Reflective essay

This essay summarises my shared experiences in this work with the other members of the group, as well as the hurdles and ways to solve them I underwent. I was in charge of the front-end layer.

1. Approach to the Project

The project was originally perceived as a challenge by all members of the group since none of us had worked on something similar before, but I considered the front-end layer to be the most difficult part, as web development is a field of knowledge I'm very unfamiliar with. I barely knew how to write HTML before this project and let alone how to integrate information from other layers in order to make a functional website.

The same day we were assigned the group members we decided the roles we would take for the project. Everyone had different interests in the layer we wanted to work on and, initially, no one wanted the front-end layer. Of course, my colleagues were thinking the same way I did. Eventually, I accepted the role because I thought of it as an opportunity to gain some understanding of something I was oblivious about, and no harsh feelings were made that day. I was also declared "the man in charge" of the group arbitrarily.

The following days we listened to the advice Andrew gave us and started working on our APIs but in a confusing and very naïve way. Later, we realised we didn't even know how to explain what an API was, which is an extremely trivial thing for me right now.

In my case, I thought about designing a fancy and good-looking website, with animations, colours, personalised logos… but keeping things as simple and functional as possible is the best approach for this kind of project, so I left the web designing after having a functional web, despite the hype I had regarding the matter.

2. Performance of the development cycle

The first month after the role assignment pretty much consisted of learning HTML, how the functionality of tags, CSS, a bit of JavaScript… I didn't rely on my group for anything, as I was using dummy data in the early steps of development to see how the code was displayed on a browser and how the CGI scripts worked.

We didn't gather many times during the development process due to the transport difficulties our accommodations arise, but we were always in touch via WhatsApp and the GitHub repository. We knew we could rely on the other members of the group in case of necessity, which was reassuring.

In the late stages of the project I was able to integrate most of the functions of the business layer, but both these layers had to use dummy data, because the database wasn't functional. In the end, the database layer is not functional and we weren't able to move on from the dummy data, which was essentially the expected API of some of the search functions from the database.

3. The development process

Once I had an idea about the functionality expected from the website, I started working on it in the most simple way I could think of: an HTML file, having two forms that would redirect to a newly generated webpage via a CGI script. Basically, two nodes connected by a single directed edge. Eventually, I wanted this to become a two-way path to reach the same destiny. One of the paths would be the table with all the entries of the database and the other would be the input from the user.

Learning how to implement CGI scripts wasn't a difficult task, but as I went through the documentation on how to write them I realised there's not a canonical way, and also that it largely depends on the programmer, so I chose a certain format and stuck to it.

For the details of the entries, the last and most important step in my layer, I struggled a bit to keep all HTML elements in place and not crash the browser, but errors were easily overcome.

As I started improving my HTML code, the header, the colours and the background, I realised my files contained too many lines of code, and that many things were repetitive and could be better implemented with functions, so I moved everything I could to an `html_utils.py` file, which contains supporting code for the CGI scripts. The same happened with hardcoding, and I created the `config.py` file that every layer would access later.

Then I moved on to the CSS. I tried experimenting and creating my own classes, but I was too inconsistent, and I felt like I didn't know enough about the matter to come up with something worth my time. In the end, I opted for using an existent style: w3.css. I found it very easy to use and quite powerful. Not an excessive amount of time was wasted on this.

The most time-consuming part was learning JavaScript and integrating the business layer code without crashing the browser. I acted as a tinkerer and found many interesting JavaScript codes that make the website really attractive, so I think the huge amount of time I spent optimising their functionality paid off.

4. Code testing

In particular, all code in my layer is very easy to test. The HTML code was constantly being evaluated on validator.w3.org and the functions were given dummy code that was

basically some variables with the expected format from the business layer API. Nothing particularly remarkable regarding code testing for me, not even when I actually had to use the actual functions from the business layer.

5. Known issues

The database layer is not functional at all, so dummy data has been used to generate the webpages through the business layer functions. The other two layers are fully functional and all bugs have been taken care of.

6. What worked and what didn't – problems and solutions

Some annoying bugs would pop up from time to time, especially in my layer, causing the browser to crash or the details webpage to display misplaced characters or unwanted information, but eventually, they were solved.

In particular, the requirement addressing the restriction enzymes was initially solved by formatting the code resulting from the business layer functions. Doing some research about jQuery plugins I found a more attractive way to implement that function. It's more dynamic, yet equally functional, and the business layer requirements are not hindered at all. Both the members of the layers discussed this and agreed on implementing it with the plugin, but the functions from the business layer are also operable and can be found on the GitHub repository.

Everything else in the business and front-end layer worked as we expected.

7. Alternative strategies

In my particular case, I considered several designs for the website, but not many alternative strategies. I would always discuss with my team my ideas, which would usually consist of asking for advice for problems with a binary solution. This inevitably ended up in having a very optimised strategy to address the requirements, but no alternative approaches.

8. Personal insights

As I expected, I've gained inestimable knowledge about web development, which seems very versatile for my future as a Bioinformatician. I'm glad I chose the front-end layer because I think I'm fluent enough with MySQL and Python.

Also, I had never been a participant in a group project of this dimension. It has been quite enlightening for me the importance of teamwork and agreement, as well as planning things ahead.

As a programmer, in general, I'm still learning to write useful and good-looking code, and I always go through documentation about code style, rules and tips when I code, so the project helped me to further improve my programmer-thinking.