CPSC 2350

Group 2 Project

Langara College

Github Repository: https://github.com/kng72/cpsc2350-group2

Colin li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng
March 31, 2022

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

Contents Pages

Project Overview ---------- 3

SDLC Model ---------- 4-5

APIs Overview and Feature ---------- 6-7

APIs & UI Integration Timeline ---------- 8-9

Tests Description ---------- 10-11

CI/CD Infrastructure ---------- 12

Data Flow Diagram ---------- 13

Project Takeaway ---------- 14

WBS ---------- 15-19

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

Overview of Project

Overall the project continues with the same ideas in mind where we still have the same functionality as we started out with. The project will aim to help travellers by allowing them to see weather conditions and Covid rates in their destination country to allow them to make an informed decision on whether they would like to go or not.

The project still uses the same agile SDLC model with scrum and kanban frameworks with the added implementation of a CI/CD pipeline using Github actions. The application uses HTML, CSS, and JavaScript as the technology stack and uses Github as the hosting platform. The weather API will be used to retrieve data on the temperature, sunrise and sunset times, and cloud coverage of the city entered. The COVID API will be used to retrieve the active cases, recovered cases, and deaths for the country that the city is in; these statistics will be used to calculate the risk factor of going to that destination country.

We have revised the schedule of scrum meets due to current exam schedules to suit the needs of team members. We have set aside Fridays as our review sessions and scrum meets where we check up on the progress of the work and see if anyone needs help as well as see if any problems will arise in the future from our current plans then also complete the scrum sprint plans for the coming week. The deadlines have been set a bit more lax during the second part of this project due to scheduling issues so each part will be given a longer deadline to fit people's schedules. We have allowed for a more lenient schedule for ourselves because we already planned the details of our project beforehand during milestone 1 so we would have less research and planning to do and we would just need to focus on development.

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

SDLC Model Chosen: Agile (Scrum & Kanban framework)

Before we decided on a Software Development Life Cycle, we had to first assess each member's technical and theoretical knowledge. After a small survey on how much experience we have, we decided it was best to go with an iterative approach such as **Agile**. Agile allows for the safest approach to building something tangible given the skills that we have, and by compartmentalising features in our website, we are able to keep adding features to it whilst still having a working product.

This gives us the most leeway in assessing our progress throughout the term and adjusting accordingly. Because there are no constraints on the project's scope, if we find ourselves to be struggling in adding a particular feature, we can choose to omit that entirely without having to jeopardise having a working solution at all. Scrum in this situation would be our best bet due to the flexibility in allowing for week to week changes, decisions, and reviews from the group.

Other methodologies such as Waterfall, would require a strong foresight of what features we want and a concrete timeline, but because we are inexperienced and our scope is not well defined from the start. We will use a kanban style WSB to make a weekly to-do, in progress, checked/tested, and completed kanban board. This will allow us to discuss who does what part during weekly scrum sprint planning and review, giving us a more dynamic way of assigning jobs that are better suited for each of our skill sets.

For the Software development lifecycle for this project we continue to use the Agile structure along with the kanban and scrum framework for the basis of our planning. However, due to the scheduling needs of our team, we have decided to merge our scrum sprint planning and our project review days into one day on Friday nights. We also decided to make the deadlines laxer when it comes to development because of team schedules and due to most of the planning being completed with only development left ahead of us. Time was also allotted to test the functionality as well as using the white box method to test the code with all group members.

One of the factors that have really contributed to the success of this project has been the kanban system we have used to organize and keep track of our work and progress. The kanban dashboard allowed us to keep track of what other team members were working on and allowed us to offer help when we saw that someone was stuck on a part. The kanban system also allowed us to change our meeting schedules to having one meeting per week instead of two because we

could easily see the parts that have been completed and we could test them on our own time without having meetings.

During the earlier phases of our scrum meetings, we had agreed on creating a barebones website first and making sure that this website would run first before adding anything like colouring and accessibility. This decision is also one of the factors that allowed us to deliver a completed product on time, by having the APIs working and already having the functions to call on them we had basically completed the most important part of our project and the rest was just simple UI and accessibility designing. We also needed to change the wireframe UI that we had created earlier in the planning phase due to the format being unpleasant to use and unaesthetic format.

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

List of chosen APIs

1. Weather - https://openweathermap.org/

   The OpenWeather API, based on the information on the site proved to be the API of choice for us. It allows for JSON responses, paired with a simple API Key endpoint as opposed to having any complicated authentication procedures. The JSON data has exactly what we needed for this website as well, which are temperature, weather description, sunrise-sunset time amongst other things. The API also includes data for every city in the world, which was the strongest reason to pick this API as we are looking to cover a global scope.

2. COVID-19 - https://covid-19.dataflowkit.com/

   This COVID-19 API is interesting in the way it works. It scrapes data from websites that have COVID data to get its data, therefore it has a fully fleshed out JSON response that includes the COVID data for all countries in the world. That is particularly important to us as most COVID APIs focus on one country only, which would mean 200 different APIs to get the data of every country. Therefore this API replaces 200 of them, which makes this an easy choice for us. The data includes infections, recoveries, deaths, etc.

Implemented Features List

	Weather API	**Covid-19 API**
Features	Temperature	Active Cases
	Duration of Day	Recovered
	Cloud Coverage	Deaths

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

APIs features (3 per each API, 6 in total)

Weather API features

1. Temperature
   The website displays the current temperature at the searched location, similar to a weather website. This data includes other data such as humidity and feels like temperature which was previously planned. We have also added more weather statistics, which are Min and Max temperatures, air pressure, wind speed, and wind direction.

2. Sunrise/Sunset time (Duration of Day)
   By using the sunrise and sunset time, we can calculate how long the days are at the search location. This may be especially useful to tourists who want to escape the long nights higher up in the Northern Hemisphere. Some tourists may want this information to gauge the level of safety in an area as more daylight may mean more light for safety or other reasons such as filming opportunities, etc.

3. Cloud coverage
   The JSON response of this weather API talks about the cloud cover of the searched location. WE no longer plan on creating animation and having pictures to depict the cloud coverage, instead, we have opted to add a description to the clouds. This information may be especially useful for astronomy reasons, but also as a general understanding of how sunny it may be during the daytime.

Covid19 API features

1. Active Cases
   This data is pulled directly from the JSON response and using this we can ascertain how strong the COVID situation is in the desired destination at the current time.

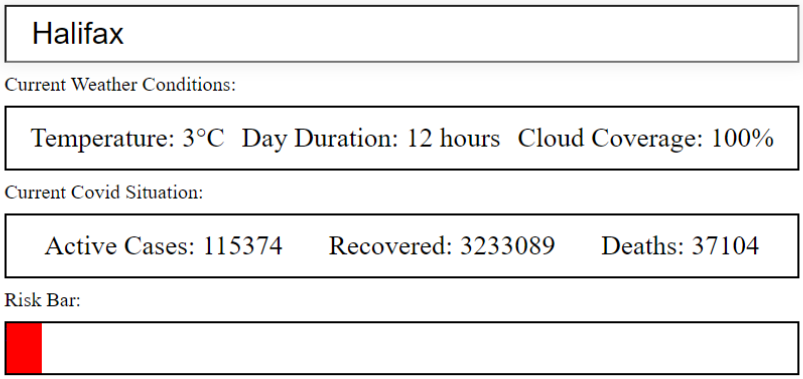
2. Recovered
   By using the recovered data, users can come to understand how good the medical services are in the area, as more recovery rates mean a higher standard of medical services and COVID policies in place.

3. Deaths
   The number of fatalities from COVID gives the user an idea of the vaccination situation in the location. A high % of fatalities could mean that a strong vaccination program is not in place, which accounts for the higher death counts.

The Covid API remains relatively unchanged from our alpha build the only difference is that we changed the risk level assessment bar to a scale of up to 5 (5 being the most dangerous).

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

API and UI Integration Timeline



The first iterations of our project revolved around getting the APIs finished first for the functionality as you can see from the image above our approach to this first alpha version of our website was to only have something working properly. As you can see in this screenshot we already had all of the planned implementations of the Covid API, which includes the active, recovered, deaths and a small risk calculator, as well as the weather API, which includes temperature, duration of the day, and cloud coverage.

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

During our process, we needed to change the wireframe that we have created in our planning phases because we found it to have an unintuitive UI and an appealing format so this is a beta of the CSS on the website with test values.

The finalized deliverable web application (depicted above) shows the full utilization of both APIs. This page includes the previously implemented functionality from the alpha version and adds more functionality for the weather API, improvements include min and max temperatures for the day, air pressure, humidity, wind speed, and wind direction. The UI has also changed in that the backgrounds have become adaptive and show pictures of the input city.

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

Test Description

White-box testing:

After we have done a huge UI improvement on our project, white-box testing was carried out to check if there are bugs in the application.

- As the sidebar which shows the details of weather was not wide enough, the description was broken down into two sentences which affected the alignment of the information. Therefore, we changed the width of the sidebar from 20 to 23.

- The minimum and maximum temperature displayed on the sidebar were in Kelvin temperature scale originally, we thought it was not user-friendly so by doing some mathematical calculations we changed the temperature scale to Celsius which is far more popular.

- The left arrow image to call out the sidebar was not shown properly. It was figured out that in the src attribute of the image element in the HTML code, there was an extra slash "/" that caused this error and we deleted it to solve the problem.

- After the user input the city name, COVID-19 statistics of the corresponding country would be displayed. However, we realized that users might misunderstand the data displayed to be just for the input city which would be scary. Therefore, we added an extra title on top of the statistics to indicate they are for the whole country.

- For some countries where the API does not have relevant COVID-19 data, our application would have displayed "NaN" in the corresponding fields. To avoid misunderstanding, we altered the code a bit so that in case there was no data received from the API, our application would display a more user-friendly sentence "no data available" instead.

- When the application was launched, the background would have been white because the user had not done any inputs. Due to aesthetic reasons, we decided to pull data from the background image API at the time the application is opened so that the interface will not look so dull.

- For the weather information, the wind direction showed only the angle. To make it more user-friendly we added an arrow next to the value of the angle to indicate its direction.

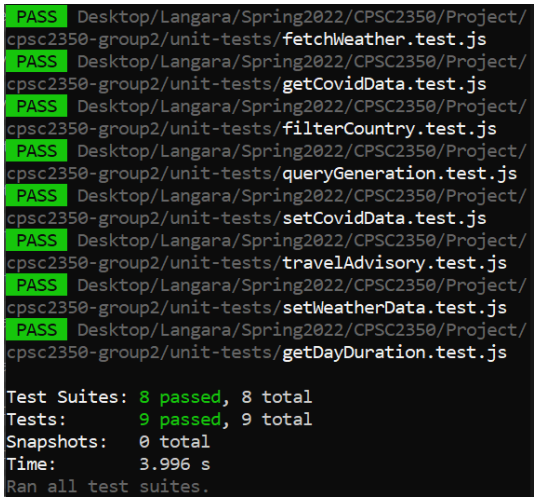
Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

Unit testing:

We have picked 8 functions from the logic.js file and have written one unit test per function. The unit tests were run by using Jest and node js on our local computers and all of them were passed.

1. fetchWeather: Test if the "feels like" temperature value can be pulled from the API object.
2. filterCountry: Test if the country name can be pulled from the API object and converted to all lower case.
3. getCovidData: Test if the number of active cases can be calculated from the number of total cases, total recovered cases and total deaths which are pulled from the API object.
4. getDayDuration: Test if the day duration can be calculated from the input sunrise and sunset parameters.
5. queryGeneration: Test if the query URL can be generated from the input city name and colour parameters.
6. setCovidData: Test if the output is correct based on different input parameters of active cases, total cases and deaths.
7. setWeatherData: Test if the calculation of temperature from Kelvin scale to Celsius scale is correct.
8. travelAdvisor: Test if the risk level value can be pulled from the API object.

Below is the screenshot of the above unit tests being passed by using Jest and node js on our local computers:



Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

CI/CD Infrastructure (test, build and deploy)

Using the Github Action feature, we built the CI/CD pipeline in the Node.js environment. Any commit or pull request made to the main branch will trigger the action, and it will automatically build and test the file and deploy it to Github Pages. Github Action provided a real-time visualizer with live logs that helped us to figure out what is not working, therefore, we can fix the problems immediately.

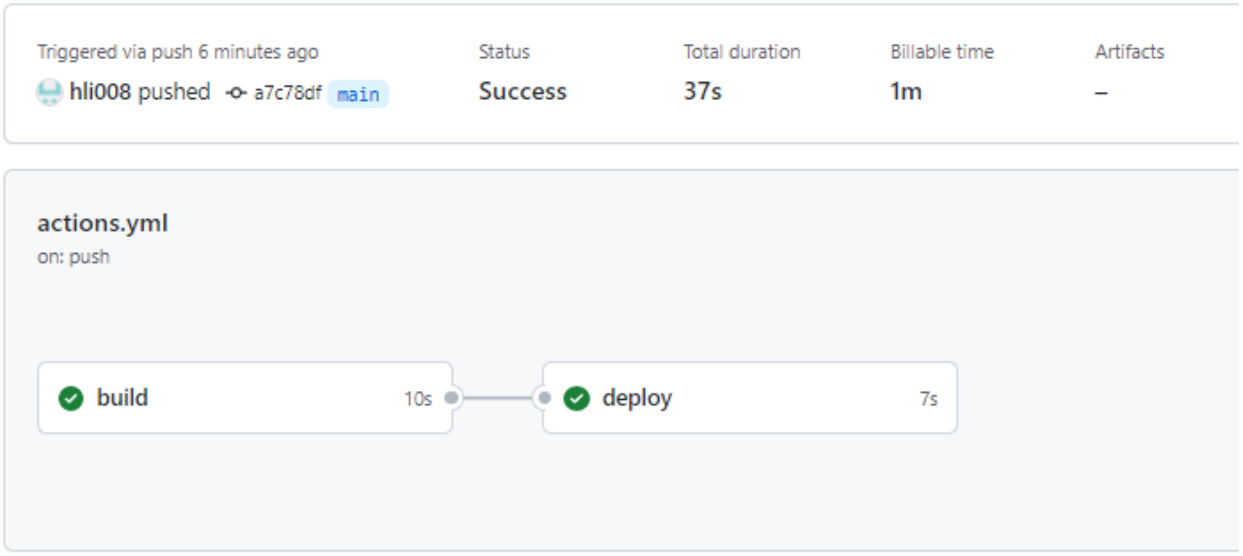
Test and Build:

By using the pre-built Node.js workflow as a template, it runs through a couple of jobs whenever a commit is pushed or a pull request is opened. It will first install and set up Node with the npm packages and dependencies, then it will run tests on the .js file, in the Jest framework.

Deployment:

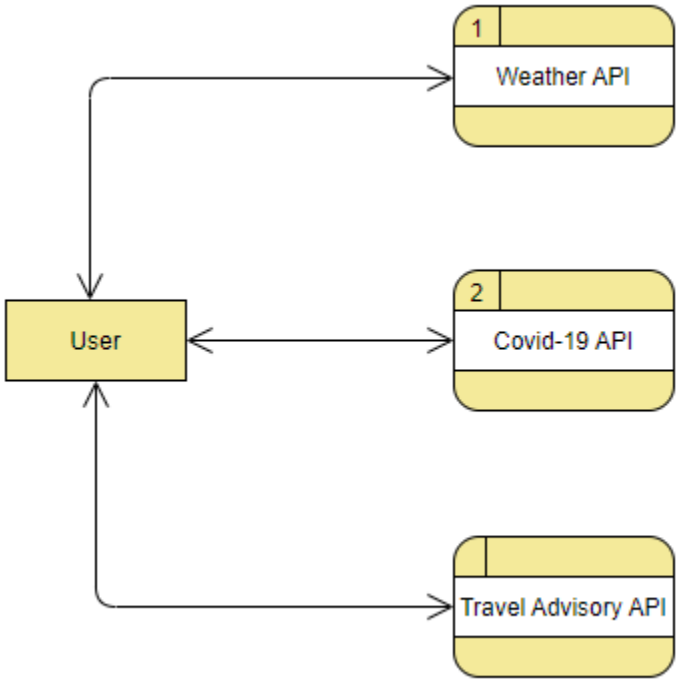
Using the build and test job as a prerequisite, the deployment stage will be triggered once the tests are successful. With the pre-built Github Pages Deploy Action included in the workflow, the project will automatically deploy to a remote branch and Github Pages.

Example Screenshot:



Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

Data flow diagrams (1 High-level)



As the user inputs the city name on the search bar, our website will retrieve it and use the weather API to search for the city name in the weather database. In return, the database will send back the weather information, including temperature, duration of the day, cloud coverage, and other weather details of the city. Our website will then display the temperature under the searching area and other details in the hidden area, which requires users to click on the sidebar.

Our website will also convert the input city name into its corresponding country name, and the COVID-19 API will scrape data from websites that include COVID-19 information about the searching country. Once the data is collected, the integrated information will be displayed on the respective area of the interface. The information includes active COVID-19 cases in that country, amount of recovered cases and deaths related to COVID-19.

Instead of following our original plan of manually calculating the relative risk of the searching country, we decided to implement another API, which provides travel advisory from authorities.

Project Takeaway

One of the greatest takeaways from this project or so-called lessons learnt is “plans can't keep up with changes”. We have planned to do something in a way according to milestone #1 but eventually, we are doing it in another way. Originally the team members have decided to meet two times a week, which is Saturday as a scrum sprint session and Wednesday as a review session. However, as the semester proceeded, each of us was packed with assignments, mid-term exams and projects. Therefore, we switched our schedule by merging the scrum sprint and review session into one day and met on Friday each week instead. On the other hand, we also make changes to the wireframe of the application. In milestone #1, we have just made a simple wireframe because we were not confident to construct complex designs. After that, we thought it might look a bit shabby, so we wanted to alter the wireframe, in order to improve our application in an aesthetic way and turned our project from college stuff to an actual product vibe. We even added an extra API for artistic purposes, all the above are the changes we made based on our plans in milestone #1. We have learnt that no matter how detailed we made our plans in advance, there are always uncertainties and unexpected events that force us to change.

Before taking this course, most of our team members were not familiar with using code hosting platforms and software practice techniques. Although we have progressively learnt about GitHub and CI/CD infrastructure throughout the semester, those have been theory-based and it was different for us when we had to actually implement our own project using these technology stacks. The actual hours for these tasks were far more than our expected ones. It was fine when we constructed our application and implemented the features, but we were badly stuck on the continuous integration and deployment of our project. We were also not used to collaborating on GitHub, for example, some of our team members were afraid of making changes on GitHub because we thought there might be a chance for the application to crash. Therefore, sometimes we have sent our HTML, JavaScript or CSS work documents to others through discord instead of GitHub. All in all, our team members were not proficient enough to use the collaboration skills taught in this course.

Please refer to the work breakdown structure below to know about more how the work was divided up and who has completed what element (including the actual report and presentation components):

Colin Li
Siam Shafiq
Ho Chun Alvin Li
Ki Hin Ng

Work Breakdown Structure (WBS)

Task order	TODO	Assigned/ Working on	Tested/ Checked	Completed	Expected hours	Actual hours
	Milestone #2					
3.0	Development of HTML, CSS, JS Boilerplate					
3.1	Base Page Development					
3.1.1	HTML Boilerplate	Siam Shafiq	Siam Shafiq	✅	2	1.5
3.1.2	CSS	Siam Shafiq, Colin Li	Whole Team	✅	2	2.5
3.1.3	JS	Whole Team	Whole Team	✅	3	4
3.1.4	Complete Base Page Development			✅	7	8
3.2	Complete Base Website Interface			✅	7	8
4.0	Functionality					
4.1	Weather Functionality					
4.1.1	Update Weather Temperature Data	Siam Shafiq	Whole Team	✅	3	3
4.1.2	Update Daylight Duration Data	Siam Shafiq	Whole Team	✅	2	2
4.1.3	Update Cloud Coverage Data	Siam Shafiq	Whole Team	✅	2	2
4.1.4	Finalise weather API features			✅	7	7
4.2	Covid Tracker Functionality					

4.2.1	Show Active Cases	Siam Shafiq	Whole Team	✅	3	3
4.2.2	Show Recovery Rate	Siam Shafiq	Whole Team	✅	2	2
4.2.3	Show Death Rate	Siam Shafiq	Whole Team	✅	2	2
4.2.4	Update travel advisory	Siam Shafiq	Whole Team	✅	2	2
4.2.5	Finalise Covid API Features			✅	9	9
4.3	Finalise Functionality			✅	16	16
5.0	Testing					
5.1	HTML Testing					
5.1.1	Validate the Code	Colin Li	Colin Li	✅	0.5	1
5.1.2	Search Bar	Siam Shafiq	Whole Team	✅	0.5	0.5
5.1.3	Complete HTML Testing			✅	1	1.5
5.2	CSS Testing					
5.2.1	Validate the Code	Colin Li	Colin Li	✅	1	1
5.2.2	Responsiveness	Siam Shafiq, Colin Li	Whole Team	✅	2	1.5
5.2.3	Color Scheme	Siam Shafiq	Whole Team	✅	0.5	0.5
5.2.4	Layout	Siam Shafiq	Siam Shafiq	✅	1	0.5
5.2.5	Complete CSS Testing			✅	4.5	3.5
5.3	JS Testing					
5.3.1	Test APIs with Different Queries	Whole Team	Whole Team	✅	5	5
5.3.2	Caching API data	Siam Shafiq	Siam Shafiq	✅	5	6

5.3.3	Test errors	Colin Li, Ki Hin Ng	Ki Hin Ng	✅	6	5
5.3.4	Complete JS Testing			✅	21	21
5.4	Complete Testing			✅	26.5	26
6.0	Deployment Preparations					
6.1	Secure API Keys in .env files	Siam Shafiq	Siam Shafiq	✅	3	3
6.2	Update Github Repository					
6.2.1	Merge branches	Whole Team	Whole Team	✅	2	2
6.3	Run the Website	Whole Team	Whole Team	✅	2	2
6.4	Complete Deployment Preparations				7	7
7.0	Presentation #2					
7.1	PowerPoint	Colin Li, Ki Hin Ng	Ki Hin Ng	✅	1	1
7.1.2	Complete power point			✅	1	1
7.2	Concept Explanations					
7.2.1	Overview of the Project	Colin Li	Colin Li	✅	0.5	0.5
7.2.2	Descriptions of APIs	Colin Li	Colin Li	✅	0.5	1
7.2.3	Application features	Ki Hin Ng	Ki Hin Ng	✅	0.5	0.5

7.2.4	Complete Concept Explanations			✅	1.5	2
7.3	Implementation Explanations					
7.3.1	Overview of CI/CD Infrastructure	Alvin Li	Alvin Li	✅	1	1.5
7.3.2	Project Takeaways and Challenges	Ki Hin Ng	Ki Hin Hg	✅	1	1.5
7.3.3	Project Demo Video	Colin Li	Ki Hin Ng	✅	0.5	0.3
7.3.4	Complete Implementation Explanations			✅	2.5	3.3
7.4	Complete Presentation #2			✅	5	6.3
8.0	Report #2					
8.1	Overview					
8.1.1	Overview of Project	Colin Li	Colin Li	✅	1	1
8.1.2	Overview of SDLC Model	Colin Li	Colin Li	✅	1	1
8.1.3	Complete Overview			✅	2	2
8.2	Application					
8.2.1	High-level List of Implemented Features	Colin Li	Colin Li	✅	1	2
8.2.2	Application Features	Siam Shafiq, Colin Li	Colin Li, Ki Hin Ng	✅	1	1
8.2.3	Complete Application			✅	2	3

8.3	Implementation					
8.3.1	Tests	Colin Li, Alvin Li, Ki Hin Ng	Ki Hin Ng	✅	6	8
8.3.2	CI/CD Infrastructure	Alvin Li	Alvin Li	✅	5	3
8.3.3	High-level Data Flow Diagram	Alvin Li	Alvin Li	✅	2	1
8.3.4	Complete Implementation			✅	13	12
8.4	Project Takeaway					
8.4.1	Lessons Learnt	Ki Hin Ng	Colin Li	✅	1	1
8.4.2	Project Challenges	Ki Hin Ng	Colin Li	✅	1	1
8.4.3	Work Division	Whole Team	Colin Li	✅	3	2
8.4.4	Complete Project Takeaway			✅	5	4
8.5	Complete Report #2			✅	22	21