Biocomputing coursework reflective Essay – Florence Lai

1. Approach to the project

- Interaction with the team:

I was nominated to be the team leader during the first meeting when we decided our roles. Oliver volunteered to do the database layer as he has not completed Biocomputing 1 yet. Maham has interest in doing the front end, which was fine for the rest of the team. Nini was happy with doing the code testing as suggested, and I was happy to do the business layer to tie everything together.

The role assignment went smoothly and there were no conflicts or team members competing for the same role.

As the team leader, I took on some project management responsibilities to organise the team. I set up the github account for the team members and reminded everyone to push their code consistently and to keep everyone up to date with any changes to the design. We had a Whatsapp group chat to discuss the design and development progress which was crucial due to the lockdown. Additionally, I organised a Google Hangouts videoconference with the team to solve some of our differences during a crucial stage in the development. When I notice any issues, I contact the team member directly but not via the group chat.

All in all, the interaction between the team members went smoothly with us all working remotely and was sufficient to produce a working product. If not for the lockdown, we could have had more face to face meeting and coding sessions which may have helped with the development.

- Overall project requirements - group

During the second (and the last face to face before the lockdown) meeting with the team, I printed out the first two records of chromosome 6 from Genbank. As a team, we highlighted the different parts of the records that are needed for the requirement.

During the development phase, we encountered some small problems with the requirement that required fixing along the way.

- Overall project requirements – personal

As I did not want to over complicate the project, I started with the skeleton code provided and adjusted it according to our team's agreed API. Also, I found suppliers for restriction enzymes with information on where the cutting site within the sequence will be located (example : https://international.neb.com/products/restriction-endonucleases). Hence, these additional restriction enzymes were added to our browser.

2. Performance of the development cycle

We follow a simple development process of identifying the requirements and then trying to satisfy them with our code fixing problems along the way. With the current situation, there's enough challenges for us in this project already and we did not have enough time and experience to implement advance methodologies such as TDD.

Considering this, we were able to achieve the requirements within the time frame of the project.

3. The development process

The initial design process was relatively short as we adopted the skeleton framework from the course repository and decided to expand upon it. The API design was also based on the skeleton code from the repository and we added the necessary elements to make it functional. This bought us a lot of time to allow us to concentrate on the coding part.

As I thought the calculating of the total codon frequency will be time consuming. I started on this part early on in the development process, which needed the data parsed in order to complete this. As such, I partly parsed the data get the ball rolling which was later adopted and edited by Oliver to use in the construction of the database. Doing this part early saved us a lot of time as we managed to resolve the data issues earlier on, for example, agreeing on what type of records to ignore (records with non-standard bases) and the best way for the regular expression to grap the necessary data from the record (the CDS part especially).

Later on, we realised that we did not fully take into account of requirement 2, where it is required to search the database to find an entry based on any of the followings – gene identifier, protein product names, Genebank accession or chromosomal location. This means that the search may return multiple records, and up to that point we have not done the detailed design on what this entails.

Furthermore, our previously agreed to API can only return search items based on the accession number as it was unique. If the user inputed the gene identifier instead, it will not work. Therefore, we had to redesign the application to accommodate this. We had a quick discussion with the team and we all agreed that the design and API needed to be updated.

I proposed that the listing page can be used as a summary displaying a table of records that may or may not be filtered according to user input. The search page would then display the details on only 1 particular record, identified by accession.

In the working version, a list of records will be displayed for the user to further select a single record for the detail page.

There were other minor issues that we had to address in similar ways. This resulted in a iterative development process, as we ran into issues and had to resolve this via analysis and redesign.

4. Code Testing

While testing my own functions to determine whether the output from my API was as expected, I have also made a dummy_dbapi in order to test in controlled environment. I used known sample data originally from the database as a basis for the functions so that it will return this data without a functioning database backend. I have also committed this to Github, as I thought this may be helpful for the other team members for their own testing.

I have also contributed to creating doctests for the blapi to test my module's functionalities after I got the initial version working. This was later placed in the testing directory.

5. Known issues

Originally, we thought that if the user did not input anything in any of the search boxes, it will return an empty string like ‘’. However, it return None instead, which had a major effect on how the dbapi constructed the MySQL queries. This is the only major bug we needed to fix affecting all three layers.

One of the problem I have encounter in the blapi was the addition of a ‘star’ to mark the restriction enzyme cutting site. As I had turned upper casing in the DNA sequence together with the ‘<span class = "re">&starf;</span>’ syntax for the star addition. In HTML, the upper cased ‘STARF’ was not recognised and it was later corrected.

Both of these issues have now been fixed and there are no further bugs that we notice.

In terms of issues for improvement, we wanted to try multi columns export for the codon frequency table, however, the result was not as good as we hoped, so it was not adopted. There maybe further room to improve the appearance of this table as right now it is a very tall table. Also, we did not implement the custom restriction enzyme sequence searching options which can be later improve upon.

6. What worked and what didn’t – problem and solutions

The iterative development process was very effective in our team by starting the coding work early, we have time to debug and fix issues that we previously did not foresee (for example, the None return). Even though we had to redesign and rewrite parts of our application, which result in more work. This benefits us more due to our relative inexperience, we don’t have the skills to design a fully functional application right at the beginning.

One of the solutions I was more happy with is preventing the restriction enzyme cutting site to be within the coding region. I made use of the upper and lower casing in the DNA sequence. By making the coding regions lower case, this prevent the replacement of the cutting site to be in the wrong position.

Something that we tried that didn’t work as well was the multi column table mention above. We did this by wrapping the table inside a div with css attribute “column-count 3”, and then adding a extra column header table with 3 headers, and then hiding the real column headers at the end of the original table.

I found the need to compile the API design in the very early stage (week 2/3 into the module) very difficult and it is very easy to overthink the API design then. At that moment, I have only started to understand the data structures of GenBank. Therefore, the API design has not been fully thought through and needed to be updated at a later stage of the project. It was fortunate that our team decided to keep things simple and went with using the existing functions in the skeleton code.

7. Alternative strategies

In terms of project design, what would have been useful is for us to review the requirements in more detail and come up with a rough design of the whole workflow before starting the coding part. This would allow us to come up with a better designed API in the first instance. It would also make it less iterative but would be a good balance having the experience going through it.

In terms of the blapi coding, as I based it on the skeleton code given, there’s only two functions. However, it can be more modular, with more functions and this could make code testing a bit easier.

8. Personal insight

I think all of the team member worked really hard to complete their task given the situation. We have a very nice front end thanks to Maham, she put lots of effort in adding Javascript to her html in order to improve appearance and usability. Also, Oliver did a great job in sorting out the database at the very early stage of the development process, especially as parsing the Genbank data and importing it was a huge task. Nini thought of many ways to implement the testing into the project, but encountered some difficulties due to the design of the code (eg. lack of functions in some parts), that is difficult to fix close to the end of the project as we were not able to make large-scale changes.

We completed the first working draft version of the genome browser by the 19th April. Although it was delayed from the original date that we agreed on (14th April) by a few days, it is still relative successful given the circumstances. This was planned so that we would have enough time to improve our application and for testing before the final deadline.

This experience emphasises how important time management is in software development. We have a working browser thanks to the managed timeline we set as a team. Even though we have a working version 2 weeks before the deadline, we were still fixing bugs and making improvement up till the last day.

Keeping the design simple and not over thinking the design and API helped us focus on solving the problems and meeting the requirements. This is important especially because none of us are professional software developers (yet).

In terms of coding I found that Biocomputing 1 prepared us well for the technical challenges in the code for this module. In fact, the programming problems in that module may be even more difficult than in this project. However, this is a more practical exercise with real-world use, which let us gain wider skills, and so has been a learning experience for me. In this module, I was able to use git much more and deal with coding in a collaborative environment, which would be useful for me in the future. Furthermore, coding in a team with multiple developers working on the same code results in issues that needs to be managed (eg. git merge). This experience is very useful to have.