

# Final Report

SENG2021 Requirements and Design Workshop

D5 Deliverable: Monday 27th April 2020

## **Group: We Push To Master**

Chinmay Manchanda	z5216191
Marina Reshetnikova	z5285381
Michelle Seeto	z5061204
Xifei (Cecilia) Ni	z5173159
Ye (Max) Wo	z5215628

## Background

Currently, we as users need to browse different websites to find events or activities that intrigue us. Unfortunately, users often succumb to the easy option of staying at home, either not wanting to put in the labour to find an appropriate activity or overwhelmed by analysis paralysis.

Our web application, 'Hello World' aims to simplify this process of choosing an activity for the day, and in doing so, encourage a more active and engaged approach to one's daily life. 'Hello World' fetches event details from across the internet and provides users with a curated list of events and activities they can enjoy, automating what would otherwise be an uninteresting and often unfruitful process.

Ultimately, we are seeking to provide an instant response to the question, "What should I do today?", saving users time and optimising results for weather, time of day, preferences and mood.

## Part 1: Requirements Analysis

### Problem statements

- 1) There is an overwhelming number of event platforms
- 2) Data is not shared between event platforms, so users must visit multiple sites for decent range of choice
- 3) The quality of a user's choices depends entirely on their knowledge of existing websites
- 4) There is no general, unified platform for events, activities and places to visit, only specialised platforms that offer results of a specific format

- 5) Most event platforms offering filtering by time and category, but not weather
- 6) Often search results will contain activities that are inappropriate for the current weather and time of day

## Features

- Display list of events and activities happening today.
- Display summary of an activity or an event.
- Redirecting to respective websites for payment and further info.
- Filter activities based on weather.
- Filter activities based on the user's moods.
- Dynamic backgrounds based on weather and time.

### COVID-19 Specific Feature:

- Display list of local businesses to support during COVID-19.

## User stories

### User Story 1

**Feature:** Display list of events and activities happening today

**As a:** Event attendee

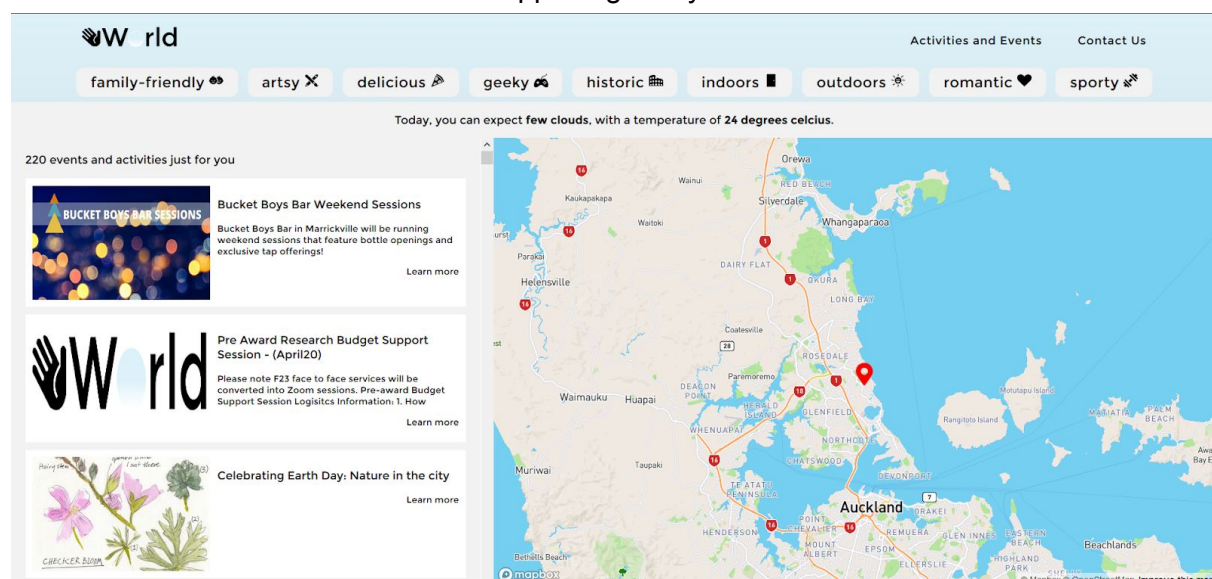
**So that:** I can attend events in my spare time

**I want to:** View a list of events suitable for the current weather conditions happening today

**Given** I am on the main page

**When** I press the “Activities and Events” button

**Then** I should see a list of all events happening today



**Figure 1.** Interface of ‘Events and Activities’ page

## User Story 2

**Feature:** View local eateries during COVID-19

**As a:** Event attendee

**So that:** I can support local eateries which have been heavily impacted due to COVID-19

**I want to:** View a list of local eateries

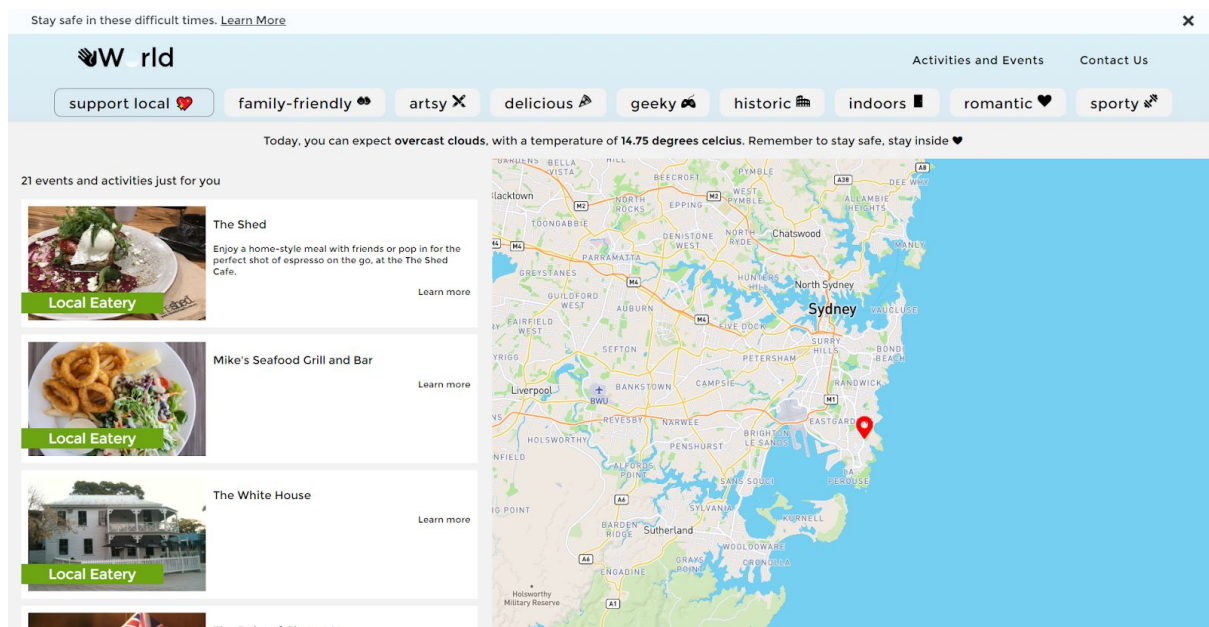
**Given** I am on the main page

**And** there is a pandemic going on

**When** I select the “support local” button

**And** I press the “Let’s go” button

**Then** I should see a list of all local eateries which I can support



**Figure 2.** Interface of ‘Events and Activities’ page in COVID mode

## User Story 3

**Feature:** Display summary of an activity or event

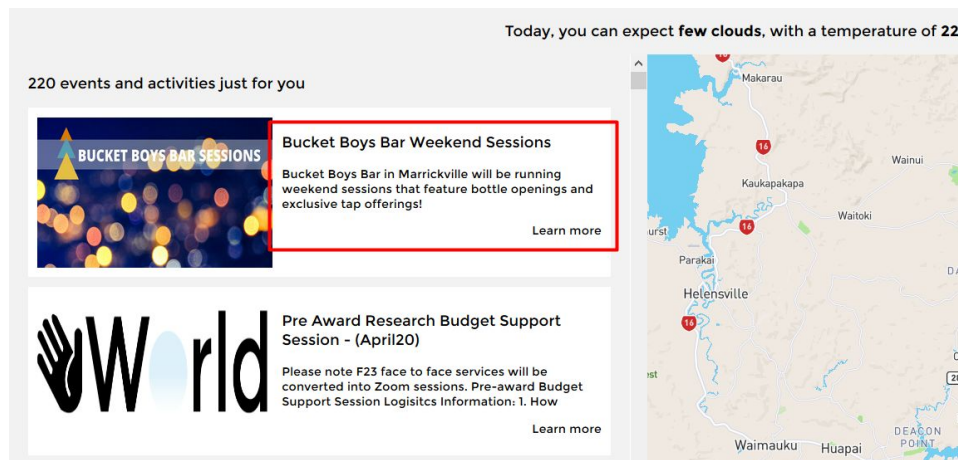
**As a:** Event attendee

**So that:** I can gain a brief understanding of an activity or event

**I want to:** View the summary of an activity or event

**Given** I am on the event page

**Then** I should the summary of each activity or event



**Figure 3.** Event/activity display card

## User Story 4

**Feature:** Allow access to the original source of an activity or event

**As a:** Event attendee

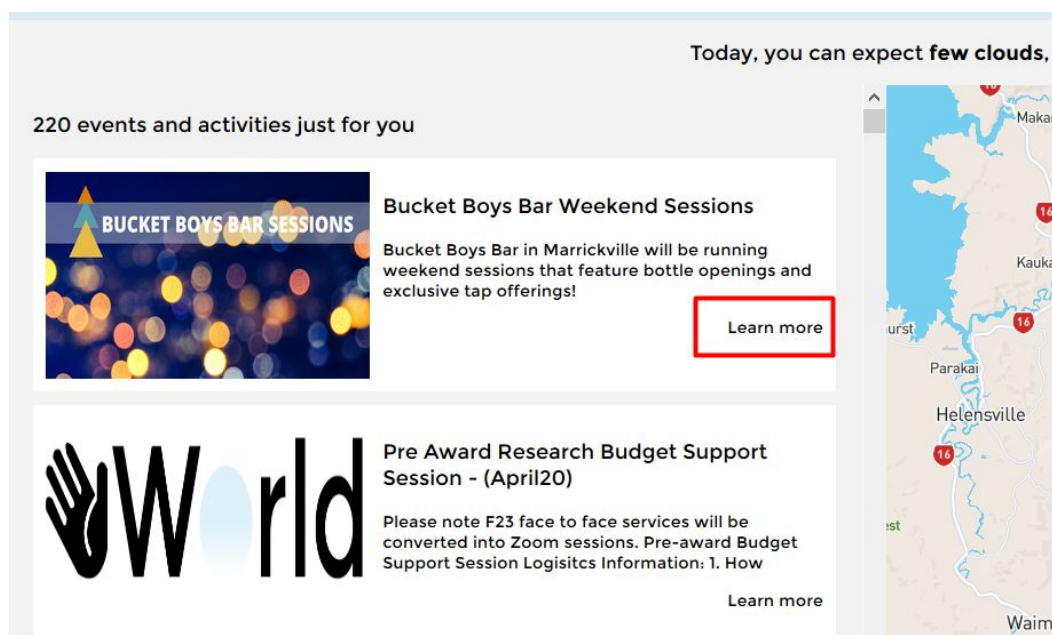
**So that:** I can read more about and pay for an activity or event

**I want to:** Be able to read more about and pay for an activity or event on the platform that it's from

**Given** I am on the event page

**When** I click the "Learn more" button on an activity or event

**Then** I should be redirected to the respective websites for further information about and pay for the event



**Figure 4.** 'Learn more' button redirecting to external event platform

## User Story 5

**Feature:** Filter activities or events by mood

**As a:** Event attendee

**So that:** I can attend events more suitable for my current mood

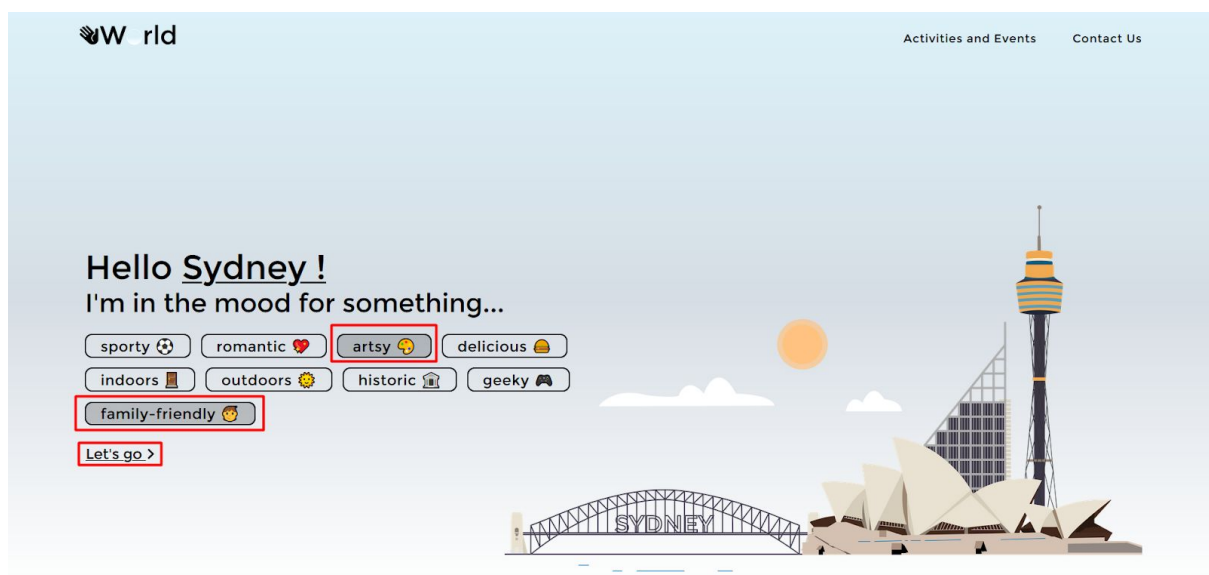
**I want to:** Be able to filter activities or events by my mood

**Given** I am on the main page

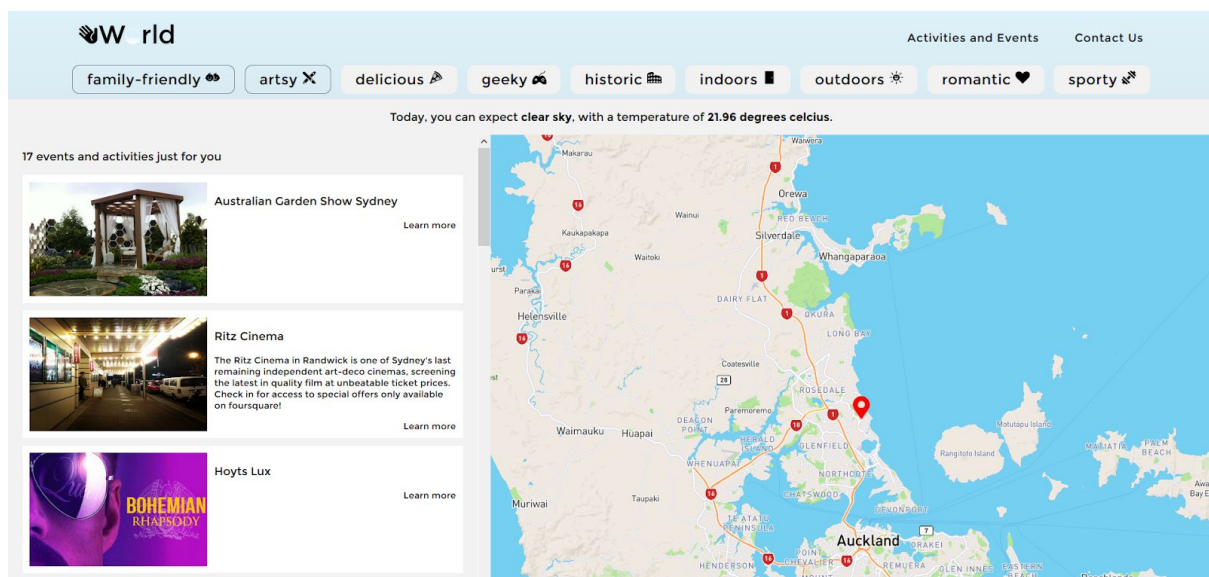
**When** I select the buttons that correspond to me current mood

**And** I press the “Let’s go” button

**Then** I should see a list of all events happening today which correspond to my mood



**Figure 5.** Interface of home page with mood tags selected



**Figure 6.** Interface of ‘Events and Activities’ page with corresponding results



## User Story 6

**Feature:** Display time and weather dependent backgrounds

**As a:** Event attendee

**So that:** I can see a personalized background for the current time or weather

**I want to:** Be able to see a background that reflects the current time or weather

**Given** I am on the main page

**Then** I should be able to see a background that reflects the current weather and time

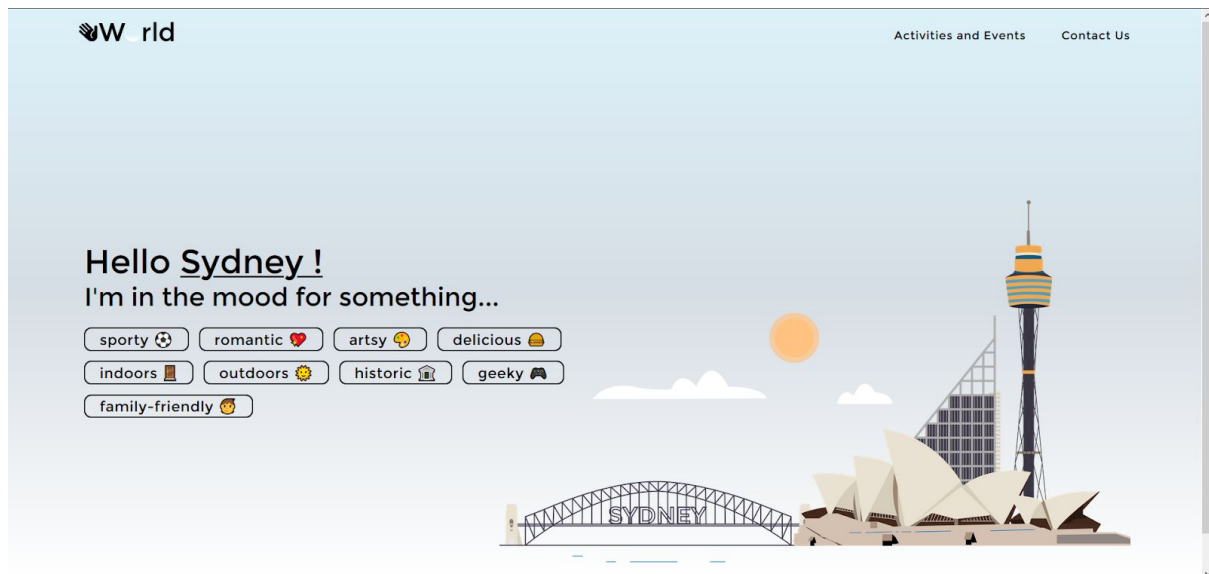


Figure 7. Background for daytime

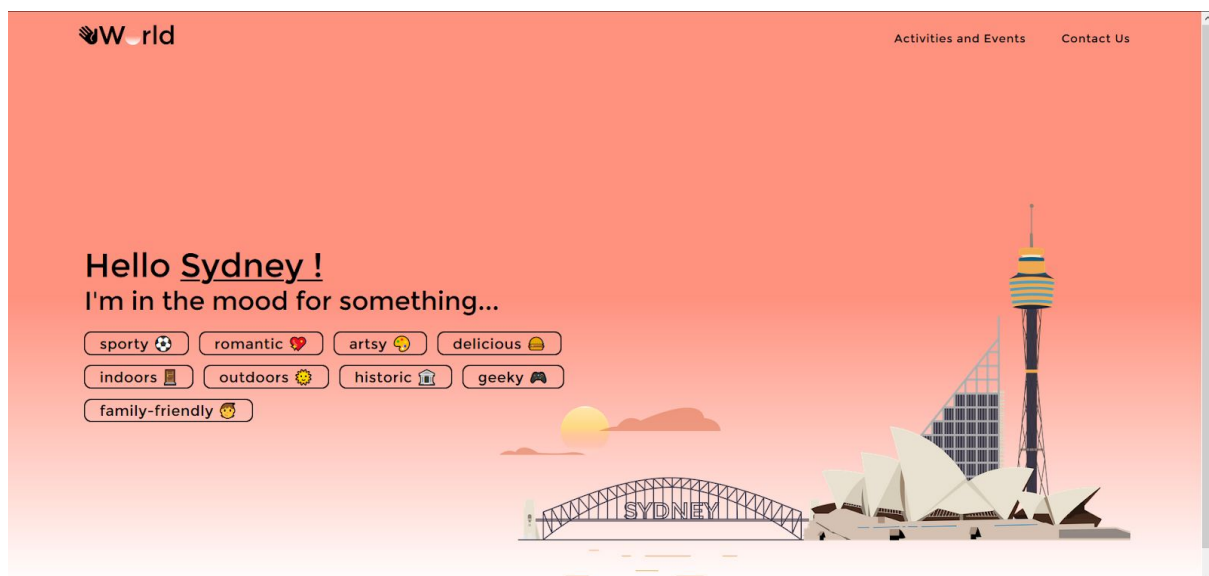


Figure 8. Background for sunset



Figure 9. Background for rainy weather



Figure 10. Background for sunrise

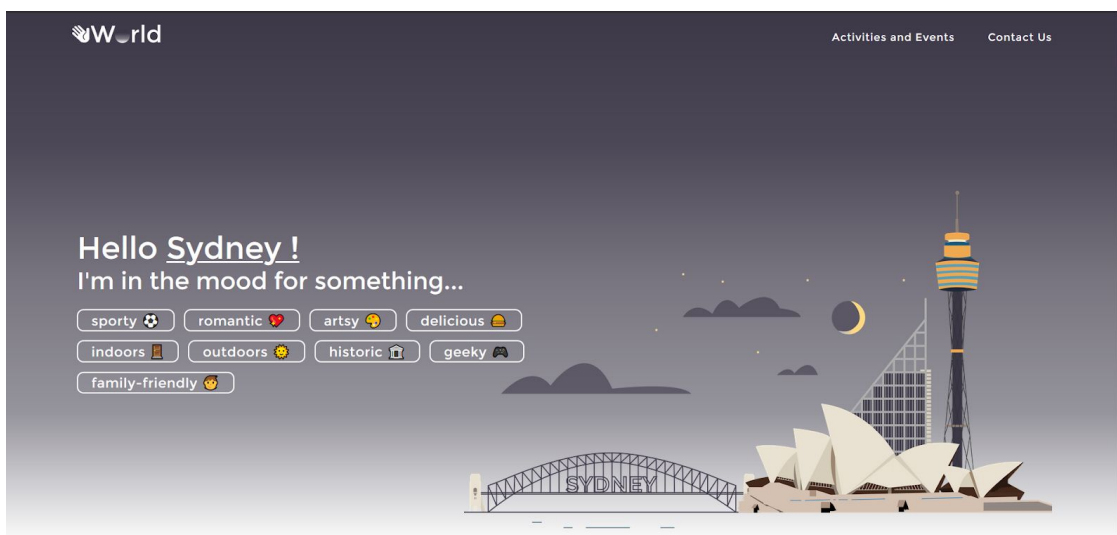


Figure 11. Background for evening

## User Story 7

**Feature:** Adjust UI layout according to different screen type/size

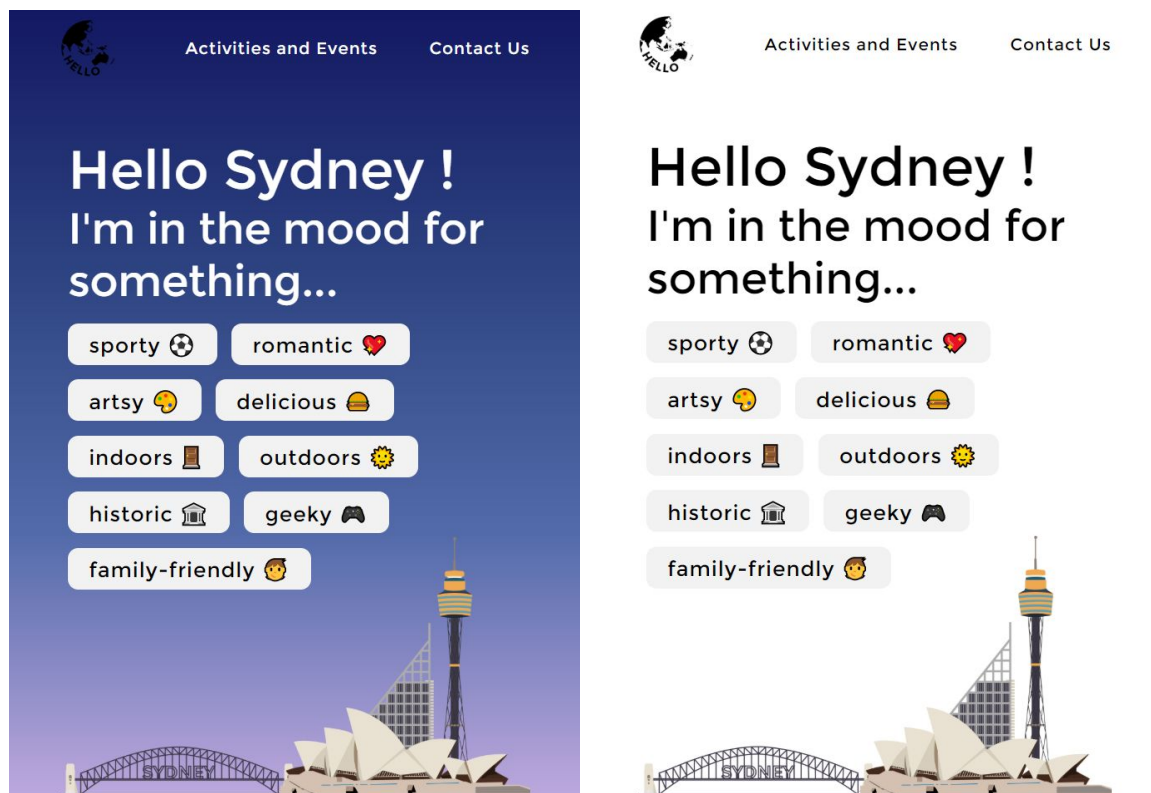
**As a:** Event attendee

**So that:** I can use this website on different devices

**I want to:** See a user interface that is responsive to different types of screen

**Given** I am on the main page

**Then** I should be able to see the website



**Figure 12:** Home page layout for smaller screens.

## User Story 8

**Feature:** View the location of an activity or event on a map

**As a:** Event attendee

**So that:** I can determine the accessibility of an activity or event

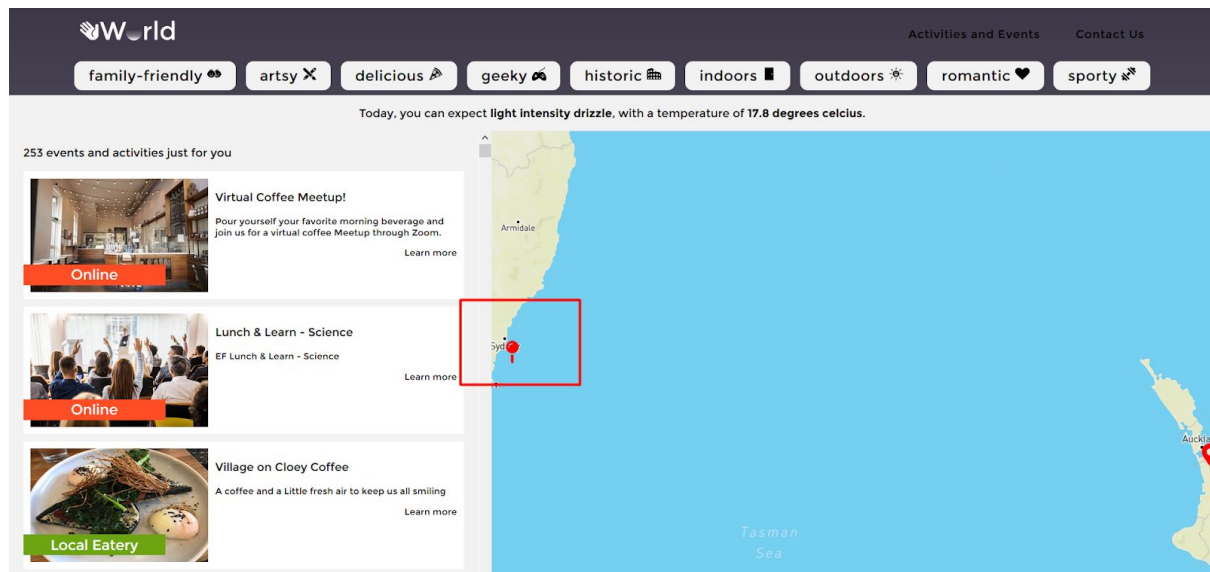
**I want to:** see where an activity or event is

**Given** I am on the results page

**And** I click on an event

**Then** I should be able to see its location on the map





**Figure 13:** Results page with the location of “Village on Cloey Coffee” on the map.

## Part 2: Software Design

### Software architecture

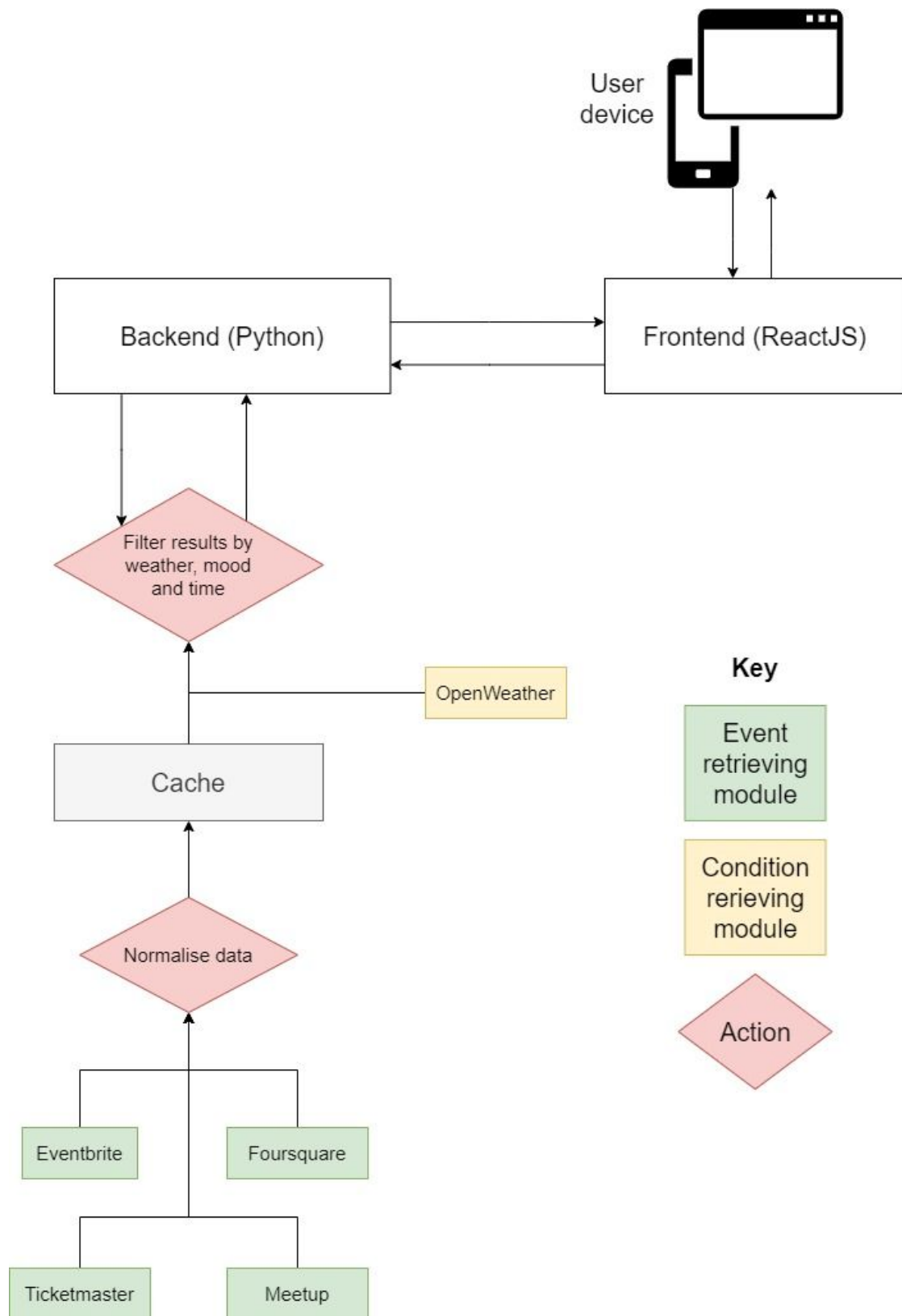
Our software consists of clearly separated backend and frontend tech stacks. Figure 13 illustrates the overall architecture, including separate components that were developed. We took this approach as it meant the backend and frontend could be developed in isolation and were fully decoupled, with all communication occurring through an API. This increases the scalability and usability of our system, as our software components operate largely independently of one another and can be used as modular units.

Our backend components were written in Python, with a Flask framework. Python was chosen for its extensive libraries and most importantly for its web framework Flask. Furthermore, all team members are familiar with Python, which meant that the project did not experience any learning pains or setbacks associated with learning a new programming language. Flask was chosen as this is a relatively small scale project and there was no need for the extra functionality offered by Django or for database integration. Flask offered the desired functionality with simple implementation while maintaining the ability to configure more extensively if necessary in the future.

**Table 1.** Event format used by the backend.

Property	Type	Example
event_id	string in format [data source]-[id]	EVENTBRITE-1234
name	string	Pilates in the Park
start_time	integer (unix timestamp)	1587546000

end_time	integer (unix timestamp)	1587556800
location	dictionary { latitude: float, longitude: float}	{ latitude: -33.8, longitude: 151.2 }
organiser	string	Mind Body Yogi
price	float	10.5
price_tier	integer	0
is_online	boolean	false
summary	string	An outdoors pilates class, to tone up the mind and body in the fresh spring air.
description_html	string containing HTML	<b>Get toned for summer!</b> <p>Join us for Pilates in the Park, led by renowned Pilates instructor Mel Blanche.</p>
tags	array of enum(indoors, outdoors, romantic, artsy, geeky, history, hungry, family-friendly, sporty)	["outdoors", "sporty"]
rating	float	4.7
url	string	<a href="https://eventbrite.com/pilates-in-the-park">https://eventbrite.com/pilates-in-the-park</a>



**Figure 13.** Software architecture

The backend consists of data retrieving modules that are responsible for interacting with external APIs. We use FourSquare, MeetUp, Eventbrite and Ticketmaster for event data, as

these organisations are popular for event listing and offer reliable, accessible and well-supported APIs. In Figure 13, event retrieving modules are highlighted in green. Each module has knowledge of only one API, and is responsible for making requests to that API, taking care of any quirks or unique features, and normalising the retrieved data in the format illustrated in Table 1. As an example of an API's individual 'quirks', Eventbrite does not provide a search or event list functionality through its API. As such, we must first scrape the Eventbrite website for event ids, which we can then use to make individual calls. This is unique to the Eventbrite API and is thus handled by its respective module. For web scraping, the module uses the BeautifulSoup library.

As such, our backend is not coupled to our choice of data source. By abstracting third-party interaction to individual modules, we are able to support various interfaces as well as easily add new data sources; the modules are implemented separately and according to their API, but are guaranteed to return data in a format recognisable and consistent with our backend.

Additionally, we retrieve weather information from OpenWeather, highlighted in yellow in Figure 13. This was chosen as it is highly reliable, fast and provides an extensive suite of information: weather conditions, temperature, humidity, sunrise/sunset times etc.

In order to minimise APIs calls and stay within rate limits, for example the limit of 500 daily premium calls to FourSquare, we implemented a cache that stores API data for at most 24 hours, illustrated in grey in Figure 13. Caching and data persistence was achieved through pickling files, however this could be easily replaced with a relational database with minimal changes to our backend system. Once the data has been normalised and cached, it is filtered using the backend's time, weather and location filters, as well as using the mood filters retrieved from the frontend. This allows the backend to seed out all events that do not meet the current conditions and user's criteria.

Our frontend is written in JavaScript with the React framework. JavaScript was chosen because group members are reasonably familiar with the language, and we decided to use React for its industry relevance. We will be able to utilise the skills and experience we have gained in this project in future industry endeavours. Following the spirit of React, our frontend is made up of reusable components that were developed in isolation and combined to form our interface. Given the complexity of our user interface, this allowed for rapid development and reasonable division of the work.

One of the most essential frontend considerations is state management, for which we follow a top-to-bottom data flow. The selected filters (artsy, sporty, indoors, etc.) are tracked as a React state at the very top level of our React application. This state is passed along to the bottommost unit, the UI elements itself. Upon the user's click event, the UI elements hook back to the top-level component, and therefore notify it of the change (Figure 14).



**Figure 14.** Data flow in frontend

This data flow model allows us to have a single source of truth across the application, in this case, the state stored in App. Moreover, we benefit greatly from this two-way data binding because React has built-in optimisation for such flow and can improve the general performance of page loading a lot.

#### Interaction with the Backend

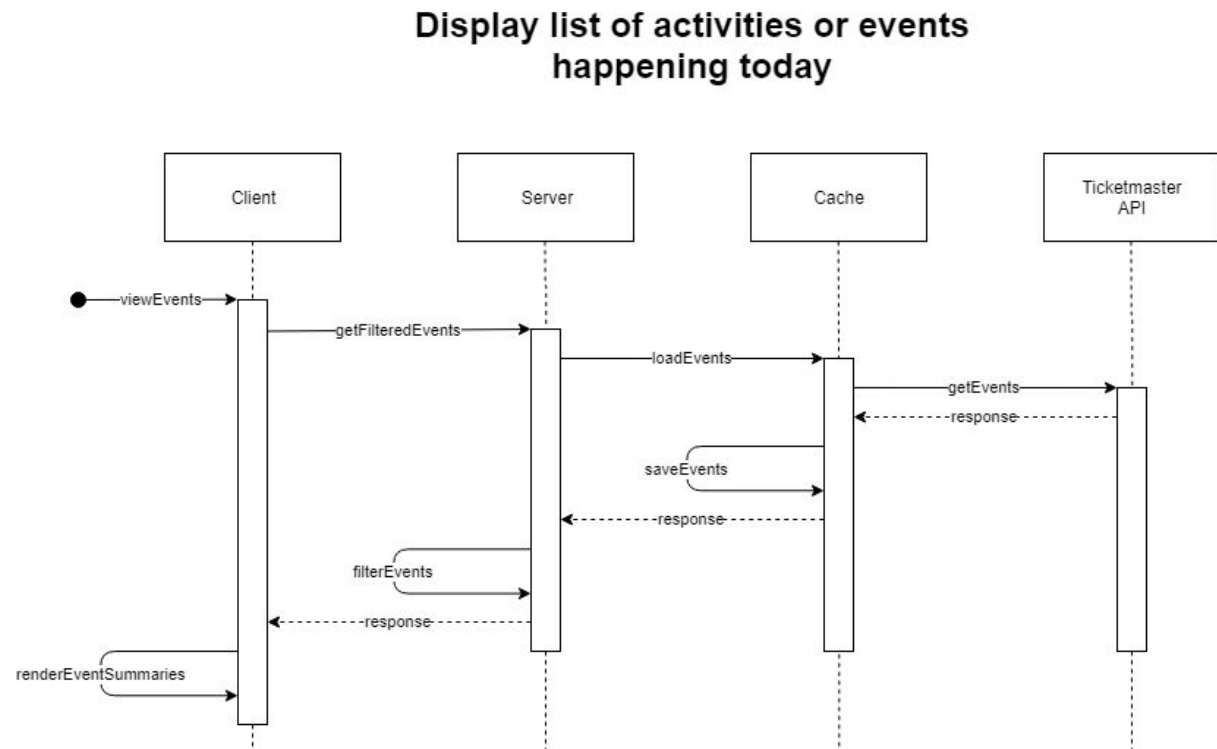
The frontend communicates with the backend through JSON objects. Specifically, there are two endpoints that the frontend uses, listed in Table 2.

**Table 2.** Interface.

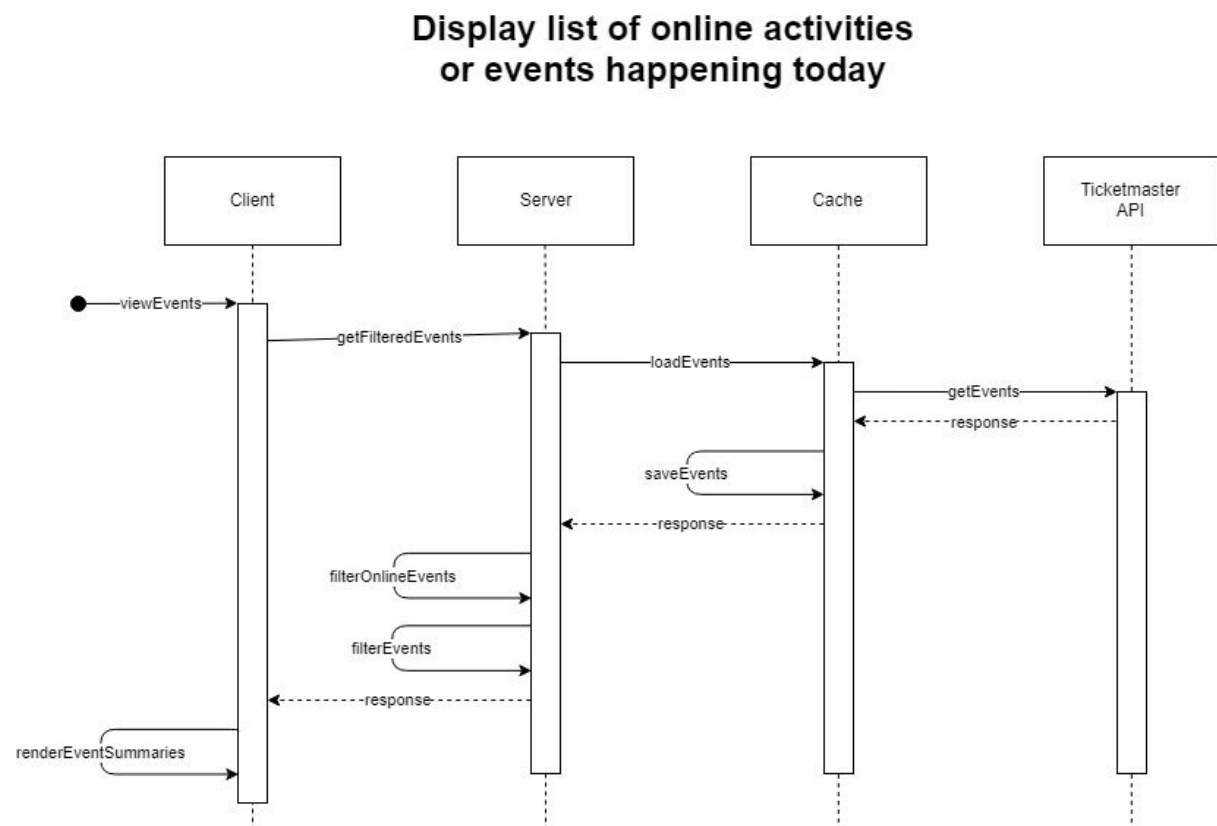
HTTP Route	HTTP Method	Parameters	Return type	Description
/conditions	GET	{ }	{ conditions: { weather: string, time: unix timestamp, temperature: float, humidity: integer, location: { latitude: float, longitude: float, activity_tags: array, sunrise: unix timestamp, sunset: unix timestamp, is_sunrise_soon: boolean, is_sunset_soon: boolean } }	Retrieves the current weather and time conditions from the OpenWeather API.
/events/recommended	POST	{ tags }	array of events, in format specified in Table 1.	Given a set of activity tags, returns a list of events suitable to the tags and current weather conditions.



## Sequence diagram



**Figure 15.** Sequence diagram for User Story 1



**Figure 16.** Sequence diagram for User Story 2

## Display summary of activities or events

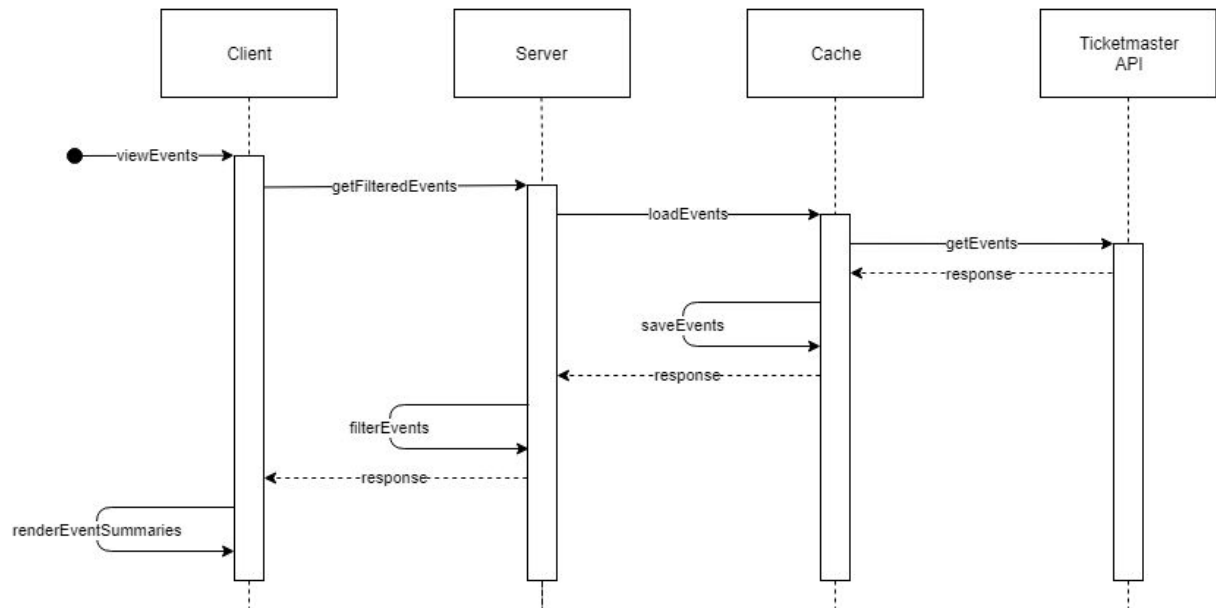


Figure 17. Sequence diagram for User Story 3

## Allow access to the original source of an activity or event

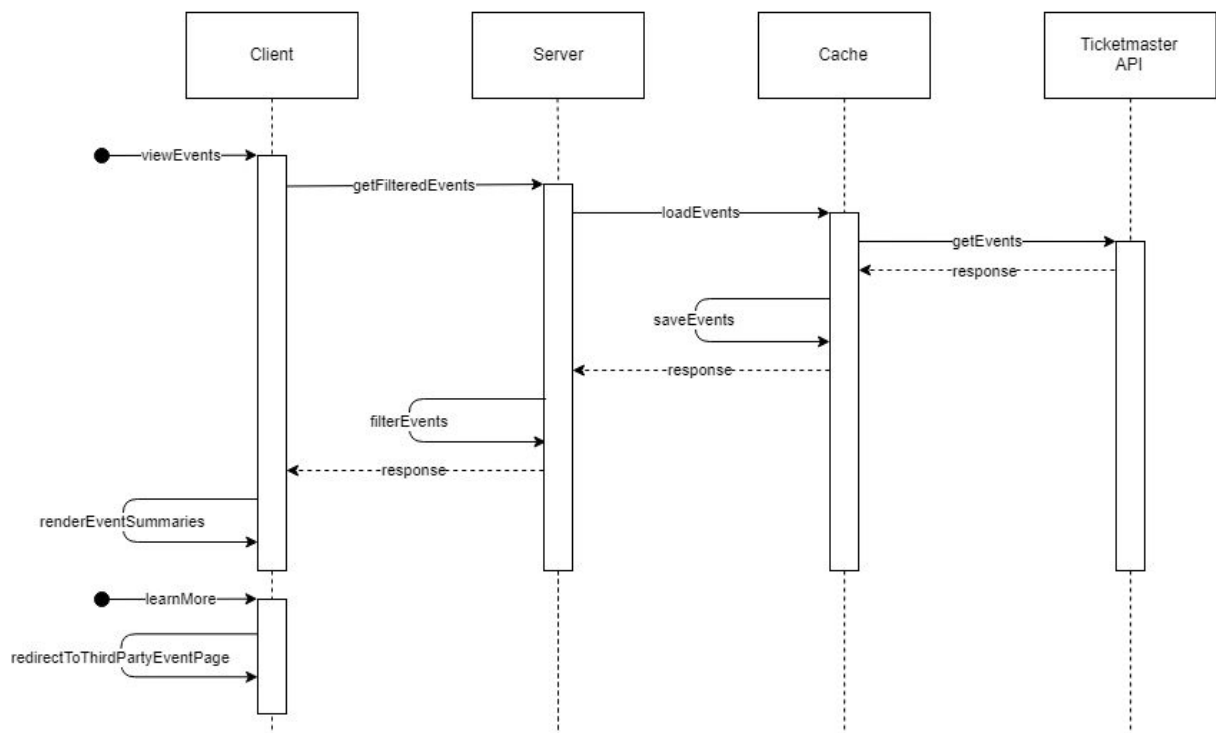
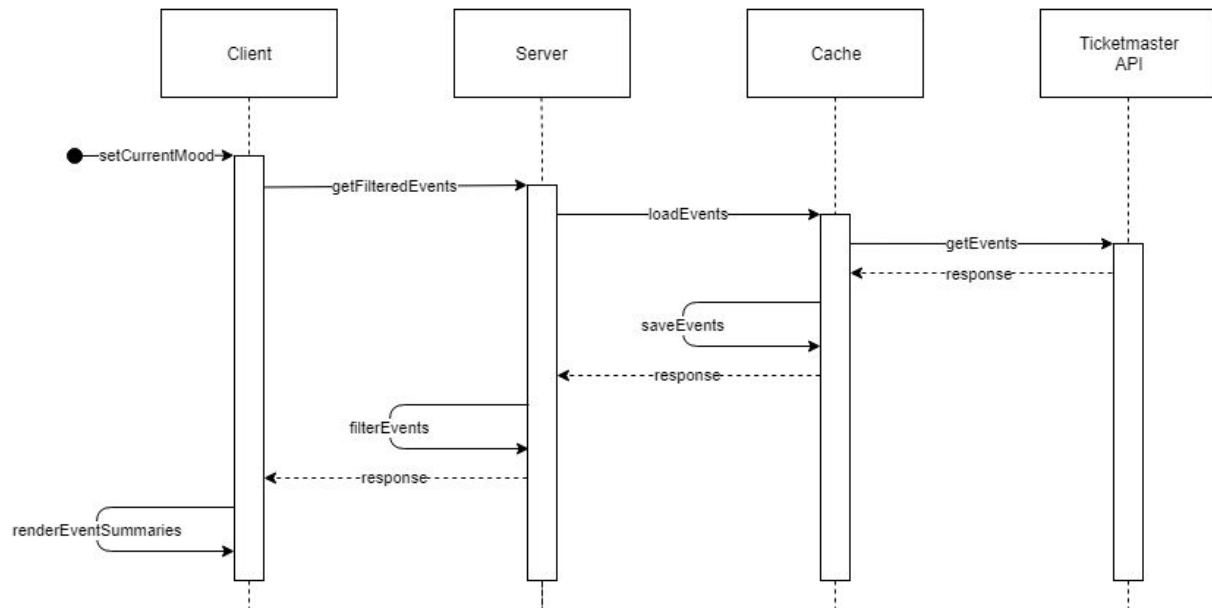


Figure 18. Sequence diagram for User Story 4

## Filter activities and events by mood



**Figure 19.** Sequence diagram for User Story 5

## Key technologies

**Table 3.** Technology stack.

	Frontend	Backend
<b>Programming Language</b>	<ul style="list-style-type: none"> <li>JavaScript</li> <li>CSS</li> </ul>	<ul style="list-style-type: none"> <li>Python 3.7.2</li> </ul>
<b>Framework</b>	<ul style="list-style-type: none"> <li>React</li> </ul>	<ul style="list-style-type: none"> <li>Flask (micro framework)</li> </ul>
<b>Built-in libraries</b>		<ul style="list-style-type: none"> <li>requests</li> <li>datetime</li> <li>math</li> <li>json</li> <li>os</li> <li>re</li> <li>pickle</li> </ul>
<b>External libraries</b>	<ul style="list-style-type: none"> <li>Fortawesome</li> <li>Axios</li> <li>MapGL</li> <li>React Router</li> <li>use-position</li> </ul>	<ul style="list-style-type: none"> <li>Beautiful Soup</li> </ul>

## Key benefits and achievements

Our choice of project architecture ensured high cohesion and low coupling through the decoupling of our frontend and backend tech stacks. Overall, the project adheres to the MVC architecture structure. Hence the backend's sole focus is on processing, obtaining and storing data in a cache. The cache helps to avoid load on our API requests, but to ensure our data is up-to-date, we refresh our cache every 24 hours. Using Python as our primary language for the backend increased our efficiency to a great extent. Python has one of the largest standard libraries as well as a plethora of additional ones that significantly and efficiently simplified the process of implementation.

Using ReactJS for our frontend tech stack ensured a dynamic UI, faster rendering and an industry level relevance. Additionally it's component based architecture allowed us to encapsulate and distribute individual pieces of our UI. This helped everyone to design and code their components without being dependent on anyone else.

Finally, since the team decided to implement a cache system for this project, HelloWorld has the potential to serve a larger number of users simultaneously, due to the reduced demand in API calls. This decision both eliminates the need to make repetitive API calls and guarantees that the application is able to redirect its technical resources towards other areas, making it faster and more effective.

As a result, our design:

- Is fast and reliable,
- Is successful in providing results for all types of users,
- Is successful in providing results to any amount of users at the same time, and
- Has a simple yet pleasing to the eye UI.

## Part 3: Team Organisation and Appraisal

### Team Member Appraisal

**Table 4.** Member roles.

Team Member	Role(s)
Michelle Seeto	Team Leader/Backend/Frontend Dev
Chinmay Manchanda	Backend/Frontend Dev
Ye (Max) Wo	Backend/Frontend Dev
Marina Reshetnikova	Design/Frontend Dev
Xifei (Cecilia) Ni	Frontend Dev

## Team Member Responsibilities

### Michelle

Throughout the project, Michelle was the team leader. Her responsibilities in this role included coming up with feature estimations and evaluating the project's quality at each stage of its completion. Furthermore, Michelle completed the code for the Eventbrite, Meetup and OpenWeather APIs, as well as brought her extensive experience in CSS and JavaScript forth through her continuous support of other team members.

Additionally, Michelle had great input in report and presentation components of the project. She was among the first to start working on her designated parts and took the responsibility of organising both visual and verbal aspects of each presentation. Finally, Michelle completed the mood button and covid GUI components, as well as made the web application responsive to different screen sizes.

### Chinmay

Chinmay's technical role in the project was mostly involved in completing the code for FourSquare and Ticketmaster APIs. Despite being unfamiliar with the technology associated with calling and retrieving data from APIs, Chinmay approached this task with great efficiency and helped the team understand how to normalise the data and what data is to be considered overall. He was also responsible for team meeting organisation and overall morale.

Furthermore, Chinmay helped implement and improve parts of the main page of HelloWorld. And had great input in the business aspects of the project by volunteering to undertake the business/financial representations of HelloWorld in all reports and presentations.

### Ye (Max)

Ye's involvement in the project included completing the entire filtering process of the application, as well as HelloWorld's caching system. He also completed the flask server, helping the frontend and the backend communicate early on in the project, hence, speeding up the team's overall progress. Ye's contribution to team work included distributing work and stepping up to do any odd or unexpected tasks that have arisen, hence helping the team overcome several obstacles throughout the project.

He was also responsible for the frontend implementation of the 'Events and Activities' page of HelloWorld and greatly participated in all report and presentation activities by continuously working on the user stories and volunteering to talk about the backend components of the project.

### Marina

As a team member, Marina participated in report writing and presenting by outlining the details of the backend implementation and talking about the possible features of HelloWorld. Her main role consisted of creating all design elements and their desirable organisation prior



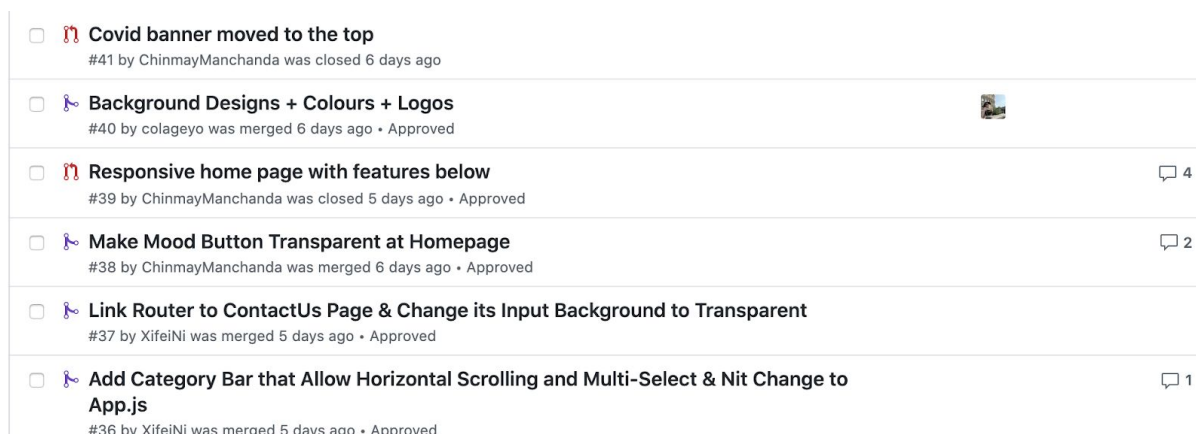
to the commencement of the project's frontend implementation, as well as continued improvement and adaptation of said design all throughout the project. Hence helping the team have a clear view of the desired outcome for HelloWorld. Additionally, Marina commenced the frontend implementation of HelloWorld's main page and finalised the navigation bar component, as well as implemented the various looks for the application's main page with Chinmay's help.











### Xifei (Cecilia)

Xifei's technical role in HelloWorld consisted of setting up the boiler code for the frontend of HelloWorld, establishing the navigation bar component and implementing the Contact Us page, as well as finalising the implementation of the mood buttons component. She has applied her great technical knowledge of the React framework all throughout the project by guiding other team members and discussing the details of Frontend's implementation in the reports. Additionally, she actively participated in all GitHub discussions by continuously reviewing all team members' code, helping improve the project's quality early on.

## Team Workflow

During this project, we followed a neat feature management workflow using the functionalities provided by Github. Each team member would break down their larger feature into a smaller working unit, create their own branch, and then submit their work as a Pull Request (PR) for all other team members to review. Following everyone's agreement on a given piece of code, the team member's branch would then be merged into a develop branch and subsequently into master. A screenshot of the team's GitHub pull requests history can be found in Figure 20.



<input type="checkbox"/>	 Covid banner moved to the top	#41 by ChinmayManchanda was closed 6 days ago	
<input type="checkbox"/>	 Background Designs + Colours + Logos	#40 by colageyo was merged 6 days ago • Approved	
<input type="checkbox"/>	 Responsive home page with features below	#39 by ChinmayManchanda was closed 5 days ago • Approved	 4
<input type="checkbox"/>	 Make Mood Button Transparent at Homepage	#38 by ChinmayManchanda was merged 6 days ago • Approved	 2
<input type="checkbox"/>	 Link Router to ContactUs Page & Change its Input Background to Transparent	#37 by XifeiNi was merged 5 days ago • Approved	
<input type="checkbox"/>	 Add Category Bar that Allow Horizontal Scrolling and Multi-Select & Nit Change to App.js	#36 by XifeiNi was merged 5 days ago • Approved	 1

**Figure 20.** Github pull requests.

## Reflection

### How did the project go

Overall, the project was a great success, since the team were able to realise the key concepts of their idea without any significant hiccups along the way.

### Active participation and code review process

All team members actively participated in any and all discussions and delivered their parts in a timely manner, allowing for further improvement of HelloWorld at each stage of implementation. The team communicated through online meetings and chats on a daily basis, as well as agreed to review each other's pull requests prior to their code being published to the development branch and then to master. This has helped keep track of all progress and obstacles in implementation, as well as, encouraged everyone to help each other and review each other's work.

### Great learning experience

Additionally, this project provided a plethora of learning experiences for all team members, both in technical and interpersonal senses. Having finished HelloWorld, every person involved can now confidently say they are able to work with APIs, know how to implement the frontend components of a web application, and finally, better understand what a successful team dynamic entails.

### Issues and Problems

#### Disagreements

A few insignificant team disagreements took place at the start of the project when trying to finalise the idea to pursue. Since the team wanted to satisfy both SENG2021 and Macquire prize requirements, it was initially hard to settle on a path that would lead to optimum outcome in all aspects. Furthermore, being unfamiliar with one another and not having an established team dynamic made it difficult to gauge what idea could and could not be implemented with the team's present skillset.

In the end, following several discussions together, an idea was established, based on the team working out an appropriate time frame and agreeing on the idea's possible capitalisation, satisfying all previously discussed requirements.

All subsequent member disagreements were settled through team agreements, usually within an online call. No suggestion was overlooked and all possibilities were reflected upon together to guarantee a fair outcome for all.

#### Technical difficulties: the learning curve

In terms of issues encountered in the technical sense, the most difficulties were experienced when implementing the frontend components. Since the team's majority weren't familiar with container spacing and CSS styling, it initially proved troublesome to meet both aesthetic and responsive requirements of each page. This hurdle was overcome with help from more experienced team members and of course through the method of learning.

### Would you do it any differently now?

Now that the project is complete, the team concluded that a few changes could be made to the way the implementation process was undertaken.

#### Using an existing JS library

Firstly, it was agreed upon that using a JS components library would've greatly aided the team in speeding up the frontend implementation process, hence possibly allowing for a greater number of features to be explored and adding value to HelloWorld.

#### Using a database

Another change that would be made is implementing a proper database in place of a cache. Since the project is expected to rely on acquiring users, it was established that just using a proper database from the start would help the team ease the process of improving HelloWorld in the near future.

#### More sophisticated filtering system of events and activities

Finally, a better filtering system could be implemented instead of the one currently in place. At the moment, the filter is quite basic, only just meeting HelloWorld's requirements. If more time has been taken to implement, test and then improve the filter, possibly more 'mood categories' could be advertised and more fine tuned events and activities examples could be displayed.