WhereTo?

Travel better in NUS.

CS3216 Software Product Engineering for Digital Markets

AY2021/22 Semester 1

Assignment 3

[Try WhereTo?]

By Group 1

- Liu Shuyang (A0206325N)
- Ong Ying Gao (A0201924N)
- Tan Wei Liang (A0135448B)
- Teo Jun Xiong (A0183852X)
- Wong Wen Wei Esmanda (A0206353M)

Milestone 0 - The Problem WhereTo? Solves

Navigating around the expansive NUS Kent Ridge campus proves to be a difficult task for students unfamiliar with the area, especially freshmen. This long-standing problem is exacerbated by the pandemic, and for some students, it has been more than a year since they set foot on campus. On top of this, freshmen may not even have been to campus before, as freshmen orientations were moved onto Zoom. These factors make it difficult for students who are unfamiliar with the layout of NUS to navigate around.

In July 2021, the Office of Campus Amenities announced a series of route amendments for the Internal Shuttle Bus (ISB) network, which promises to better meet the needs of the NUS community. The numerous amendments to the ISB routes add another source of confusion and unfamiliarity for students looking to get around the campus.

Currently, students generally look to navigation applications like Google Maps and gothere.sg. However, when it comes to walking routes, these apps will only recommend walking along driving roads, as it does not have knowledge of the various corridors and shortcuts around NUS. For instance, the route recommended by Google Maps to get from S2 to COM2 takes 25 minutes. However, by taking the shortcut through the centre of campus, it should only take 7 minutes. Furthermore, these apps do not include data on ISB routes and hence are unable to recommend travelling by shuttle buses.

To figure out shuttle bus routes, the NUS community often relies on maps pasted on bus stops or the NextBus app. Unfortunately, these solutions do not have any navigation capabilities and do not help students find their way around the campus. Someone unfamiliar with the campus is still not going to know that they should alight at the LT27 bus stop if they wanted to get to S15. Furthermore, these tend to get confusing if the journey requires switching bus services midway.

Right now, there is simply no easy or convenient way to navigate around the campus. There isn't a solution that can tell students which shuttle buses to take to get to their destination, there isn't a way to get optimal walking routes, and there isn't a way to compare between walking and taking shuttle buses. These pain points inspired us to create WhereTo?.

Milestone 1 - Description of WhereTo?

WhereTo? is a navigation application that recommends users the best route to get from one place to another in the Kent Ridge campus. We aim to provide the same core navigation features of Google Maps, including route plotting, trip directions, estimated travel time, as well as support for various modes of transport (walking-only and shuttle bus). Users will be able to compare and make a decision on whether to walk or take the shuttle bus. Note that our recommended shuttle bus route may be the same as our walking-only route, if it turns out that walking is the optimal way to get to the destination.

On top of navigation, we have other features that improve the user experience. If users are signed in, they are able to favourite certain locations, so that they are able to easily select them when they get directions. Furthermore, we store their recent trips, so users can simply choose from their recent trips if they are making the same journey. Our app is also customizable, allowing users to select their walking speed to provide more accurate travel time estimates.

WhereTo? makes sense as a mobile cloud application. Users will be using WhereTo? on the go, so creating an app tailored to mobile devices is important. Both web apps and native apps can serve this function. However, implementing WhereTo? as a web app means users do not need to download or install anything to use it, thus lowering the barriers to entry of usage.

Finding the route is a computationally intensive task. With a mobile cloud application, the route finding algorithm is offloaded to the server, which has greater processing power than the clients' devices. At present, with "only" a few hundred nodes and a few thousand edges in the graph that supports our shortest-path algorithm, client devices *could* feasible perform this calculation, but as we map out the campus in greater detail and expand the graph, the limited computational power of mobile devices will present some issues, such as long computation times.

The necessity of a database in WhereTo? is also clear. Besides using it to store user information, such as their favourites and recents, it also stores all the graph information required for the shortest-path algorithm.

As such, a mobile cloud application is the best way to implement WhereTo?.

Milestone 2 - Target Users

Target Users

The primary target users of WhereTo? are the NUS community - NUS students and staff. At first glance, it may seem that the app is only useful for new students and staff who are not yet familiar with the campus. However, even for those who are familiar with the campus, WhereTo? can help them decide whether walking or taking the shuttle bus would be faster for them at the present moment.

Apart from the NUS community, visitors will find WhereTo? extremely useful, as they are likely to be unfamiliar with the campus. They are our secondary target users. These include prospective students, tourists, and external speakers. However, it will be harder to get visitors to know about our app, as they will be unfamiliar with tools available to NUS students, just like how exchange students may be unaware of the existence of NUSMods to plan out their semester.

Promoting WhereTo? - Purpose and Goal

The most challenging part of obtaining users is getting the first batch of people to try out the app. If the app proves to be useful to them, they will continue using it, and they will likely tell their friends and peers about it. After all, we probably all heard about NUSMods after a friend or senior recommended it to us. As such, our goal will be to **gain users for our app** and achieve **200 route queries** along with **40 user sign-ups** in the **first month**.

Promoting WhereTo? - Customer Awareness

When promoting our app, it is important to consider the level of customer awareness. In his book “Breakthrough Advertising”, Eugene Schwartz lists 5 stages of customer awareness: unaware, problem aware, solution aware, product aware and most aware. Our primary target users largely fall into the **solution aware** category. They are aware that a problem exists (they don’t know how to get to their destination) and they are aware that solutions currently exist (they use Google Maps or fumble around with NextBus to figure out which shuttle bus they should be taking). However, these solutions are less than ideal.

For solution aware customers, it is important for us to highlight the features that our app offers, that are unavailable elsewhere. By pointing out deficiencies in current solutions, it will invite users to try out our application.

Promoting WhereTo? - Physical Channels

The nature of the problem our app solves presents us with an advantage, that is we are able to reach users via physical channels. When users are looking for a way to navigate around the campus, they are most likely physically present in NUS. With approval from the Office of Campus Amenities (OCA), we can paste the posters at shuttle bus stops around the school. Those who are unfamiliar with the campus will likely seek out a shuttle bus stop to figure out their next step, and this will lead them straight to our poster. These posters will be seen by students, staff, and visitors alike.

Besides bus stops, we can also paste these posters at venues of high human traffic, such as study spaces, canteens, and noticeboards. These posters will be seen by our primary target audience, which is the NUS community.

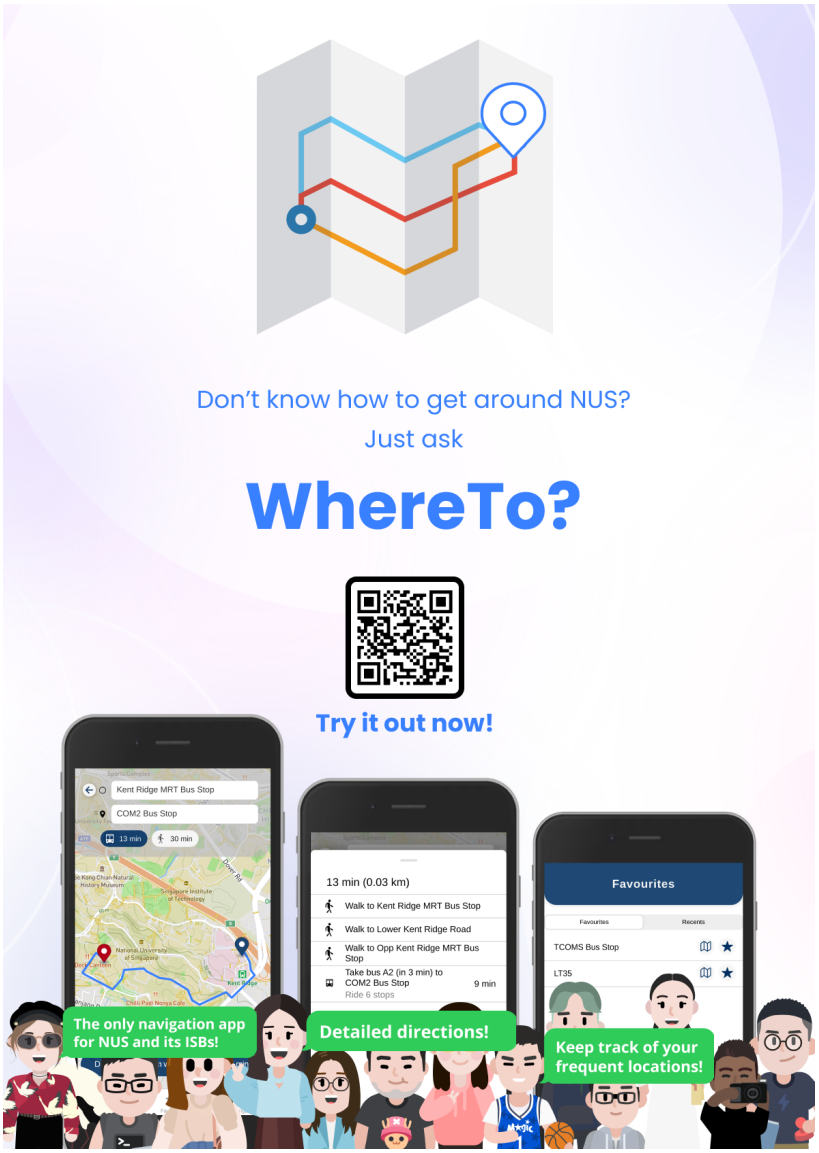

Figure 0. Promotion poster

Promoting WhereTo? - Non-physical Channels

Promoting WhereTo? via non-physical channels will be done in two stages.

Looking at NUS NextBus, we realised our app has certain functionality overlaps, such as support for shuttle bus routes. Furthermore, there is a significant overlap in user base, since those who are interested in knowing about shuttle bus arrival timings are likely to be interested in knowing about quick ways of getting around campus.

Hence, the first stage is to reach out to the team behind NUS NextBus and see if they are interested in working with us. There are many possibilities with such a partnership.

By integrating our routing capabilities into NextBus, we are able to greatly expand on NextBus's feature set and improve its usability. Furthermore, with access to the data behind NextBus (each shuttle bus is equipped with live tracking), WhereTo? will be able to provide users with accurate bus arrival timings, and we can provide more precise travelling time estimates by using historic shuttle bus data. We can even dynamically adjust travelling time estimates by factoring in the current travelling speed of buses and how crowded the buses are, which will vary during peak periods.

The second stage is to reach out to the NUS community via NUS's official channels. Getting official channels to promote our app will lend more legitimacy and convince users to try it out. We can reach out to the Office of University Communications (OUC), which sends out weekly newsletters to the NUS community. By demonstrating the effectiveness of WhereTo?, we aim to convince OUC to promote our app in this newsletter. With over 30,000 students enrolled in the current academic year, this will encourage readers to give WhereTo? a try.

Furthermore, it seems that uNivUS is trying to be an all-in-one portal for tools that students may need, such as exam results checking, hostel dining and mental health. Many of these are implemented as items under the "Explore" page, which will redirect users to the relevant website/application. If uNivUS could add WhereTo? as one of these tools, our app would likely garner way more users.

We foresee that there may be challenges in getting official NUS channels to come onboard. However, if we can convince them that our app is effective and brings about a legitimate improvement to student welfare, they may be willing to promote our app. A limitation in using official channels is that we are only able to reach the NUS community and not visitors, but this does not present a big issue as visitors are our secondary target users.

Milestone 3 - Entity-Relationship diagram for Database Schema

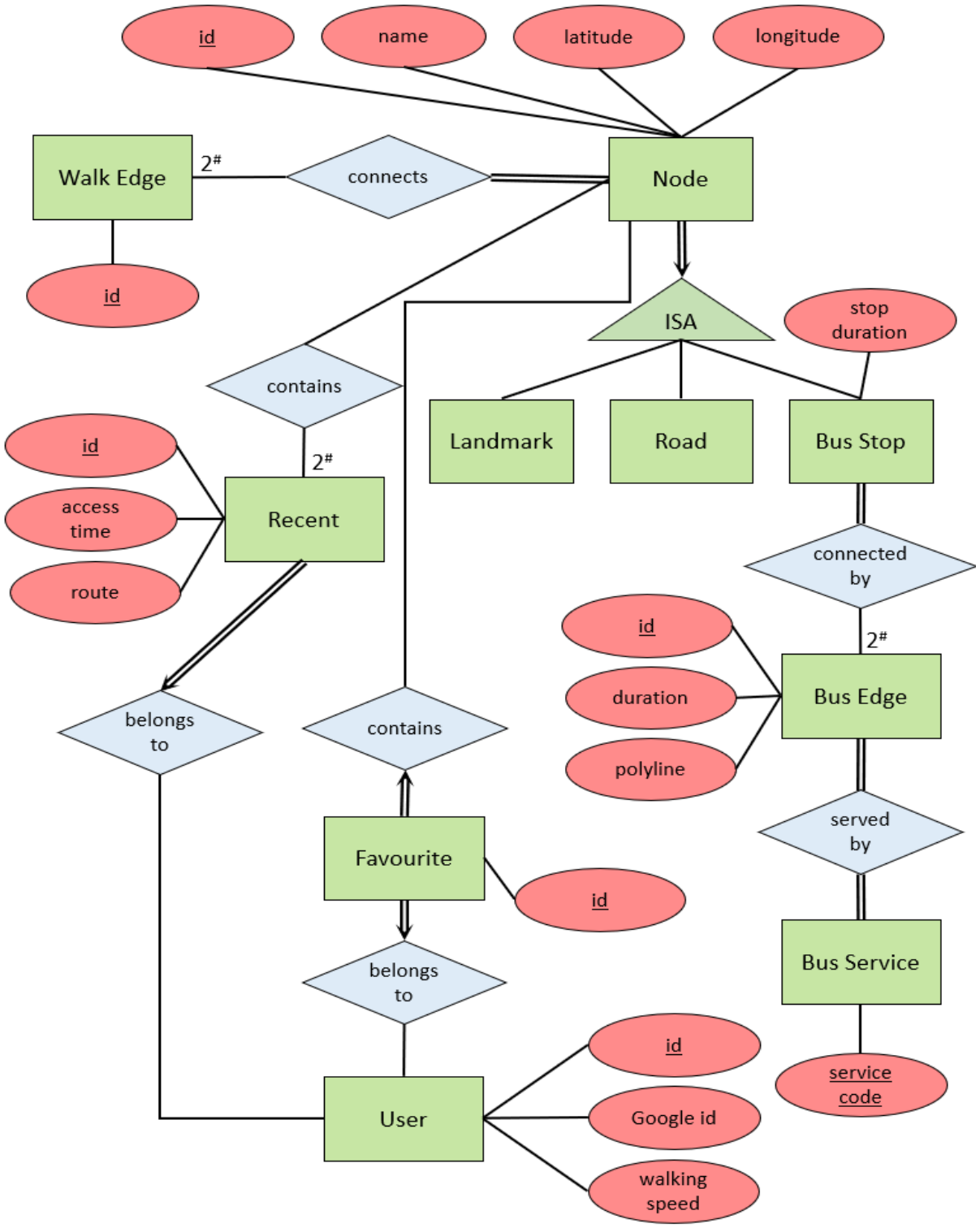

Figure 1: Entity-relationship diagram

Notes:

- # denotes an alternate notation, to better illustrate cardinality constraints.
- This ER diagram is used to document and model the 'real-world' business requirements. However, due to some implementation-specific considerations, our actual database implementation does not precisely match the ER model given above.
    - Our database contains a table for the "Bus Edge served by Bus Service" relationship and does not have a table for "Bus Service" itself. This is because the crucial information in this relationship is which Bus Services serve a given Bus Edge. As a result, a query to "Bus Services" is always accompanied by a query to "Bus Edge served by Bus Service", thus to avoid table joins during queries (thereby improving performance), we combined the "Bus Service" and "Bus Edge served by Bus Service" information into a single table.
    - In the ER diagram, it is implied that each Bus Stop is connected by at least one Bus Edge, and each Bus Edge is served by at least one Bus Service. In our database, this may not be the case. Due to frequent rerouting of shuttle bus routes (twice in the past year), we retain Bus Stops that may not be connected by Bus Edges and Bus Edges that may not be currently served by any Bus Services. This means that upon rerouting, assuming that no new Bus Stops and Bus Edges are added, we can simply make changes to the contents of the "Bus Edge served by Bus Service" relationship to support the new routes. This does not pose any issues for our routing algorithm; for example, if a Bus Edge is not served by any Bus Service, it will not be recommended to the user.

Milestone 4 - Client-server Communication Architecture

Rather than directly compare REST and our chosen alternative, we break this milestone down into two smaller considerations.

Consideration 1 (architecture): REST vs RPC

The first consideration is simply to decide how the API endpoints should be styled, ignoring implementation details. Since the job of the API is to serve client requests, it is important to design the endpoints such that it is easy for clients to use.

A key distinction between REST and RPC is that REST is *resource-oriented* whereas RPC is *action-oriented*. In REST, endpoints are designed around individual (or collections of) resources and the HTTP methods used at each endpoint intuitively describe the action to be performed on the resource (or collection of resources). In RPC, endpoints are named with actions to be performed (e.g. `add_favourite`, `find_nearest_node`), with less restriction on the HTTP methods used.

It follows that a REST-style API is optimal for handling CRUD operations, with each endpoint supporting multiple methods each corresponding to a CRUD operation. CRUD operations are also possible in the RPC architecture, but they may be clunkier as individual endpoints need to be created for each operation. However, for features that do not conform to standard CRUD operations, REST may not always be able to handle it well due to its reliance on resource-oriented naming, whereas RPC can support it as an action.

In our case, finding an optimal route does not depend on any single resource, but is rather an action that is performed on multiple resources. In some cases, such actions may be offloaded to the client, allowing a RESTful API to focus on resources instead. However, in our case, we know that one of the advantages of mobile cloud computing is having low hardware requirements by moving intensive computation to the cloud. This is a great opportunity for RPC to shine, as it can easily model these computations as actions to be called remotely by the client.

Thus, we decided to go with an RPC-style API design.

Consideration 2 (framework): gRPC vs 'our own' RPC

Next, we consider how we want to implement this API. An obvious choice given on the CS3216 website is the gRPC framework, but in this milestone, we also consider another alternative, which is to use a standard web framework (such as Django/Flask) and support RPC-style endpoints[1].

In theory, gRPC is a perfect solution. With the use of protocol buffers, messages can be sent efficiently as binary data, and the Protobuf definitions themselves provide descriptive API documentation.

However, gRPC can be difficult to implement, precisely due to the use of protocol buffers. Since messages are encoded as binary data, it becomes impossible for developers to test the API with tools like Postman and cURL. Furthermore, gRPC cannot be used directly by web browsers (our client) due to the lack of HTTP/2 support[2], so additional tools like the gRPC-web proxy is required.

Keeping in mind the tight timeline of this assignment, we decided not to go with gRPC and instead simply implement an RPC-style API with Django.

[1] The Slack API is one example of an RPC-style API that does not use gRPC.
[2] https://grpc.io/blog/state-of-grpc-web/

Milestone 5 - API Documentation

API Specification:
https://docs.google.com/document/d/1KhTa9FEZms1Onr2xfog9xRf3yJLtgFBGoniDQpNtUHk

Based on our conclusion in Milestone 4, the API is designed with the RPC architecture in mind. This means that endpoints are named after verbs (to represent actions), instead of nouns (which is the case in REST/resource-oriented design). The HTTP method to be used at each endpoint is either GET or POST, where the former does not modify the state of the application (e.g. querying the list of nodes) while the latter saves some changes to the state (e.g. add/remove favourites).

Milestone 6 - SQL Queries

We used the Django ORM for our database queries.

Note that the table names have the respective Django app name prepended (e.g. favourites_favourite refers to the Favourite model in the favourites app). In addition, we use `_id` in the model to make it clear that the column references the primary key of User, but in Django, an additional _id is appended, forming _id_id (e.g. `user_id` becomes `user_id_id`).;

1. **Get the 10 most recent routes for a given user**
    a. ORM query

```
MAX_RECENTS = 10
recents = Recent.objects.filter(user_id=user).order_by('-access_time')
recents = list(recents[:MAX_RECENTS].values('start_id', 'end_id',
'route'))
```

    b. SQL query

```
SELECT start_id_id, end_id_id, route
FROM users_recent
WHERE user_id_id = <user.id>
ORDER BY access_time DESC
LIMIT 10;
```

2. **Get all of a user's favourite nodes**
    a. ORM query

```
Favourite.objects.filter(user_id=user.id).values_list('node_id',
flat=True)
```

    b. SQL query

```
SELECT node_id
FROM favourites_favourite
WHERE user_id_id = <user.id>;
```

3. **Retrieve all nodes**
    a. ORM query

```
Node.objects.values_list()
```

    b. SQL query

```
SELECT * from nodes_node;
```

Milestone 7 - Splash Screen and App Icon

The icon is designed with vibrant colours to be eye-catching. The location marker placed over three routes represents a map to make it as apparent as possible that WhereTo? is a navigation assistant. A slogan is also added beneath to better get the message across that the app aims to ease the headache associated with navigating the campus.

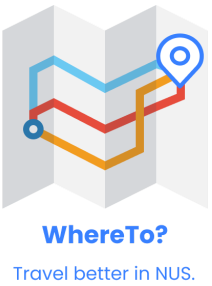

Figure 2: App icon

The splash screen is designed using an animation (GIF) that shows the location marker hopping from left to right towards the map to bring across the idea that WhereTo? is bringing users towards their destinations. The animation also serves to keep the user visually engaged as they wait for the application to load, enhancing the user experience.

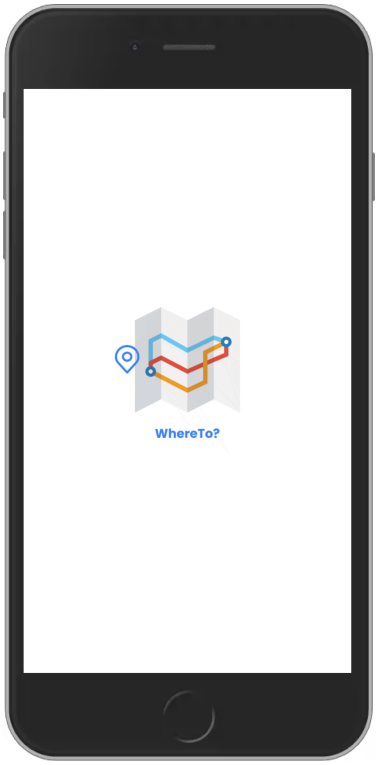

Figure 3: Splash screen

Milestone 8 - CSS Methodology and UI Design

CSS Methodology

Our team agrees that structuring Cascading Style Sheets (CSS) and *styling* CSS is difficult. Often, CSS becomes messy and readability deteriorates as development continues. Our team considered three CSS methodologies, Object Oriented CSS (OOCSS), Scalable and Modular Architecture for CSS (SMACSS), and Block Element Modified (BEM).

To streamline our decision process, our team came up with two metrics to evaluate the suitability of each CSS methodology for our development:

1. Is the CSS methodology suitable for our scale of development?
2. Is the CSS methodology easy and quick to adopt during the limited time we had for assignment 3?

From our understanding, OOCSS aims to identify objects in CSS, and abstract them into an independent snippet of HTML, CSS, and JavaScript, which supports reusability. This meant that a lot of development time had to be poured into planning and sorting out which elements should look or behave the same way. Given the nature of WhereTo?, there is not enough repeating visual patterns to justify the effort required to separate structure (HTML, JavaScript) and style (CSS). Moreover, since our team decided on using Ionic (more on that below) and React, which are designed to build standalone and reusable components, it did not make sense to adopt OOCSS and do "double work", which had essentially the same concept. Therefore, OOCSS was not suitable for our scale of development and the frameworks we adopted.

Next, we compared how easy and quick it would be to adopt SMACSS and BEM. We acknowledge that SMACSS is more documented than BEM as it has a detailed online ebook elaborating on it, we found that it would take too much time to adjust to the naming conventions. On the other hand, BEM follows the simple and intuitive `block--element__modifier` as the name suggests. While this may cause CSS classes to become long and less readable, we accepted this tradeoff for ease of application.

Another factor that caused us to adopt BEM over SMACSS is similar to why we decided OOCSS was not suitable for WhereTo?. SMACSS has a similar concept of modularity that overlaps with React's core principles. BEM provides easy-to-use and descriptive class names and a flat CSS hierarchy, which made it suitable for us to use with React, which is optimised for modularising UI components.

After the decision process, we decided to employ BEM as our CSS methodology.

UI Design

Due to the nature of WhereTo?, we decided to keep it simple. Users' main goal is to search for routes between two locations, and the steps allowing them to do so should be minimal. Since WhereTo? is related to NUS, we chose the primary colour for WhereTo? to be the NUS official blue colour. This also gives a sense of familiarity to NUS students using the application and helps them to make the association that WhereTo? will help them navigate around NUS. Apart from this association, dark blue also serves a practical purpose. We chose to use it as the colour for our call-to-action buttons since it has a high contrast from the application's white background, making it easy for users to spot. We also included drop shadows on the buttons for added emphasis.

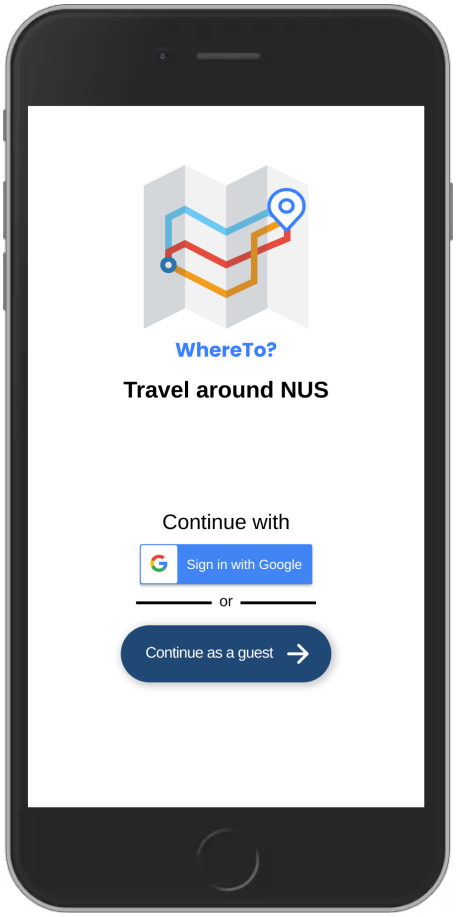

Figure 4: Login screen

We chose to display the login screen first to encourage new users to sign in to WhereTo?. Doing so allows them to make use of the full suite of functionalities (favourites, recents, walking speed) offered by WhereTo?, improving their user experience. Leveraging the visual hierarchy principle, we intentionally positioned the Google sign-in button above the guest sign-in button to form a visual hierarchy among the two buttons, placing emphasis on signing in (figure 4).

We decided to position our tab bar at the bottom of every page (except for the login page) instead of the top so as to reduce noise and remove distractions from the key functionality of WhereTo?. Upon logging in with their Google account or as a guest, the user is first greeted with the navigation screen. As mentioned above, since the main functionality of WhereTo? lies in helping users to navigate, we intended to bring them to this page with the least amount of clicks.

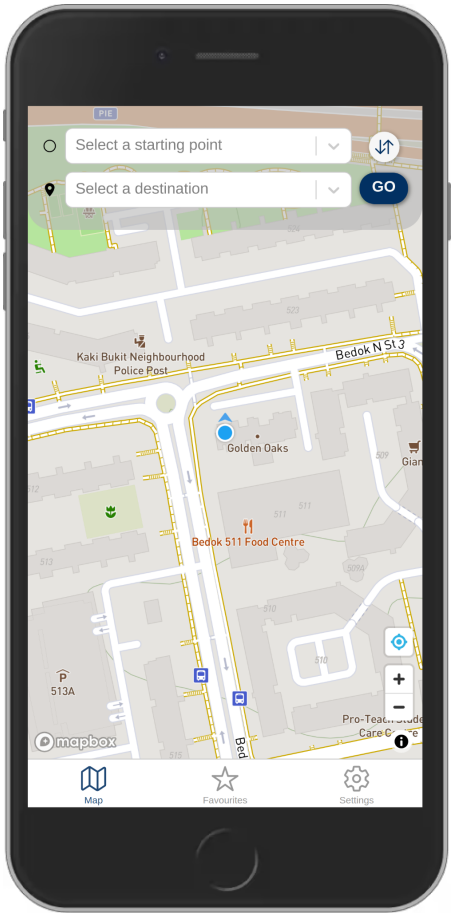

Figure 5: Search screen

On the search screen, the main actions of selecting starting and ending points, and clicking the “GO” button is clearly displayed at the top of the screen (figure 5). Being at the top of the visual hierarchy, the users are able to immediately grasp the three main actions on the screen, where are to:

1. Select a starting point
2. Select an ending point
3. Search for the fastest route between the two

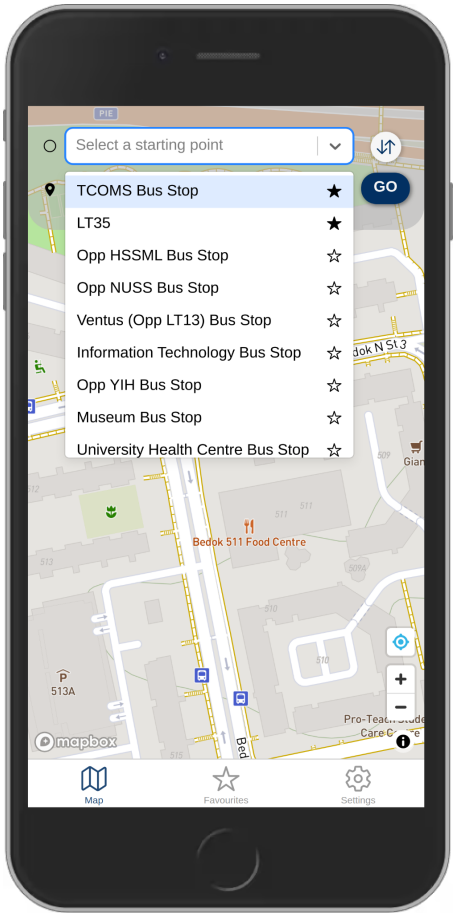

Figure 6: Search screen (dropdown expanded)

The additional functionality of favouriting a location from the dropdown is hidden from sight initially to prevent distraction (figure 6).

The favourites screen and the settings screen are designed similarly to provide a visually consistent experience for the users. This allows them to get acquainted with WhereTo? quickly.

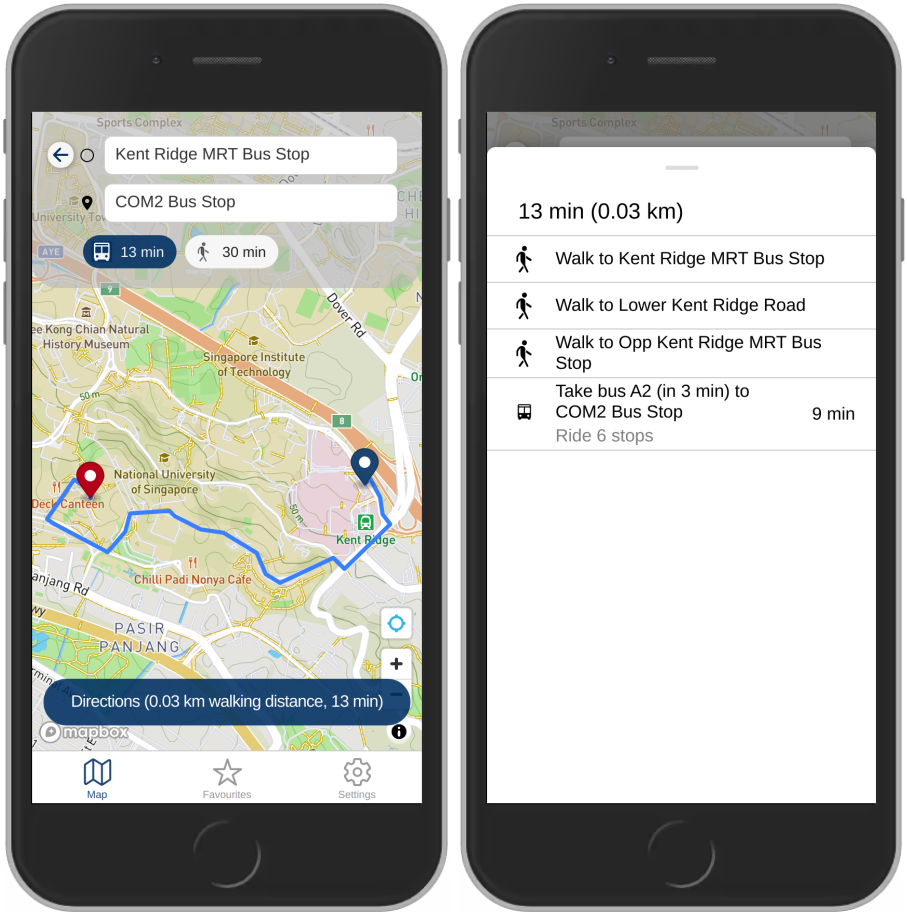

Figure 7, 8: Search results screen and detailed directions screen

Upon a successful search, users will be shown a plotted route with a start (navy) and end (red) markers (figure 7). On this page, the two main actions users can do are:

1. Switch between bus and walking directions
2. View detailed directions (figure 8)

When the chosen direction type is changed, WhereTo? automatically re-plots a new route and updates the detailed directions.

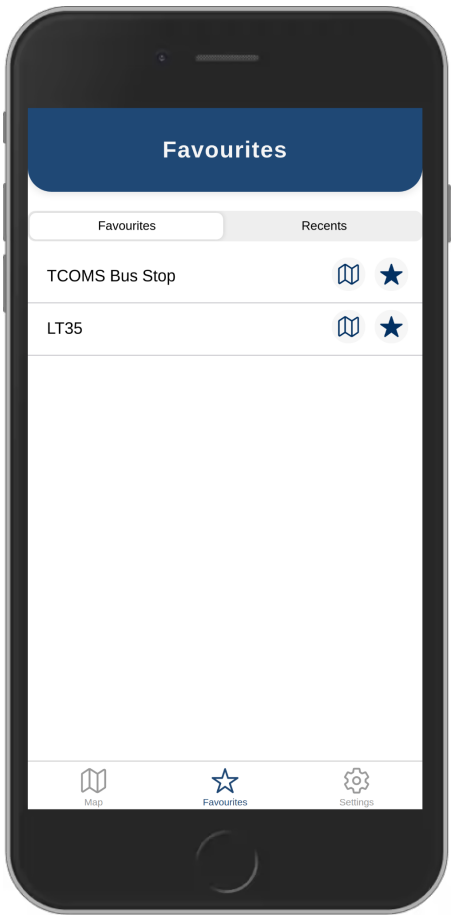

Figure 9: Favourites screen

The favourites screen has a simple layout that is easy to comprehend. The three main actions for users on this screen, switching between favourites and recents segments, favouriting, and getting directions, are made immediately visible (figure 9). We added circles behind the map and star icons to give a more intuitive representation of a button. The lists are also separated with lines to help the users visually distinguish between each list item.

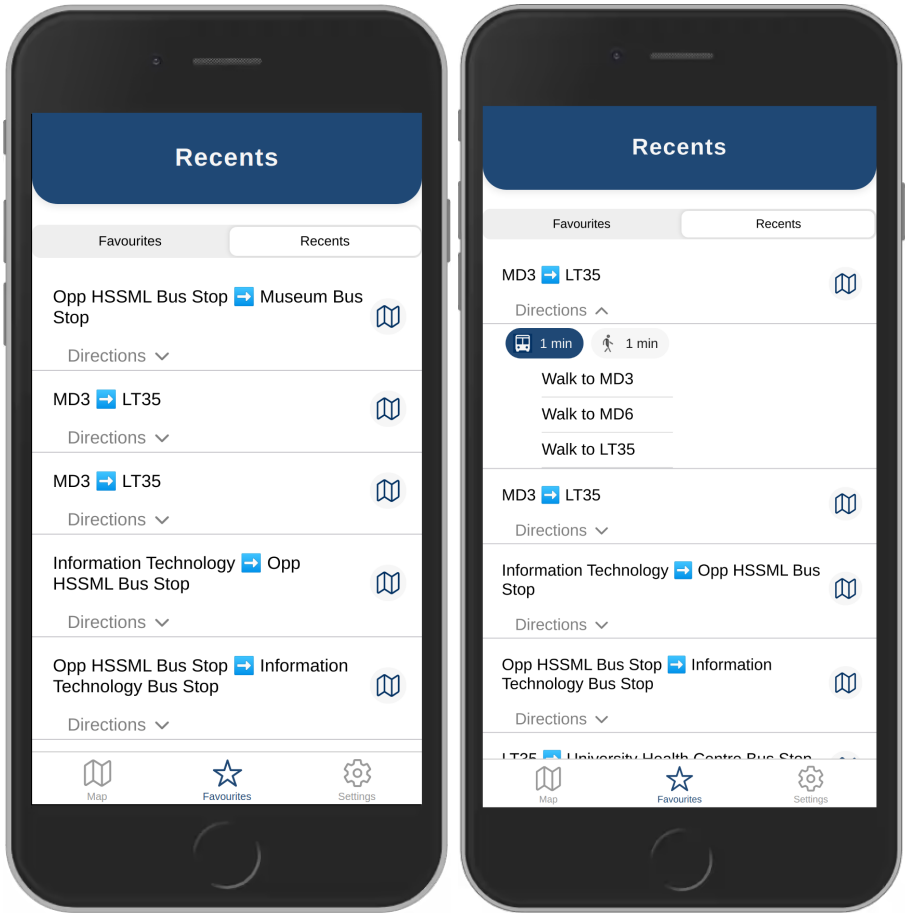

Figure 10, 11: Recents screen (unexpanded and expanded directions)

To support ease of reviewing past routes, the Recents page is populated with the directions and time required between two locations (figure *10*). Following the idea of displaying a visual hierarchy, all the directions for each recent route are hidden unless the user expands them (figure 11). Doing so ensures that the act of expanding the detailed directions is intentional, and users are not flooded and overwhelmed by the directions of recent routes that they may not be interested in.

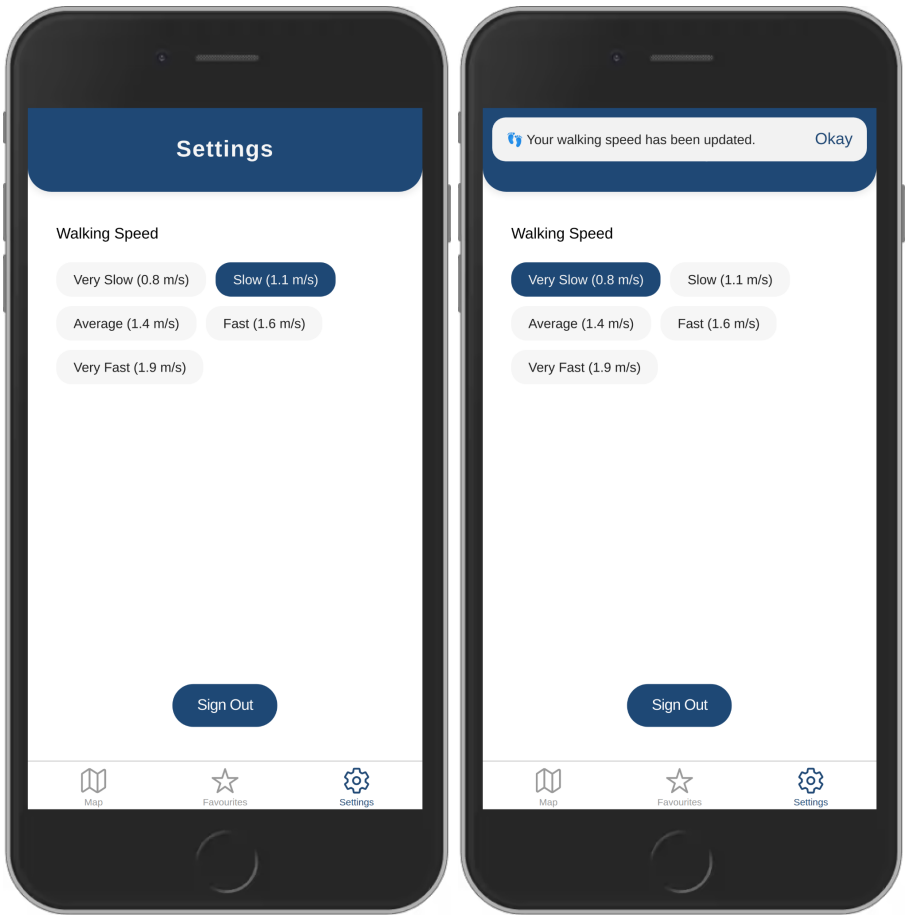

Figure 12, 13: Settings screen (with and without a toast)

The settings screen makes use of the high contrast between the dark blue and off-white background of the options, making it obvious which option is being selected (figure 12). By adding a toast (figure 13) upon changing their options, it gives the user visual feedback and assurance that their action has resulted in an outcome. The sign out button is also positioned on this page as it conforms to the normal behaviour of apps and matches the user's expectations.

Milestone 9 - HTTPS Best Practices

Our team recognises the undeniable value of securing our website with HTTPS to ensure the confidentiality of the transmitted data. We chose to use AWS Amplify to deploy our front end because it provided simple and automated HTTPS.

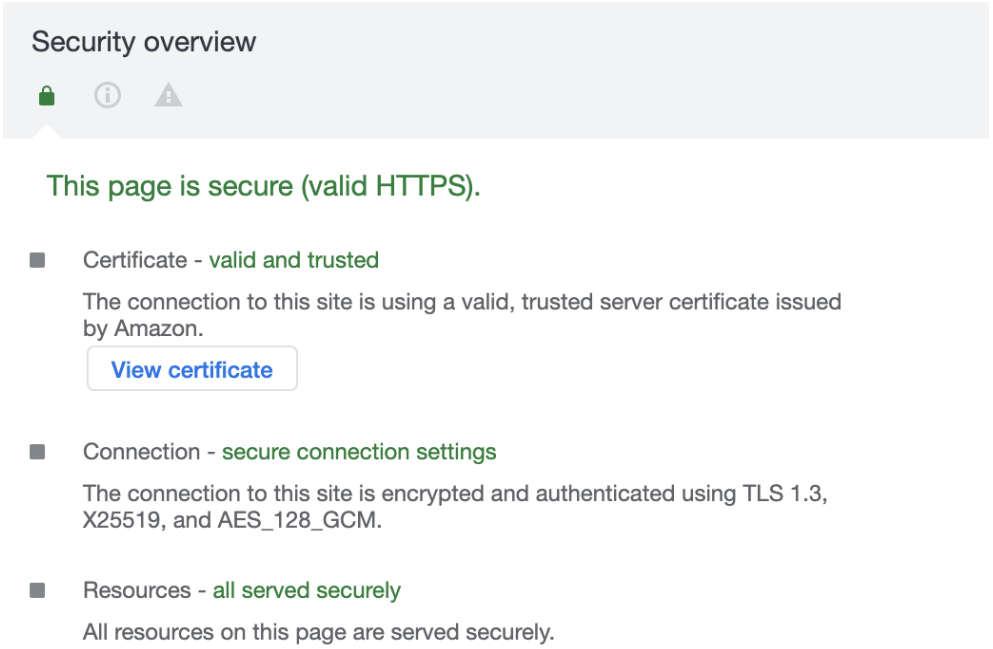

Figure 14: Security Overview of the site

AWS Amplify makes use of industry best practices for adopting HTTPS.
Firstly, it uses a security certificate that is issued by a reliable certificate authority that is trusted by all modern browsers. Furthermore, it uses a RSA key length of 2048 bits as recommended by NIST providing a greater level of security for our site without compromising performance.

Next, AWS Amplify makes use of the latest version of the SSL/TLS protocol (TLS 1.3). Apart from having other benefits such as increased speed, it also enhances security as it rectifies insecure features that were identified in the earlier versions. This provides a higher level of security to the site and makes it less vulnerable to attacks.

Finally, AWS Amplify also automatically handles HTTP redirection. Requests to the site using an HTTP connection are automatically redirected to HTTPS, ensuring that the user uses the site over a secure connection. This prevents users from accidentally accessing an insecure version of the site, leaving them vulnerable to attacks and compromising confidentiality.

Milestone 10 - Offline Functionalities

One key drawback about Mapbox is the lack of offline functionality for its JavaScript package (mapbox-gl). The ability to download offline maps is not implemented for mapbox-gl, which makes it difficult to allow users to search while unconnected to the internet. Hence, the main offline functionality of WhereTo? lies in being able to view the directions for their recently searched routes. This aligns with the user's expectations as it fulfils the main functionality of WhereTo? to help them navigate. While the benefit of this functionality may not be immediately obvious, we argue that there is merit for users who primarily use NUS WiFi over mobile data, and this is especially so now with NUS requiring us to be signed into NUS WiFi constantly while on campus. NUS WiFi coverage may not be always available, especially if the user is moving around. However, WhereTo? still allows them to view directions for their recently searched routes.

Moreover, students tend to go to the same few venues for lessons, it is reasonable that they would usually go back to what they have recently searched for. In addition, this also helps to circumvent the scenario where the user gets disconnected in the middle of navigating their way to a destination. By providing the functionality to access their last searched route, they would still be able to continue on their way without a connection.

In order to keep the client synchronised with the server, every API call triggers an update to the local storage upon completion. This is achieved with stores and gateways simulating the MVC structure. Gateways make direct calls to the server and stores hold the retrieved data in memory and also save them to local storage when change is detected. The UI in turn observes the store and is updated with changes to the underlying state.[3] This keeps the locally stored variables updated with the latest actions performed by the user and consistent with the server. That way, even without internet connection this data can still be displayed to the user.

In WhereTo? offline users are unable to perform activities which affect the server. This includes being unable to search for routes, add or remove locations from favourites, or change their walking speed. This is done by disabling such buttons if `navigator.onLine` is false. This ensures the client's state is always the same as that of the server's, eliminating the need for synchronisation when internet connection is restored.

[3] Design inspired by this example on decoupling UI from the database in React

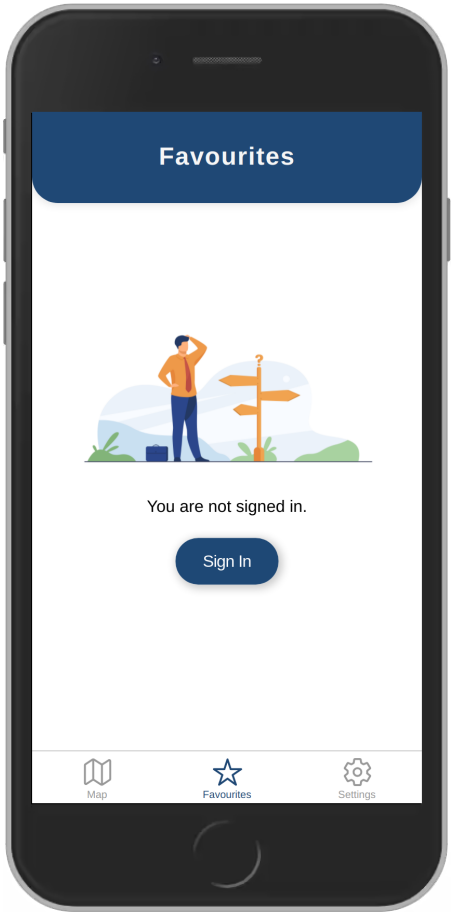

Figure 15. Offline notice

A scenario we have considered is how to keep the user signed in when they lose connection. Initially, every action that involved retrieving information from the server or updating the server would trigger an API call to verify the stored JWT token. However, this resulted in the user being logged out when they performed such actions while being offline. As such, they would be classified as unauthorised and not logged in and will not be allowed to access their favourites and recents (figure 15). In order to overcome this problem and achieve the above functionality, we changed the implementation to only check whether a stored JWT token is valid if there is an internet connection. If there is none, the user will be able to use WhereTo? as we hold off the API call to verify their stored JWT token. This allows offline users to still view their recents.

Milestone 11 - Authentication Scheme

Session-based authentication

In session-based authentication, the server will create a session with its ID for the user upon login. While the user stays logged in, this session ID, stored locally on the browser as a cookie, is sent to the server on every subsequent request. The server can then compare the submitted session ID against information stored in the memory to verify the user's identity before sending a response.[4]

Token-based authentication

In token-based authentication, user information is encoded as claims in a JSON Web Token (JWT). Once the user is logged in, the server creates a JWT and sends it to the client to be stored (usually in local storage). The client includes the JWT (containing the encoded user information) in each request to the server, so the server only needs to validate the JWT itself to extract its claims and does not need to consult any other memory (e.g. database).

Comparison between the two

The main advantage of using token-based authentication over session-based authentication is scalability. With token-based authentication, all information is held in the token. As such the server does not need to store user state in memory, effectively reducing the overhead of maintaining a session. This allows the server to potentially serve a greater number of users concurrently, which is highly pertinent to our application. We foresee our application serving a large target audience, with the student population being more than 30,000 strong in the last academic year[5]. Furthermore, since lesson timings are usually within the same time bracket, it is more likely that students will be using our application at similar timings.

One disadvantage of token-based authentication is that the size of the token is typically larger than a session ID used by session-based authentication. If there is a lot of user data encoded in the token's claims, this may slow down the application, affecting the user experience. However, this is not the case for WhereTo? as we do not need to store a lot of user information, hence our team decided that this was an acceptable tradeoff and that being able to serve all users using the application took precedence over shortening the application loading time.

---

[4] Taken from (Hsu, 2018)

[5] Enrollment statistics for AY 20/21

Milestone 12 - Native Design

Choice of framework

Before deciding on a framework to recreate the native experience and design on our progressive web app, we listed a few criteria to choose a framework from Ionic, Ratchet, Framework7, Onsen UI, Vuetify, and Material-UI:

1. Does the framework support React?
2. How extensive are the native components and UX provided? Which framework had the components we required?
3. How well-documented is the framework?
4. Is it the aesthetic we wanted?

The first criteria is obvious, our team had the most familiarity with React, therefore we were interested in a framework that had support for React, and there was no need to pick up another framework. This led us to eliminate Vuetify.

Next, we stacked the remaining frameworks against each other and found out that the sentiment about Material-UI's learning curve and documentation was really negative. This meant that we had to put in more effort to pick up Material-UI as compared to the alternatives.

The aesthetics provided by the frameworks out of the box were an important factor as well because it reduced our team's workload in styling the app, and our team did not feel like Ratchet, or Onsen UI had the clean and minimalistic aesthetic we wanted.

In the end, we decided that Ionic was the right tool for our use case because it had more documentation and resources available online. With almost three times the number of stars and issues on GitHub, we felt that Ionic's larger community (and hence a larger pool of resources) would provide more assistance when we run into issues or bugs.

Mobile site design principles

1. Keep calls to action front and center

   For all pages in WhereTo?, the main call to action was made visible and within the viewport (see UI Design). This makes it easy for users to easily identify the main actionable tasks for every page.

2. Make it easy to get back to the home page

   The presence of the tab bar makes the home page easily accessible by the user from any page. The user is able to navigate to the home page with a single click.

3. Let users explore before they commit

   Understanding that the need for upfront registrations can be a barrier to entry for new users, we included the option to use the WhereTo? as a guest user. This allows the user to explore the main functionality of the application before deciding to register to use the additional features (favourites, recents, walking speed).

4. Optimize your entire site for mobile

   Using the Ionic framework, it provides mobile-optimised UI components, which our team used to wrap around our own React components. This resulted in a mobile-responsive app that fits different screen sizes of different phones. In addition, with the Ionic UI components, it changes the style of the elements according to the operating system, giving the native mobile application feel.

5. Ensure site search results are relevant

   We foresee that students will usually go to and fro a few locations repeatedly in their academic life and hence should have the ability to mark destinations as favourites. As such, as soon as the user taps on the search bar, the top results displayed will always be the user's favourite locations (figure 6), eliminating the need to even start typing. For instance, even if a user often travels to COM2, there might be times where he will need to travel elsewhere on campus from that location. Having COM2 appear on the top of the original search will save him the time of having to manually search any further.

Milestone 13 - User Experience

In the decision of which workflows were important, we felt that it was important to tie them back to the main function of WhereTo? -- to help users easily find directions around campus. As such, our points of consideration were mainly centred around helping users to execute the searching of directions more easily. With that, we decided upon the three common workflows of WhereTo?, searching for directions, being able to favourite a location as well as viewing the recently searched routes.

The first workflow is for users to be able to easily search for locations and view detailed directions on how to get there. WhereTo? performs all the calculations and immediately presents the optimal route, allowing the user to only focus on navigating according to the route given. The geolocation feature also helps to provide visual cues of where the user is at, providing them assurance as they are able to ensure that they are not straying from the path. We decided that since searching directions was the core functionality of WhereTo?, it should be accessible to all users whether they have signed in or not.

Students tend to have common places that they like or have to go to frequently. Allowing them to favourite such locations can help short circuit the searching process by automatically filling in the end location. Users can then easily retrieve directions to these favourite locations, improving the user experience of finding directions. To improve the user experience, we decided to implement favouriting of locations in the search bar. This reduces the user's effort to do so, as the alternative would have been to switch to the favourites screen, search and select a location, and add it to the favourites. This action is concentrated on just one screen instead of three, and it flows intuitively from searching for locations.

The last workflow of viewing the list of recent routes came out of the recognition that students commonly have to visit the same few venues on a cyclical basis for their weekly lessons. They would often have to travel from the first lesson to the next, with the start and end locations being the same. By storing a list of their recent routes, it relieves them of the need to manually search the start and end location. This reduces the number of clicks they have to perform to get the directions, easing their searching process.

We chose these three workflows over other alternative workflows, such as the workflow to change their walking speed, as they were important in helping users retrieve directions more efficiently. On the other hand, the function to change the user's walking speed serves only as an embellishment to the existing features, allowing the application

to provide more targeted and accurate estimations of the travel duration. This is not essential in helping the users get directions, thus not being one of the common workflows.

Milestone 14 - Google Analytics

Link to PDF here

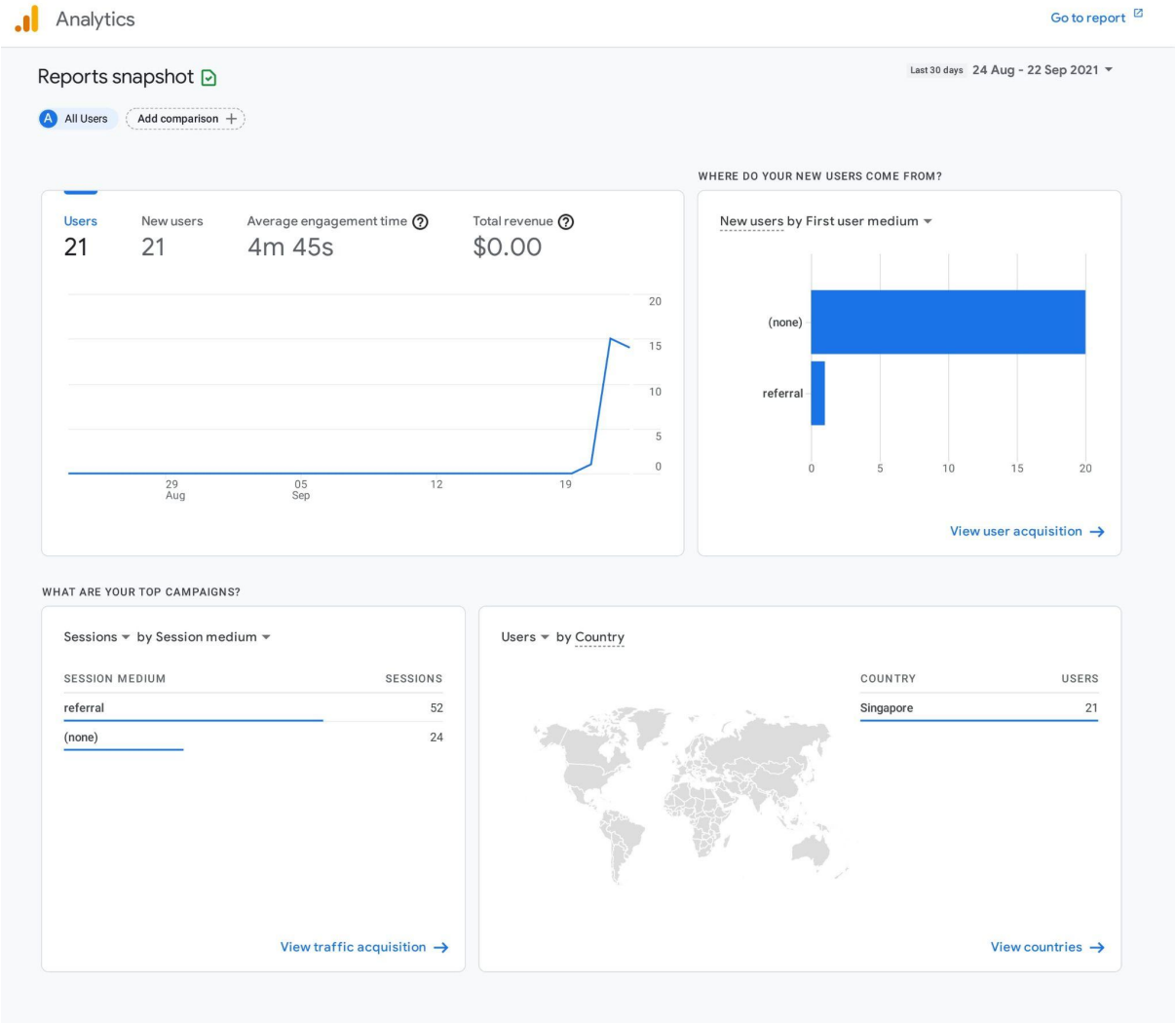

Figure 16. Google Analytics report page 1

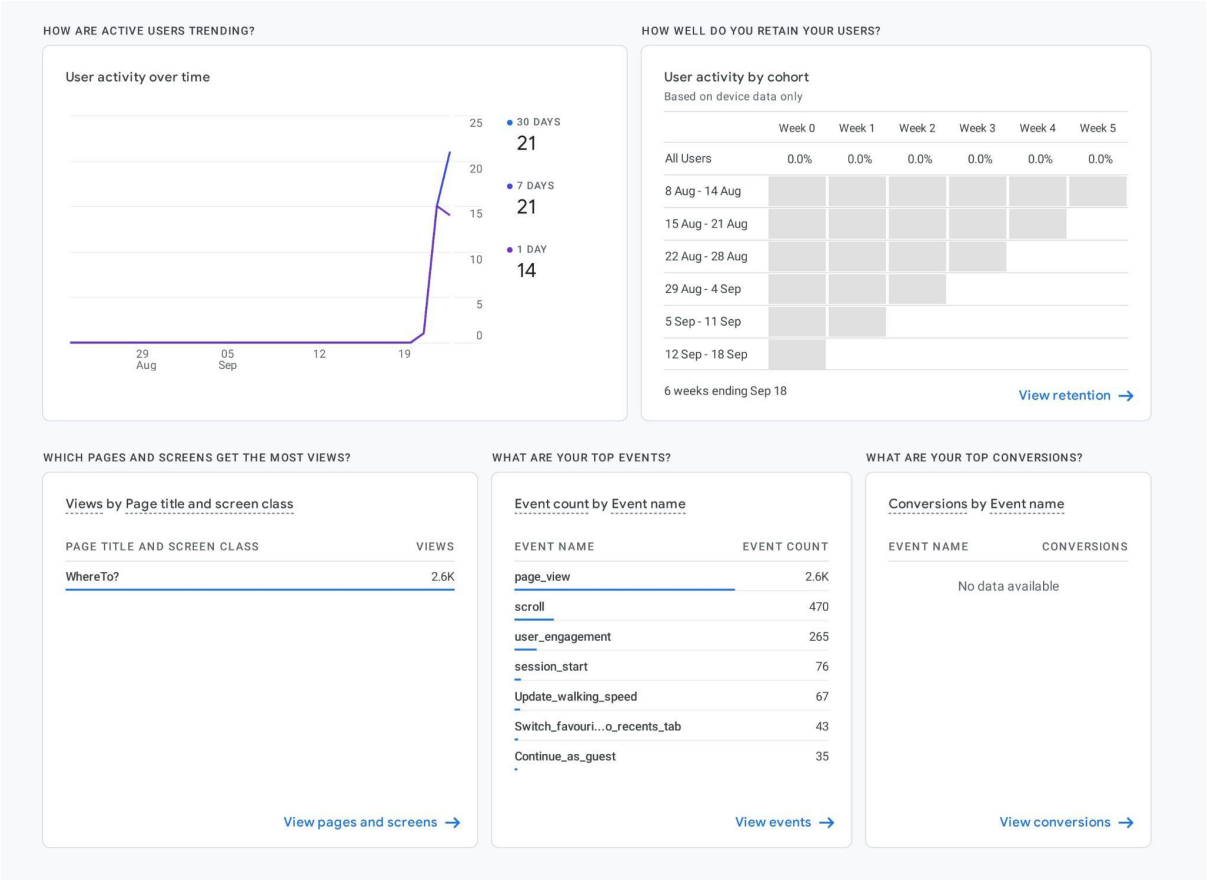

Figure 17. Google Analytics report page 2

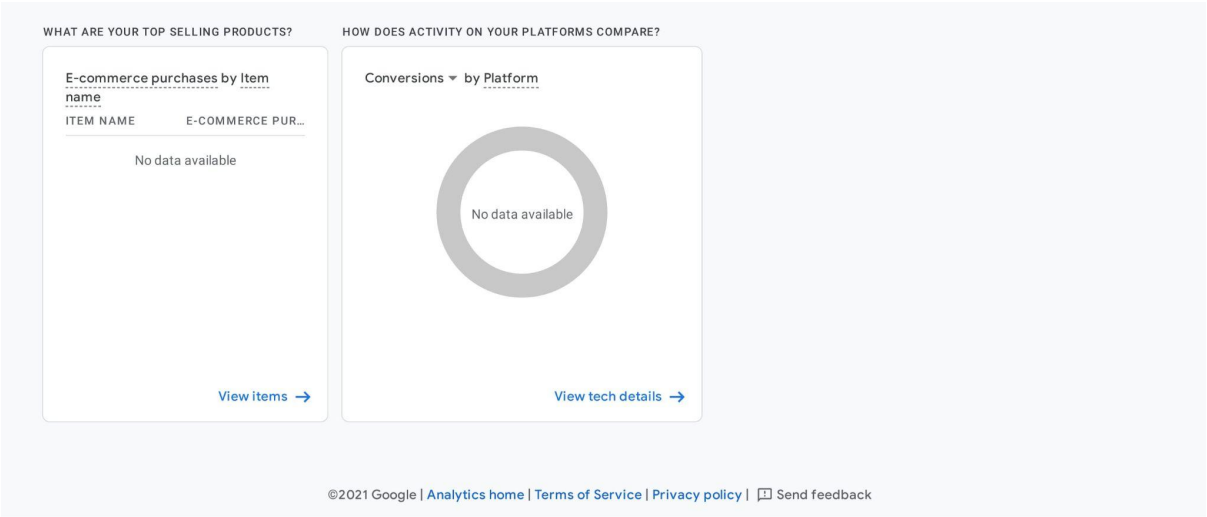

Figure 18. Google Analytics report page 3

Milestone 16 - Social Integration

Registering for a new account stinks, it's repetitive, it's tedious, it's lame. Coming up with a unique password for every new account is an admirable endeavour, but for the most part, it is just another thing to forget. Reusing passwords saves you that hassle, but it doesn't take an Einstein to realise that it becomes a massive security hole. And when your service requires email confirmation or 2FA? That's just icing on the cake.

Our application is just a navigation application at its core, the only information it needs from users is simply the origin of their trip and their destination. As such, this means users can use this application without ever needing an account. That said, we do allow users to track their favourite and recent locations if they wish to, so we require some form of user account. However, an entire sign up process would be overkill for what would effectively be optional functionality, and this is where social plugins come in.

As for which login plugins to use, we opted for Google because it is the most relevant to our target audience. Facebook for instance is still popular, but its use has been declining among the younger generation, with the majority of its users being 25 and older[6]. On the other hand, all students will probably have a Google account especially when so many modules require them to use a shared Google workspace. As such, Google was a natural choice over other social media plugins such as Facebook when it comes to easing the login process.

As previously mentioned, our application is a navigation utility and we do not foresee users needing to broadcast any information to their feeds on Instagram or Twitter. As such, we had opted to not integrate any plugins for those purposes.

[6] Facebook user statistics, 2021

Milestone 17 - Geolocation

Given that WhereTo? is a navigation application, the geolocation feature is integral to our application functionality. By integrating the geolocation API, users are able to see their current location on the map (figure 16). They can thus make use of the location marker and simply follow along the route that is depicted on the map. Such a visual representation is easier than following directions in words, aligning with the goal of helping users to navigate easily and enhancing the user experience.

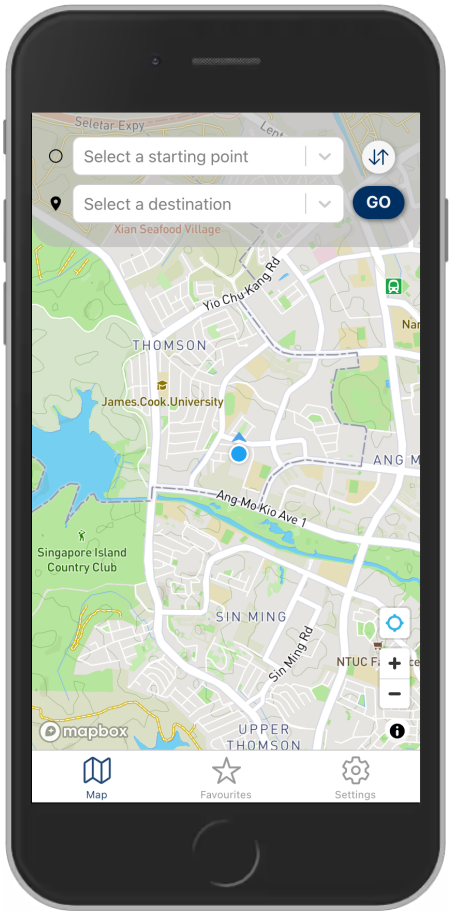

Figure 19: Location marker using Geolocation API