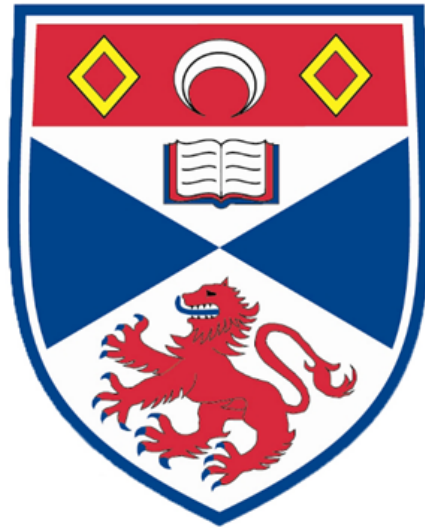


CS5003 - Masters Programming Projects

Practical 3 Joint Report



University
of
St Andrews

Word Count: 3410

Date: 08/04/2022

Group P

210031998
210002122
180009047
210033515
200008657

Table of Contents

1. Overview	3
2. Technologies & Resources	3
3. Design	4
3.1 UI Design	4
3.2 Data Structure Design	6
3.3 Function Design.....	7
4. Teamwork	13
5. Evaluation	13

1. Overview

The table below summaries the requirements we were able to achieve and who was the main person or persons responsible for meeting the requirement, called primary editors.

Requirement	Primary Editors
User Interface	210002122 210031998 200008657
RESTful API	180009047 210033515
Database	180009047 210033515
Algorithm	210002122 210031998 200008657
External API	210002122
JavaScript Library	210031998 210002122

The user interface allows you to register, login and logout as a user. On the left-hand side of the html page at all times is the user's statistics about their personal best, their latest race and their twenty-five-kilometre statistics.

There are several tabs on the user interface, below are the tab names and the functionality they allow.

Tab	Functionality
Add Record	Allows the user to add a trip. The user inputs, start time, end time, distance, type of race and the date it took place. Distance calculator and weather forecast are provided to assist users to plan their rides.
My Trips	A user can view their trips here and delete their choice of trips.
Calendar	User can view a calendar which contains parts of their trip information.
Route Recommendation	The map tab allows the user to input a start point and a destination. It can then recommend routes to the user. It displays this on a map.
Graph	This page contains an interactive graph, they user can select the type of race and a scatterplot of distance vs duration appears.

2. Technologies & Resources

Codes were amended using test editor "VS code" and tested on mac Firefox browser version 96.0.3 and window Chrome version 99.0.4844.51. NodeJS v16.14.0 was used for the development.

Axure was used for making the website design prototype (<https://www.axure.com/>).

JavaScript code was written in modern JavaScript (i.e. ECMAScript version 6+) with appropriate HTML, CSS. The server side was written in Node.js and connected to MongoDB database. The external resources included are the following:

External Resources	Resources Name	Function	Usage
API	Google Directions API	Format directions between locations	Provide distance calculator for users to input a cycling record Plot a recommendation route
	Google Distance Matrix API	Calculate travel distance and time	
	Google Geocoding API	Geocoding and reverse geocoding of addresses	
	Google Maps JavaScript API	Create and display maps	
	Google Places API	Return information about places (auto-fill-in)	
	Open Weather API	Obtain weather forecast information	Provide weather information to assist user plan their cycling
JavaScript library	D3	Create graphs	Scatter plots for race types.
	Moment.js	Manipulating date/time	Match weather forecast information with the correct date.
CSS framework	Bootstrap	Improve the UI	Appearance of Calendar

In the process of the software application development, MDN Web Docs[1], some blogs [2]–[4], some node.js tutorials[5], database tutorials [6], [7], Google Map API tutorial [8]–[10] and bootstrap tutorial [11] were read for learning purposes. The application did not adopt any code directly from the reference.

Git version control is used throughout the development as requested.

3. Design

3.1 UI Design

Since for this assignment, the lecturer didn't provide an example, so everyone did their own draft based on requirements and we discussed all versions, and all agreed on the following one.

Not all features for this design have been achieved and some extensions we have done are not included the original design, but this shows the process of decisions taken.

Links Available: <https://jhxsr.axshare.com/>

Register & Login

Considering this program is for recording personal data (Cyclist), so login and register functions are necessary. According to the advanced requirements, users can find nearby cyclists with similar interests, so when a user registers, besides basic information for athletes (Age, Gender), city and interest data is also required.

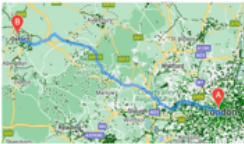
UserName	<input type="text"/>
PassWord	<input type="password"/>
<input type="button" value="Login"/>	

Given Name	<input type="text"/>
Famaly Name	<input type="text"/>
Gender:	<input type="text" value="Selection"/>
Date of Birth:	<input type="text" value="Selection"/>
City	<input type="text"/>
Interst:	<input type="text" value="Selection"/>
<hr/>	
UserName	<input type="text"/>
PassWord	<input type="password"/>
PassWord Confirm:	<input type="password"/>
<input type="button" value="Register"/>	

Add

Because all the plan and history attributes are the same, the only difference is the duration - for the plan the user doesn't have the time. So, we combined add plan and history in one add function. To meet the "Creating interactive map" requirements, we designed this radio button to enable users input origin and destination to calculate the distance.

Add	
*Date:	<input type="text" value="Selection"/>
*Type:	<input type="text" value="Commute"/>
*Distance:	<input checked="" type="radio"/> Input <input type="radio"/> Route <input type="text"/> KM
Duration:	<input type="text"/> <input type="text"/> <input type="text"/>
<input type="button" value="Cancel"/> <input type="button" value="Confirm"/>	

Add	
*Date:	<input type="text" value="Selection"/>
*Type:	<input type="text" value="Commute"/>
*Distance:	<input type="radio"/> Input <input checked="" type="radio"/> Route Origin <input type="text"/>
	Destination <input type="text"/>
	
	<input type="text"/> KM
Duration:	<input type="text"/> <input type="text"/> <input type="text"/>
<input type="button" value="Cancel"/> <input type="button" value="Confirm"/>	

Index

For the index page, because a lot of information needs to be presented, so we used a dashboard style, listing the personal and statistic data in the left side, and using a tab to show different details.

My trips – Showing all the data, we designed this based on requirements data.

Calendar – Showing data in the calendar

Map – Showing the route on the map

Nearby Cyclists – Showing the table for the cyclists in the same or close city, enabling the filters for interest.

The screenshots show the 'CYCLE LOGGER' application for user James Donan. The interface includes a top navigation bar with 'My Trips', 'Calendar', 'Map', and 'Nearby Cyclists' tabs. The 'Personal Info' sidebar on the left displays user details: James Donan, Male, Age 25, City London, Interest Golf, and a Best Record of 8.64 km/h. The main content area shows a table of trips with columns for Date, Type, Distance, Duration, Average Speed, and Action. The 'Calendar' view shows a grid for April 2022 with red highlights for specific dates. The 'Map' view shows a map of the United Kingdom with a red line indicating a recommended route. The 'Nearby Cyclists' view shows a table of other cyclists with columns for Name, Gender, Age, Location, and Interest.

3.2 Data Structure Design

User Data

To keep the data clean and simple, we used those attributes to define and record user information: username, gender, city, birthdate, password

Username is unique, so it can be used as login identification. If the user input the same user's name, it will alert error.

localhost:23360 says

this user name has been used, please change one.

OK

Other attributes are for presentation and extension features.

Record Data

Those attributes are required for a record information: Username, Date, Type, Distance, Start and end time.

Because it's personal data, so we need username for security check.

To improve the user experience, the distance can be calculated from origin and destination input.

For the duration time, to save the effort of users calculating themselves, we chose inputting the start and end time, leaving the calculation to the program.

3.3 Function Design

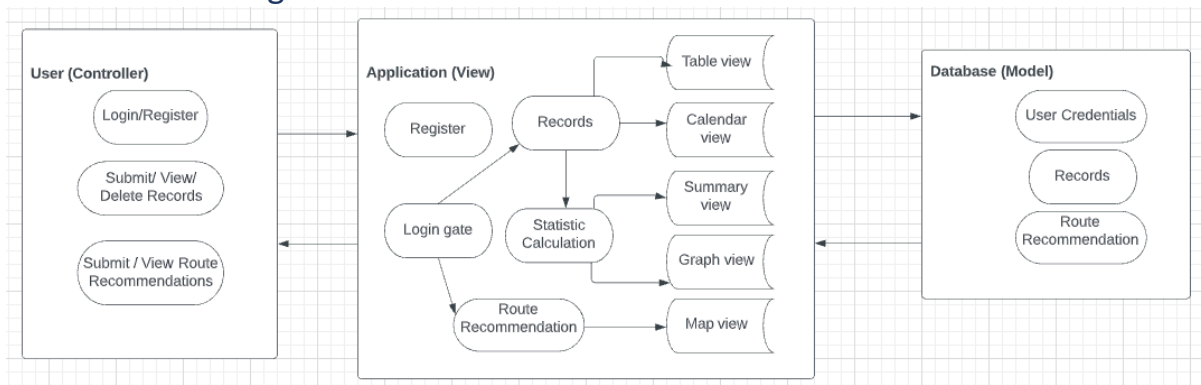


Figure 1. MVC design illustration

High-level discussion on the application structure

MVC model

Although the conventions (e.g., name convention, location conventions) of adopting a Model-View-Controller model was not used, the application was developed based on the same concept. The application was broken into three parts: the database (Model), the presentation of that data to the user (View), and the actions taken on any user interaction (Controller).

As illustrated in figure 1, the application is exclusive to logged-in users. However, users can register with the application and the credentials would be store in the database.

After logging into the application, users can access and update their private records by sending request to the application. The application would process and retrieve data from the database and construction the display based on the instruction of the users. There are 4 views available for displaying records and 1 view for displaying the route recommendation. Among the 4 views of cycling records, 2 of them are processed by statistical algorithm.

Only 3 kinds of data are stored in the database.

Focused discussion on the application features

Statistical Algorithm

JavaScript and D3 were used for this section. The Statistics Algorithms are the section of coding on the client side where the data is processed and some calculations are done to fill in the section about a user's statistics, on the left-hand side of the html page and the scatterplots in the Graph tab. In this section everything is done through function.

Displaying Statistics:

There are no bits of code outside a function, we did this to ensure that everything ran smoothly. There are various functions, to calculate duration, to filter data based on distance, to find the maximum distance cycled by the user. Various functions are called within other functions, this can be seen in the functions used to display the information.

Below is a table detailing the three functions used to display the users statsics

Display Function	Displays Dependent Functions	Information Outputted
DisplayPS	information(MAX(duradd())) <ul style="list-style-type: none"> duradd() calculated durations of each cycle in the data and adds it to the data. 	Personal Best Race <ul style="list-style-type: none"> Date of race Type of race

	<ul style="list-style-type: none"> • MAX() returns the races that had the maximum distance in the dataset. • Information() picks out the most recent race if there are multiple and creates an info object that contains everything that needs to be displayed 	<ul style="list-style-type: none"> • Start time of race • Average speed of race • Distance of race • Duration of race
DisplayLR	late() <ul style="list-style-type: none"> • late() finds the latest trip by date and creates an object with the information to display 	Latest Race <ul style="list-style-type: none"> • same as above
Display25	time25(isdis25()) <ul style="list-style-type: none"> • isdis25() firstly passed data through duradd() function. Then filters the data by distance to find trips with distance =25. Returns a dataset of trips with distance =25. • time25() takes in the dataset from above and finds the trip with the lowest duration time and then finds the one with the most recent date in case there are two trips with the same lowest duration time. It creates an object called info containing all the information to display. 	25km Race <ul style="list-style-type: none"> • Date of Race • Type of Race • Duration of Race • Average speed of Race.

Graph

This section did have some code outside a function. This was to create an svg. The JavaScript Library D3 was used to create this. The creation of the svg is stand alone code it is filled by the plot function. This function takes the user's selection of race type and filters the data by this. It plots circles for each pair of distance and duration. The axis scale changes with race type as well.

HTTP Post and Response

We used the express framework to deal with HTTP method and URL parsing.

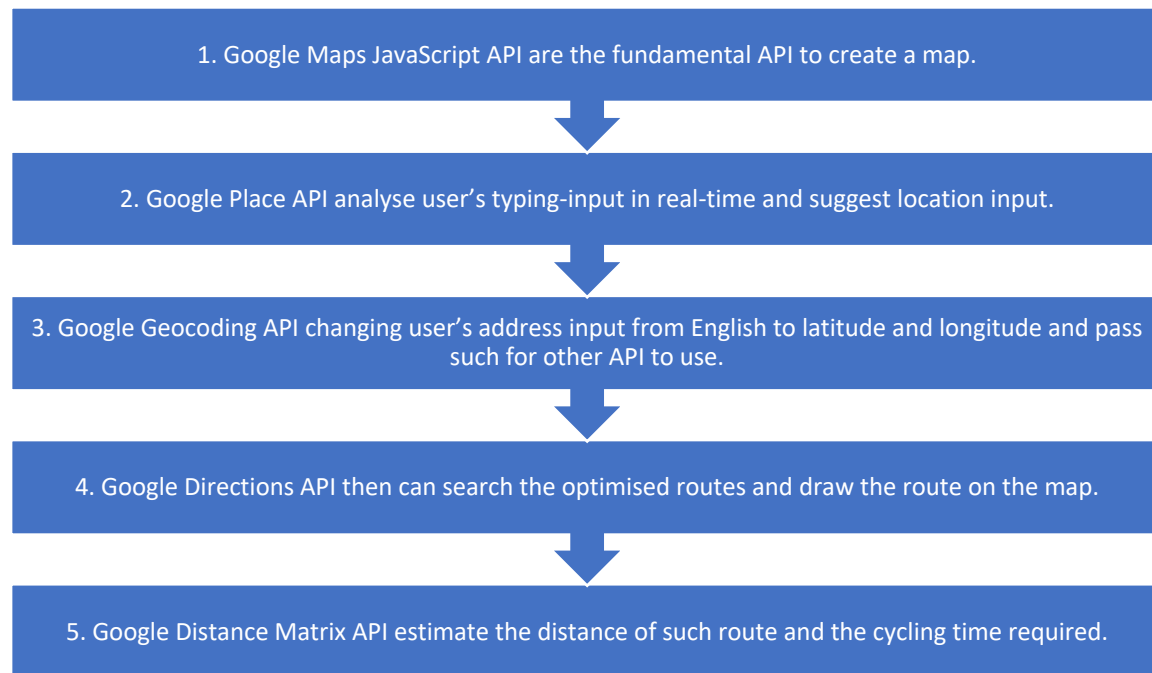
There are four main functions in server side, which are register, login, add record and delete record. Data is deleted by reaching the correct hierarchy level by comparing the current record index and the fetched id. This is an efficient method as it only locates the current record in database and delete it. Then updates again the left record in database.

Maps APIs

Brief description

In order to provide better user experience, 5 Google Map APIs were implemented to construct 2 additional features: distance calculation and route recommendation.

These APIs collaborate well to provide supplement function, as an example:



Choice of map API

This are multiples of Map API available on the market, for example, Salesforce Maps, Mapbox, Google Map API, etc. Researched showed that Google API is one of the most easiest API to set up, admin and the search result has the highest accuracy[8], [9].

Implemental Choice

For the codes to interact with the API, they are encapsuled in the `initMap()` Function and would only be called when the relative html called it. This is to avoid calling the API in unrelated pages, as the use of Google API charge after the monthly quota is used up and fetching the APIs would incur computation power.

Referring to the above graph, API 1 and API 4 are visualisation API, they are not necessary for the feature "distance calculation". Since there might be few routes from one location to another location, the visualisations can prevent the misunderstanding by displaying the exact route on the map. In addition, visualised map can improve the user interface by providing such interactive animation.

Distance Calculator

By inputting two locations, this calculator estimate the cycling distance and cycling duration

📍 Origin

📍 Destination

📍 Waypoint (if any)

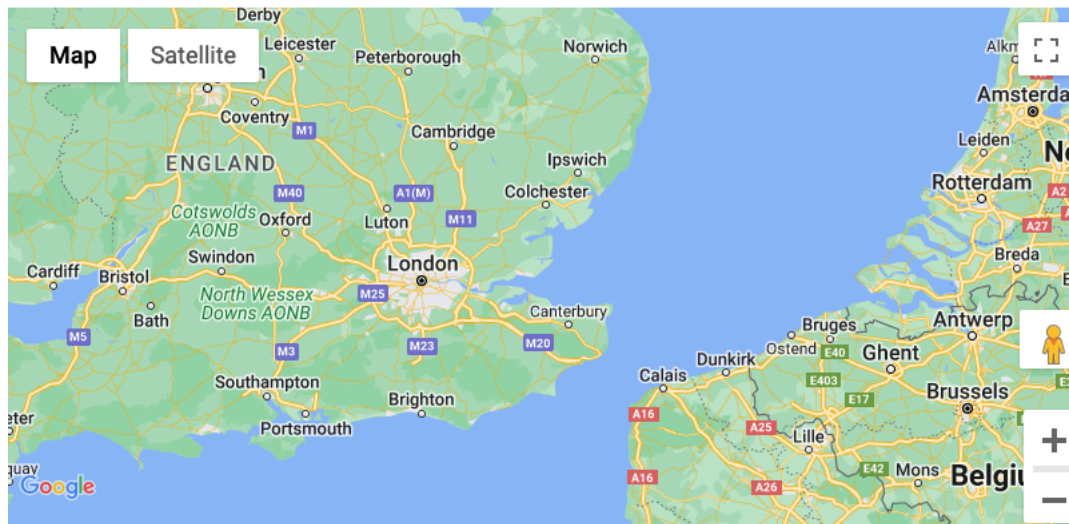


Figure 2. Distance calculator

Adding on multiple possible routes between two locations. User can enter the waypoint they wish to customise the route. Unlimited waypoint can be added as long as the user wish. To allow multiple waypoint input, and array was created to store all of user's input before formatted them into API accepting format. For route that is not possible for cycling, i.e. passing a sea, there will be a error alert as demonstrated in Figure 3.

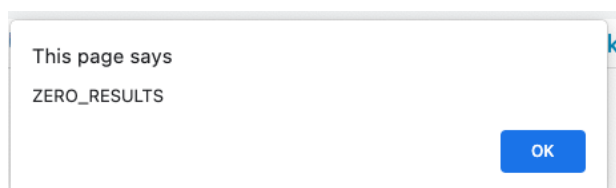


Figure 3. Searching result of cycling distance from the UK to the USA

As the waypoint specification function is added, the map is set to show only one optimised route instead of multiple routes and allow user to choose. Additionally, the unit system is set to be KM for consistency purpose.

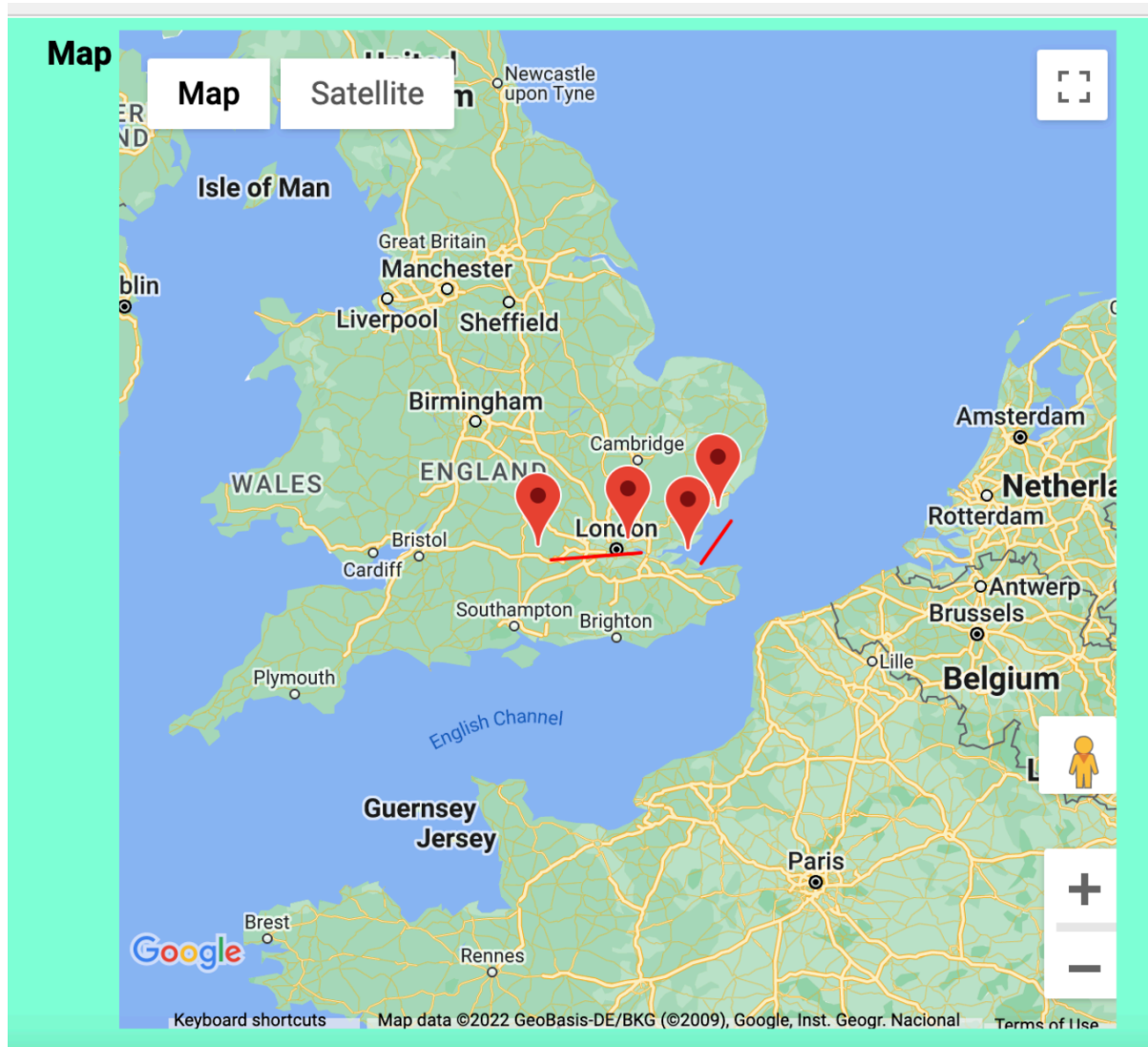


Figure 4. recommendation design one

For the route recommendation features, 4 of the above mentioned Google API were used again to show the possible routes to explore for the users. One big challenge was how to draw multiple routes on the same map as the API default refresh before loading the next route. It was solved by avoiding using the `google.maps.directionsDisplay()` function and switch to function `google.maps.DirectionsRenderer()` and pass an array into such pure drawing function.

Figure 4 was the first prototype of showing the recommendation route. That is pint-pointing the start and end location by API function `marker()` and draw a polyline between them by `polyline()`. However, the polyline is inflexible to fit into the map and the pint-point would dislocate upon zooming.

Calendar

Apr 2022						
Navigate To:		Apr	2022			
Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3 ultra-distance 12 km 12:12 - 15:15	4 ultra-distance 12 km 12:12 - 15:15	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
< Previous		Next >				

Figure 5. Calendar view

Brief description

A calendar interface allows users to have a clear view of their past and future cycling records/plans.

Implemental Choice

Google Calendar was one of the alternatives to create the calendar interface. Despite the strong functionality of the Google Calendar API, it requires the user to have an google account, which create unnecessary dependency to the users to use this function.

Thus the table is created by manipulating the 'table' HTML element. The first date of each month would be detected once user want to navigate to such month, in this way, the structure of the map do not need to hard-code. The days in each month are calculated by the manipulation of date. One alternative could be setting up an array to store how many days there is in each month. However, since the calculated method is easy to implement, it was adopted.

Another interesting point here is creating an id for each box of the table. That is to allow the data array obtain from database can search for their responsive box and append on it. For consistency purpose, slice method was used to make sure the date format is YYYY-MM-DD.

Upon loading, the function new Date() is used to capture today's date, month and year. So users can view the monthly-calendar when they first clicked the tap. An alternative could be setting certain month and year as "selected" in html. However, it would require monthly update.

Weather API***Brief description***

Weather API is used to fetch the weather information for the future 7 days and is included in the “Add record” tap to help users to plan their cycling.

Implemental Choice

There are numerous weather APIs available in the market. OpenWeather API was chosen as it is responsive, reliable, and easy to implement.

The API could provide information on the weather in by minutely, hourly or daily. As the aim of include this API here is to allow users to pick a day to cycle, thus only daily weather information is requested.

Then the information is processed and append to a table in the DOM.

4. Teamwork

Approach to work

Our group adopted a combination of plan-driven and incremental development processes. At the beginning, we meet on teams every time. Initially we got together to decide on a sketch of the appearance, interaction, features of the product and created a prototype to demonstrate how the final product would be. This was our aim for the project.

Tools and techniques

Version control and merging tools: git, kdiff3.

Communication tool: Microsoft teams.

Method of job allocation and progress tracking

In our second meeting shortly after the sketch was decided upon, we allocated the work. Each of discussed our strengths and weaknesses, telling everyone our preferences. The work was split and we each began to work on our sections.

Ever few days we would have a call on Microsoft teams to discuss where we were along the project route. We attempted first to complete the more basic requirements such as computing the user's statistics.

We kept going like this until all the sections were complete. For the client side development, a common imaginary array was created to make sure everyone develop their code compatible to the format will be stored in database. In such way, client side can test their code vigorously before combining with the server side, and server side can understand what client side is expecting.

When the deadline is approaching, some team members meet in person to assist each other and conduct pair-programming for some of the difficult logics.

Up-skilling of team members

All of our team members gained a comprehensive understanding of how applications operate on both the front-end and back-end, including how to work with a MongoDB database and how to create handlers to connect to the event-based RESTful API.

5. Evaluation

Things that went well

Visualised Planning

The layout and logic flow of the application was visualised using prototype tools. Such prototype acted as a guide and goal for the team of what the finished product would look like and function. This also provide a basis for the team to discuss the work allocation.

A common testing dataset were creating before the serve side and database connection has been set up. This allows the team to work and test parallelly, which help to isolate the possible location of the errors.

Nice teamwork

Team members are assigned specific responsibilities based on their experience and strengths. For example, teammates with more frontend developing experience were responsible to develop the html, css and client side JavaScript, teammates enjoying learning new things are responsible to develop the fancy features, e.g. Map, graphics, calendar etc.

The group work as team to strike for a common goal. Despite the duties has been divided among everyone, teammate are willing to help each other outside their pre-set duties.

Good documentation

In each meeting, we have documented the progress and next steps in detail. This prevents misunderstandings and provides a reference point when results do not meet expectations. As a result, our communication is efficient and effective.

Expectation management

In the kick-off meeting, we discussed our expectations for this project. So, everyone can reach a consensus on how much effort we should devote.

Things could be improved

Keep the user data when the user refreshes the page

Because the add record function is in a new HTML, when the record added, the data can be successfully inserted into database, but when goes back to the index.html page, user data cannot be gained, all other functions including records, calendar, graph requires user information. So the user must login again. We have searched many solutions and still failed to solve.

Nearby cyclist function

In the original design, we intended to finish this feature, so we required the interest and city data, for the distance comparison and interest match. However, due to the time limits, this feature has not been achieved.

User-friendly Interface

For the adding record page, we intended to use a radio to enable users choosing to input a value for distance or choosing routes inputting location to calculate distance. So, for one option, the other one input should be hidden. But when implementation, we found if hiding one, another one will lose data. And for the weather, we intended to represent seven days after the chosen date but resulted to show seven days after current date. We believe if more time could be put into this project, all this can be solved.

Time management

It took teammates some late nights to complete the project well before the deadline. As the project specifications were published a month ago, the project could be started sooner.

Fail to follow the plan

A schedule was created to set the milestone and internal deadlines for each part. However, and understandably, some internal deadline was missed.

Module pattern and closure

The module pattern can be used to wrap the variables and functions together in a single scope. By defining objects and specify the variables and the functions that can be accessed from outside the scope of the function.

What might you do differently next time or if you had more time

Quality of code

Code quality can impact the overall performance of the application, for example, how responsive, secure, or reliable the application could be. For example, in the code, there is occasion that double for loop is used. This can deteriorate the performance of the code.

Code documentation

Documentation is important in coding, especially in a collaborative coding work. JSDoc could be used to add documentation comments directly to the source code, for example, to namespaces, methods, parameters that are used. So other teammates can have better understand on what resources has been used and the dependencies of the functions. This help to manage the complex project.

Automated unit testing

A frustrating incident happened to us a few times: small changes in the code break the whole application. Thus we can write tests as we write code and run automated test regularly. For example, Mocha and its library Chai can be adopted to describe the expected results of the code and let automated testing tools to test the response of the code. This can improve the application's robustness of the of running asynchronously.

End-to-end API testing can be conducted by curl to test the response of the application upon user authentication, HTTP post, etc.

References

- [1] MDN Developers, "Canvas tutorial - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial (accessed Mar. 18, 2022).
- [2] Red Hat, "What is a REST API?" <https://www.redhat.com/en/topics/api/what-is-a-rest-api> (accessed Mar. 18, 2022).
- [3] Cod pen, "kanban board with html, css and js." <https://codepen.io/karthikdevarticles/pen/PopxPwO> (accessed Mar. 18, 2022).
- [4] Prashant Ram, "A simple Registration and Login backend using NodeJS and MySQL | by Prashant Ram | Medium." <https://medium.com/@prashantramnyc/a-simple-registration-and-login-backend-using-nodejs-and-mysql-967811509a64> (accessed Mar. 18, 2022).
- [5] OpenJS Foundation, "Guides | Node.js." <https://nodejs.org/en/docs/guides/> (accessed Mar. 18, 2022).
- [6] Quackit, "Database Tutorial," 2022. <https://www.quackit.com/database/tutorial/> (accessed Apr. 08, 2022).
- [7] Tutorialspoint, "MongoDB Tutorial," 2022. <https://www.tutorialspoint.com/mongodb/index.htm> (accessed Apr. 08, 2022).
- [8] "Top 10 Google Maps API Alternatives 2022 | G2." <https://www.g2.com/products/google-maps-api/competitors/alternatives> (accessed Apr. 08, 2022).
- [9] Google Maps API, "Google Maps API Reviews & Ratings 2022," 2022. <https://www.trustradius.com/products/google-maps-api/reviews> (accessed Apr. 08, 2022).
- [10] Google, "Tutorials | Maps JavaScript API | Google Developers." <https://developers.google.com/maps/documentation/javascript/tutorials> (accessed Apr. 08, 2022).
- [11] Nick Schaferhoff, "Bootstrap Tutorial - How to Set Up and Use Bootstrap (Beginners)," 2020. <https://websitesetup.org/bootstrap-tutorial-for-beginners/> (accessed Apr. 08, 2022).