SENG3011 - LetsHD
Deliverable1 Report

By -

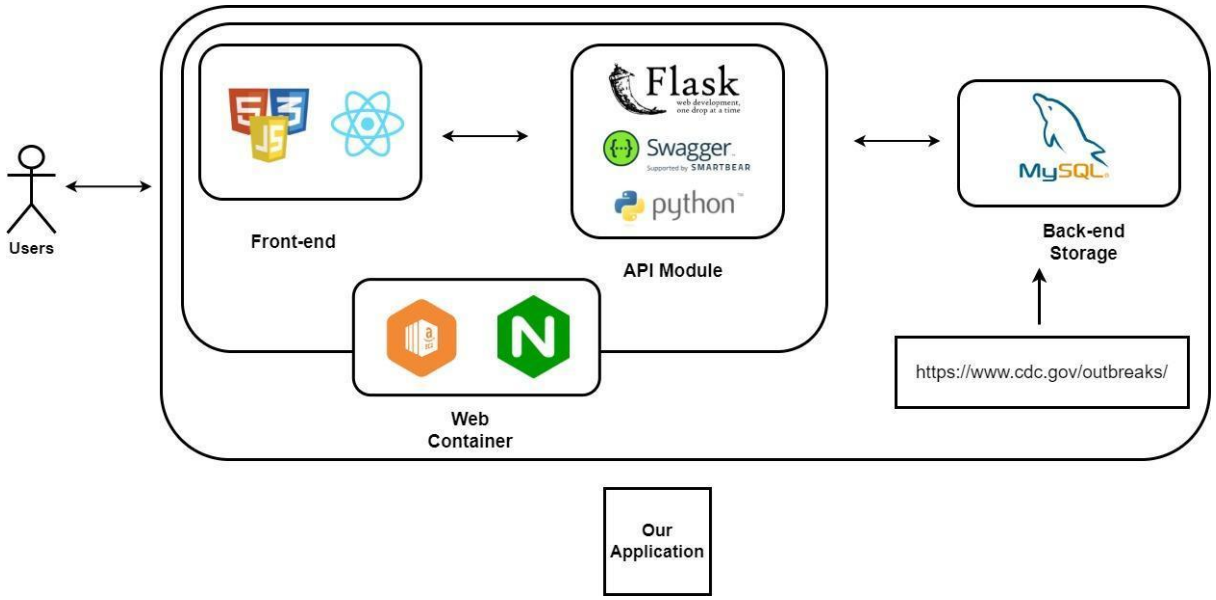
Jiaxin Liu(z5191691), Zhiyuan Yin(z5191561), Yilin Zhu(z5212995), Ziyu Zhou(z5212919) Jing Deng(z5213505)

Table of content

GitHub Repository - https://github.com/Zephyr8097/SENG3011_-letsHD-

Design Details

API module design and Web support



(SAD Diagram)

API Module development

In order to enforce stable and efficient communication between our front-end infrastructure with our database, a well-organized API is inevitable. As shown in the software architecture diagram above, the chosen programming language and framework for the API modules and back-end storage are Python, the Flask web framework and MySQL database. Python and the Flask web framework together will provide a portal to receive and handle different requests coming from all kinds of clients, these requests will be typically querying for existing data entry inside our MySQL database. Depending on the permissions of the clients and the requests, the API should either reject the requests or proceed to return desired information. The CDC API as well as any necessary scraping will be put into use in order to populate our SQL database with meaningful and accurate information.

Web Support

For deploying our API services to the web and making it publicly available, we have chosen to use the Flask web framework. The Flask web framework offers straightforward

implementation to declare how our API should behave when different HTTP requests are received. Once all of our intended behaviors of the API are integrated with corresponding Flask web handlers, launching the Flask server inside our EC2 instance will then deploy our API to the address where our EC2 is located.

The Rationale

Python

Python being one of the most used languages across ranges of different fields in the industry has its very own advantages. Firstly, it provides developers the options to run the program with particular dependencies in a virtual environment which is extremely efficient when multiple developers are working together and iterating the API in fast-paced development cycles. Additionally, being a popular option in the field of API development, support from the community is widely available on the internet, thus reducing the difficulties of any necessary learning. Lastly, Python is highly compatible with other frameworks and tools as it provides built-in functions to efficiently interact with external entities such as MySQL databases and the Flask framework. Other programming languages such as JAVA and JS provide benefits of their own, but are not as light-weighted as Python due to their compatibility and readability issues.

Flask

Although there is a wide range of different web frameworks available for Python, Flask was an obvious option for our development team. The Flask framework is well known for its lightweight and easy to negotiate characteristics, it is effortless to set up and integrate into any system. Additionally, it is extremely scalable meaning the potential of the application is not restricted. And most importantly, the Flask framework happens to be the most familiar framework to our back-end development team.

Swagger

Documentation plays an important role in communicating efficiently how our API behaves to the users. Swagger documentation is a documenting platform that specializes in explaining the behaviors of an API. It is capable of showcasing all available endpoints provided by the API in detail. It is highly customizable, developers have the options to group endpoints by characteristic and provide samples inputs or outputs. Incomparably, Swagger also provides viewers the ability to send sample requests to examine the API in real time.

MySQL database

As many databases as there can be, they mainly come in two types, either document-based or schema-based. We have chosen to integrate schema-based databases with our system as it provides more emphasis on the correctness of the data entries, the additional flexibility provided by document-based databases is not appreciated in this particular project as most of the entries our application will be storing comes in fixed form. Out of ranges of schema-based databases available, we decided to proceed with MySQL database as it

smoothly interacts with Python and is highly portable which fits our rapid development cycles. The database sizing and debugging issues that come along with MySQL are not concerning as the amount of data our development team are planning to store is not extensive, and debugging can be achieved through the use of MySQL workbench.

API Naming Convention

During the build out process for the API infrastructure, the name of the endpoints should be a brief description of the behavior of the endpoint, if the description is longer than a single word, words should be connected via a "_". For example, an endpoint for fetching reports should have the name of "get_reports"

API Version Control

As the source code of our API infrastructure is managed through GitHub, our API's version will also be managed through GitHub. If for any reason, our development team wishes to revert the version of the API, this can be easily done through GitHub's version control. Before commencing to build out a new iteration of API, the development team should work together to come up with a documentation on Confluences clearly describing the intention for the new version of API and why it is necessary. Each iteration of API should also have their own swagger file clearly identifying their available endpoints and corresponding behaviors.

API parameters considerations

Foremost, we are using the RESTful API architecture and design principles, so it is necessary for us to build an API that follows the constraints of the standard. Thus, the parameters of the API we use should also follow the constraints, otherwise improper design of parameters will straightly delay the delivery of the project.

Besides, the open data that we can have are from the following website: http://www.cdc.gov/outbreaks/ and also navigate to the articles about the outbreaks on it. The specific steps of how the API works can be demonstrated as the following steps.

1.Retrieving parameters from the user

To grab the search conditions offered by the user, we will be using a HTTP GET request with an HTML form. The parameters will include:

- key_terms, which can be referred to the name of a specific disease
- location, which can be referred to a specific city in a specific country.

A successful request will return a response to GET/results. Otherwise, if there is something wrong from the parameters input by the user, like the start_date is in the future, there will be an error box to inform the user to try again. A possible input is shown below.

Parameters:

Key_term: <string:>
Location: <string>

Responses:

Code	Mean
200	successful
400	invalid input
404	not found
500	internal server error

2.Retrieving data from the website

After the user has submitted a valid request, the API will search the related articles from the CDC, which is the website mentioned above, grab the related details then return the response of the GET request to the user. The data will be returned as JSON format, and with a list of report objects. Each object will have some relevant information about it, including report ID, url, date, title of the report, the disease name and other outbreak details like the number of people infected, number of deaths, etc.

Some possible interactions are shown as follows.

- **GET /reports**

Users send GET request from frontend to backend. Then backend returns all the reports from Mysql database.

Result:

[
{
"id": 1,
"headline": "\"ABCs (Active Bacterial Core surveillance)\"",
"date_of_publication": "https://www.cdc.gov/abcs/index.html",
"url": "July 19, 2021 ",
"main_text": null,
"reports": [
5
]
},
{
"id": 2,
"headline": "\"Abdominal Aortic Aneurysm \u2014 see Aortic Aneurysm\"",

```
        "date_of_publication":
"https://www.cdc.gov/heartdisease/aortic_aneurysm.htm",
        "url": "September 27, 2021 ",
        "main_text": null,
        "reports": [
        5
        ]
        },
]
```


Responses:

Code	Mean
200	successful
404	not found
500	internal server error

- **GET/reports_keyterm/{keyterm} (a specific report selected in the list above)**

  Users input with keyterm in string format (For example like:coli, fever, etc) and send GET request from the frontend to the backend. Then backend returns all the reports that contain the information of this location by filtering the reports using fuzzy search in Mysql.

  Result:


```
[
        {
        "id": 525,
        "headline": "<em>E. coli",
        "date_of_publication": "March 3, 2022 ",
        "url": "https://www.cdc.gov/ecoli/",
        "main_text": null,
        "reports": [
        5
```

```
        ]
        },
        {
        "id": 601,
        "headline": "<em>Escherichia coli",
        "date_of_publication": "March 3, 2022 ",
        "url": "https://www.cdc.gov/ecoli/",
        "main_text": null,
        "reports": [
        5
        ]
        },
        {
        "id": 1690,
        "headline": "Shiga toxin-producing E. coli (STEC) \u2014 see ",
        "date_of_publication": "March 3, 2022 ",
        "url": "https://www.cdc.gov/ecoli/",
        "main_text": null,
        "reports": [
        5
        ]
        },
]
```


Responses:

Code	Mean
200	successful
404	not found
500	internal server error

- **GET/reports_location_keyterm/{location}/{keyterm} (a specific report selected in the list above)**

Users input with location in string format (For example like: Africa, China, Japan, etc) and input with keyterm in string format(For example like: fever, Covid, etc) and send get request from the frontend to the backend. Then backend returns all the reports that contain the information of this location,

and the headline contains the input key term by filtering the reports using fuzzy search in Mysql.

Result:

[
{
"id": 1,
"headline": "African Tick-Bite Fever",
"date_of_publication":
"https://wwwnc.cdc.gov/travel/diseases/african-tick-bite-fever",
"url": "February 26, 2021",
"main_text": "African tick bite fever is a disease caused by bacteria. You can get infected if you are bitten by an infected tick.Travelers to sub-Saharan Africa and the West Indies are at risk of infection. You may be at higher risk for African tick-bite fever if your travel plans include outdoor activities such as camping, hiking, and game hunting in wooded, brushy, or grassy areas. Ticks that are infected with tick-bite fever are usually most active from November through April.African tick bite fever is the most commonly reported travel-related tick-borne disease. The ticks that spread this disease are found in sub-Saharan Africa, parts of the Caribbean (French West Indies) and Oceania (Australia, New Zealand, Melanesia, Micronesia, and Polynesia).Activities that increase a travelers\u2019 chances getting infected include camping, hiking, and game hunting. African tick bite fever occurs year-round, but more travel-related cases are reported during March through August.There are no vaccines that prevent African tick bite fever. Travelers can protect themselves from infection by taking the following precautions:Prevent Tick BitesFind and Remove Ticks\u00a0",
"reports": [
5
]
}
]

Responses:

Code	**Mean**
200	successful
404	not found
500	internal server error

- **GET/coivd**

  Users send GET request from the frontend to the backend. Then backend returns all the data about covid condition of all countries in CDC by using the data stored in Mysql obtained by scraper.

  Result:


```
[
   {
      "Country": "\"Andorra\"",
      "level": "Level 4: COVID-19 Very High"
   },
   {
      "Country": "\"Antigua and Barbuda\"",
      "level": "Level 4: COVID-19 Very High"
   },
   {
      "Country": "\"Argentina\"",
      "level": "Level 4: COVID-19 Very High"
   },
   {
      "Country": "\"Armenia\"",
      "level": "Level 4: COVID-19 Very High"
   },
   {
      "Country": "\"Aruba\"",
      "level": "Level 4: COVID-19 Very High"
   },
   {
      "Country": "\"Australia\"",
      "level": "Level 4: COVID-19 Very High"
   }
]
```


  Responses:

Code	Mean
200	successful
404	not found

500	internal server error

As for data processing and storing, we plan to use MySQL as the DBs of our API by considering its security and portability. Compared with other mainstream DBs on the market, MySQL is portable and maintains high performance at the same time. It is well established for use with python and flask for many years. Besides that, we also plan to develop our scraper as a tool which can assist us to manipulate the DBs and retrieve information from the CDC website.

Once we have the scraper developed, the scraper will parse HTML code to collect relevant information and store it in our DBs. The scraper will run to make sure all the data in DBs is timely and concurrent every time when users send requests to our API. If there was any upgrade detected on the website, the scraper can refresh the corresponding part in DBs so that users can access the newest data.

Development environment considerations

Implementation Languages

Backend

As discussed in the section “Design Details - API Module Development” our development team decided to use the combination of Python and the Flask framework for our API services. The back-end database of our choosing is MySQL.

Alternative Options For Backend

Justification for choosing is also presented in the earlier section “Design Details - The Rationale - MySQL database”.

Frontend

For front-end infrastructure, JavaScript and the React framework was chosen, as it is flexible, compatible and easy to navigate. The React framework also provides a range of external libraries built by other React enthusiasts which can significantly reduce our front-end building workload and focus on other elements of the project. The combination of HTML5 and CSS3 will also be used to create styled web templates.

Alternative Options For Frontend

1. **Implementation languages -** Other languages such as TypeScripts, Elm, ClojureScript, Amber although despite them being JavaScripts' transpiler they do not have the same level of smooth integration with our chosen framework.
2. **Implementation framework -** In terms of framework selection, we wanted to pick from the three most popular web development frameworks, Vue, Angular and React. Vue is a framework that has been growing in popularity rapidly over the recent years, it is particularly handy in building smaller and less complex non mobile applications and is easier to learn from scratch since it requires less JavaScript knowledge. Considering our frontend engineering team is rather familiar with JavaScript, and the potential of deploying a mobile variant of our application we decided to leave Vue behind. Angular on the other hand is the opposite to Vue, it is more suitable for intermediate to advanced JavaScript developers, it provides everything developers ever need to build out a full-fledged solution, but it comes with a higher learning curve and most likely bigger project size. React sits in a sweet spot between the two frameworks mentioned above, it is much more manageable than Angular, and is more flexible than Vue.

Deployment environment

Chosen Environment

Once the product is completed, tested and polished, the product will be launched as a web application using the AWS cloud hosting service. We can achieve cloud hosting by using the EC2 service provided by AWS. EC2 is a virtual machine service which allows us to host both our front and back-servers non-stop without allocating physical resources. The operating system of the virtual machine will be Ubuntu linux, as it provides maximum flexibility for users working in the development environment.

While the EC2 service sets up the portal to host our server to the public, we also need an extra layer to serve our front-end content to the public. Nginx is one of the two most common open source software that is made for web serving in the world, it is capable of reverse proxying, load balancing, media streaming and much more. By serving our web application through Nginx, it can efficiently manage the resource which ultimately means more end-users can connect to our application securely and stably at once since CPU, RAM and bandwidth of the virtual machine is better managed.

Alternative Options For Deployment Environment

Alternative options for using virtual machines were using the S3 file cloud hosting services also provided by Amazon. Although S3 provides an easier hosting pathway, it is a static file hosting service, meaning it does not satisfy our need of hosting both the web page itself and the API server. Knowing we are going down the virtual machine parth, Amazon was not the only we had either, other virtual machine hosting services include Google computer engine, Linode ana Azure virtual machines are also popular, however Amazon is the most reliable cloud provider out of all services. The major disadvantage of using Amazon EC2 comes from its pricing, since our scope of project can live inside the free EC2 tier, we will not need to worry about it hence we decided to go ahead with Amazon EC2.

Other SENG3011 team's API

After careful consideration, we have decided to integrate the Dwen API provided by the SENG3011 Dwen team to obtain a range of common diseases and their corresponding symptoms and affected areas.

External API

Our development team uses three external APIs besides our own API. The CDC API was used to fetch necessary information to populate our database. The COVID19 API was used to obtain daily global COVID19 updates. And the Google GeoCoder API was used to obtain longitude and latitude from location names in order to plot them in the frontend of the application.

High Level Architecture Recap

Architecture Recap		
Architectural Component	**Software Component**	**Software Details**
Cloud serving	AWS EC2	Virtual machine hosted at a certain remote address that allows external users to interact with our application.
	Nginx	Power web serving tool to optimize resource management
Frontend	Javascript	Most used language in the web development industry as it smoothly integrates with most web-dev software.
	React	One of the most used web development frameworks, provides rich and powerful external libraries to build out the application, lightweight and compatible.
	HTML5	The lower level web page templates used to present users with information.
	CSS3	Markup language used to add styles to the webpage.
Backend	Flask and Python	Web framework and programming language that build the communicating portal between the database and client

	Swagger Documentation	Documentation tools used to explain the behavior of our API service
	MySQL	Lightweight schema-based database that will be used to store our data entries
External API	CDC API	API provided by the CDC website to fetch information
	Covid19	API provided online through the Postman platform
	Dwen	API provided by another team in SENG3011
	Google Geocode	API provided by google to transfer address to coordinates

Benefits of our application

Our project aims to develop an application that provides our users with the most up to date information about current outbreaks and diseases that are being spread around the globe. With our target audiences being the active travelers, we wish to aid our clients in preparing for their trips and lower the risk of being infected by disease during their travel.

Key features

In order to assist our user to better prepare their trip, we are offering 4 features.

1) Pandemic report look up
We offer our client the ability to look up a particular pandemic or disease by entering a start date, end date, location and a key term. Users can enter details according to their trip and we will return a list of related articles and reports we gathered from reliable sources and display them to our users.

2) Covid-19 cases look up
Since the entire world right now is being affected by the Covid-19 pandemic, we are offering express look up for daily Covid-19 deaths, recover and confirm updates across the globe. Our users can simply scroll to look up their interested Country.

3) Country risk ranking
While we do offer a range of stats and report to our clients, a lot of our users might only be interested in whether a particular Country is safe to travel. In order to cater for this particular type of user, we are offering a risk ranking functionality, this ranking aims to give our users a high level estimation of the potential danger they could be dealing with if they wish to travel to a particular Country.

4) Disease information
While our application aims to help other clients to assess the risk of traveling to a certain Country, we can not stop our users from traveling despite how dangerous it could be.

Therefore we also aim to educate our users about common diseases including possible symptoms, affected areas etc.

Design language

As our primary target audiences are travelers, the main design language we are implementing is simplicity. We are offering direct access to most of our features through the top navigation bar, with the only exception of report look up only available in the home page.

One of the initial issues we had with design was that the application used to be extremely text heavy which is challenging for our users to stay interested for an extended period of time. Luckily, we were given feedback regarding this issue and managed to implement more visual elements such as maps and tables to increase the readability of our application.

Team organization and conclusion/appraisal of your work

Responsibilities/organization of the team

As we are a small development team made up of five members, every member is expected to contribute equally to deliver a high quality product within the given time frame.

Team composition

As discussed in the earlier section, our development team is made up of five members. The zID and name of each member are shown in the table below,

zID	Name
z5213505	Jing Deng
z5191691	Jiaxin Liu
z5212919	Ziyu Zhou
z5191561	Zhiyuan Yin
z5212995	Yilin Zhu

Responsibilities

In preparing for the project, we have divided the project into two parts, namely the “presentation section” and “implementation section”.

The “presentation section” consists of report writing, meeting notes taking, documentation building and project presentation in a demo session. This part of the project is expected to be completed by every member as we believe it is necessary to have everyone’s contribution in order to have efficient communication.

On the other hand, we are taking the “divide and conquer” approach for the “implementation section”. We have divided this section into three sub parts, “design”, “front-end development” and “back-end development”. By analyzing the interests, strengths and weaknesses of each team member, we were then able to come up with a reasonable plan of distributing responsibilities as shown in the table below.

Reports and documentations	**Backend infrastructure**	**Designs**	**Frontend infrastructure**
All members	Jiaxin Liu Zhiyuan Yin Yilin Zhu	Ziyu Zhou	Jing Deng

Ziyu Zhou being the only member on the team that is interested in frontend design was assigned to work on user interface and user design for the application.

Jing Deng being the only member on the team that has frontend development experiences was assigned to collaborate with the design team and backend development team to build out the frontend of the application.

While **Jiaxin Liu**, **Zhiyuan Yin** and **Yilin Zhu** all showed interest in building the backend infrastructure of the application, none of them have rich experience in this particular area of development. Therefore all three members were assigned to work and learn together in order to build out the backend of the application.

Project Reflection

Major Achievements

Exposure in new areas

As most of our development members did not have prior experience in the field of full stacked app development, we were able to get exposure to new areas of study and obtain valuable experience. For example, our backend development team were able to gather experience in setting up EC2 virtual machines, interacting with AWS cloud services, managing cloud databases, building web scrapers, handling necessary web request headers etc.

Project Management

Project management is an unavoidable topic in any form of group based project. Throughout the journey of the project, our development team not only had to deal with the development challenges, we also had to cater for other workload and difference in time zone which resulted in unfair work distribution and miscommunication. As the term progressed, our development team were able to pick up these issues and gradually came up with solutions to resolve these challenges as much as possible. This experience is extremely valuable in the future, as team working and communicating skills are believed to be the most valuable assets in the engineering space since modern engineering work is mostly carried out in groups.

Issues/problems encountered

As stated in other sections of the report, there were many challenges our development had to face and conquer in attempting to deliver a product that satisfied our clients.

Experience

One of the biggest challenges our development team faced was the amount of new skills or tools some of our members need to pick up before they can contribute to the project. Besides professional experiences, our development team also lacked experience in performing efficient communication and team management, which ultimately led to poor collaborations between our development team members.

Time constraint

While the given time frame specified in the requirements of the project sounds reasonable and it is in fact feasible, it did not turn out to be enough for our team. The long learning curve of some of the ability required by the project resulted in some of our members not being able to contribute to the project, the lack of efficient communication resulted in overdue milestones and inevitably elongated our development cycles.

Below is a visual representation of the amount of the time each stage of the project occupied.

	Week1	Week2	Week3	week4	Week5
Ideas brain-strom					
Forming requirements					
Deliverable 1					

UI / UX designs					
Deliverable 2					
Backend Dev					
Frontend Dev					

	Week6	Week7	Week8	Week9	Week10
Deliverable 3					
Backend Dev					
Frontend Dev					
Deliverable 4					

While the above diagram does not match exactly with the previous iterations of estimation, it is highly similar. The only two stages that took longer than expected were the front and back end implementation stages. This is due to our underestimation of the amount of time required to pick up some of the skill sets required for back end development. As a result, our development team had to extend those two development phases and ultimately work till the deadline.

CDC website

Another issue we noticed during our backend development cycle is that the cdc articles are vastly different from each other in terms of format, which made it challenging for our web scraping team to efficiently obtain data and process them.

Skills we wish we had

There are in fact a range of skills we wish we had prior to this particular workshop, below is a brief explanation to each of them.

Skill	**Description**
Web development	Web development is a big part of this particular workshop, some of the members did not have a solid enough understanding

	of the entire web request framework, which made it very challenging to build any meaningful web app.
Team managing	While we do understand this is a topic covered by a lot of the prerequisite courses, none of the courses enforce the use of team management tools such as Confluences and Jira, although some of the members had experience with them, those who are unfamiliar with it had a hard time making use of it.
Cloud services	SENG3011 is the very first workshop course that requires actual cloud hosting. While some of our members have experience in building applications under a predefined environment, they struggled to host actual applications such as SQL instances and EC2 instances.
Documentation	Communication is key to success. Without meaningful and efficient communication, we were not able to smoothly execute our proposed schedule. While some of our members have an idea of what documentation is and are capable of producing simple documentation such as user stories, they are unfamiliar with producing useful documentation that aids the development in the long run. Such as meetings notes and maintenance documents.

Potential changes if possible

As illustrated repeatedly throughout the report, there were many challenges and issues throughout the term. If our development team were to do it again, we imagine ourselves taking a vastly different approach.

Workload distribution

While the challenges come from different areas, we believed it ultimately came down to our execution. Therefore we believed if we change the work distribution, there is a higher chance of succeeding in this particular workshop.

Firstly, instead of everybody working on “presentation” and dividing the “implementation” as mentioned in the section “Responsibilities” we should have divided the project from the start.

This is because some of our members are less familiar with coding and require a longer learning period to pick up a certain skill set, therefore it would be more time efficient if we allocate those members on pure documentation.

Secondly, we should value each members' strength way more than their corresponding interest during the responsibility allocation phase. While some of the members claim to be interested in a certain field of study, they might not be able to pick up that particular skill set within the time required. Therefore, if we allocated responsibilities based on members strength we would have a higher chance of delivering milestones on time

Lastly, we should have allocated work based on members' willingness to contribute or their desired course mark. It is fair some of the students have a higher workload than others and therefore less likely to contribute to the project. As a development team we should allocate more work to those members who are aiming for a higher course mark or less loaded. By doing so, we are guaranteed to have enthusiastic members working on tasks they want to work on and therefore have a better chance of producing a product with high quality.

Roles

Another potential change we envision having is the removal of the head leader role. We found that the amount of collaboration between each team required is way more than expected, the role of head leader will only slow the communication flow which is the exact opposite of what we wanted. Additionally, as the requirements and the scope of the project is pretty much set, there is no need for a head leader to watch over each step we take. Lastly, each development team progresses along the project, our development team already builds a solid understanding of how we wish to meet and communicate, considering the size of our development team we believe our development team is capable of delivering flexible communication without a head leader.

Tech stack

As for our tech stack, we believe we made the right decisions and therefore remain the same if we were to do it again.