing insertion of information of a particular gene

Deletion of a particular Gene:



Figure 10: showing deletion of a particular record

BLAST Module:

* Unsuccessful attempt

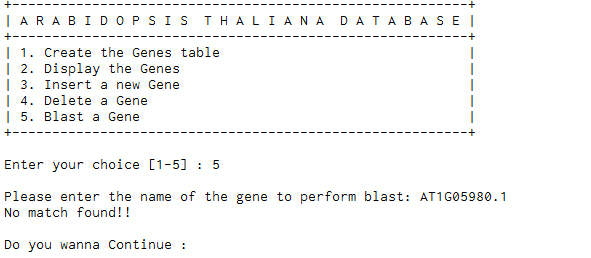


Figure 11: showing the unsuccessful attempt of BLAST

* Successful attempt

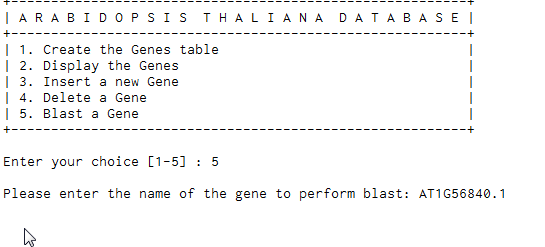


Figure 12: showing successful attempt of BLAST



Figure 13: Showing the BLAST output

Conclusion

Setting our project with the aim of creating a bridge between python programming language and genomic data bases, the results showed that integrating Python programming language with databases of big genomic information ensures reliability and completeness in accessing genomic data output. Moreover, we were able to show that Python code working in coordination with databases helped to create table, insert genomic data and run BLAST query successfully. Finally, we comprehended that it is also possible to integrate query output with supplemental information, such as image by fetching from the web through python code. In the process of accomplishing the above major tasks, we fully grasped the knowledge to code using different python constructs and SQL Alchemy, which is a python based API for object relational mapping. Also, as part of our objectives, we were able to interact with relational database of plant genome, and managing data of creating, inserting and deleting genomic data.

Contributions

**Ermiyas Birara:**

As a team member, my contribution was to come up with a data set that the whole project was based on. Beside coming up with a data set, I also guided the project as how best it could be integrated with python code constructs and what we expect from the implementation of the objectives. Moreover, I came up with the idea of what would be the objective of the whole project and expectation after implementation. Finally, interpreting FASTA and BLAST output of sample genomic sequences to the rest of the team members was my contribution in the whole project.

**Nishant Goel:**

In this project, I started with understanding the plant data set that Ermiyas provided. Since, we were asked to use SQLAlchemy for our project, I started exploring it like how can we accomplish our objectives using this API. Once, I had the basic understanding of SQLAlchemy, I started with creating the bridge between MySQL database and our python programming environment. Since, I was responsible for designing the basic interactive Menu, Creation of database tables, Insertion of Gene data, Displaying the data and Deletion of a particular gene, I started with designing the menu and creation of database tables. After that I implemented the insertion module in which I integrated various validations like range validation, Length validation, Name validation, etc., related to the data to be inserted. Then, I implemented the functionalities of displaying the data and deleting the data from the database tables using SQLAlchemy.

**Vigneshwari Chandrasekaran:**

Started with exploring SQLAlchemy packages for creating database objects in MYSQL database from Python. Once the dataset was ready, started implementing BLAST module for the A. Thaliana genes with the target plant genes those downloaded from NCBI website. Implemented Google Search module for BLAST output to present the image of the plant that had the best match with the gene user asked for. As a part of validation for insert module, initiated coding for validating the newly entered values with the data from our database.

References

* http://pgsb.helmholtz-muenchen.de/plant/athal/tablejsp/geneticElementList.jsp?type=all&contig\_id=81&page=2
* <http://www.sqlalchemy.org/>
* <http://pajhome.org.uk/blog/10_reasons_to_love_sqlalchemy.html>
* <http://aosabook.org/en/sqlalchemy.html>
* <http://blast.ncbi.nlm.nih.gov/Blast.cgi>
* <http://docs.sqlalchemy.org/en/latest/core/dml.html>
* <http://mapfish.org/doc/tutorials/sqlalchemy.html>
* <http://www.ncbi.nlm.nih.gov/guide/howto/run-blast-local/>
* <https://en.wikipedia.org/wiki/BLAST>
* https://www.blaststation.com/intl/members/en/howtoblastwin.html