

SafeTrip

Neil Alombro

Zipporah Price

Angelica Silva

William Thompson

Todd Vermeir

Christopher Wareing

SENG202

Software Engineering Project

2023

University of Canterbury

Executive Summary

SafeTrip is an application that has been thoroughly planned, designed, and developed by a team of developer SENG202 students at the University of Canterbury. The target demographic of SafeTrip is everyday road users in New Zealand who have a particular concern for their safety on the road and want to feel secure in their route-planning choices. The app provides users with visual displays of crash data through a map and allows them to plan different routes based on calculated safety ratings. Possible application competitors such as Google Maps and Crash NZ were considered while exploring the business context, however SafeTrip combines aspects included in both apps to provide users with a unique functionality that is not offered elsewhere. Thorough research about possible stakeholders, risks and requirements was conducted to develop a clear vision for how the intended application will be implemented. Initial system designs were presented through a deployment model diagram, a use case diagram, and a UML class diagram. Possible graphical user interface designs were also explored in reference to these system designs. During the development stage of the application, many decisions were discussed and implemented regarding user features and functionality. These choices have been logged accordingly.

Table of Contents

Executive Summary	ii
Change Log	5
1. Business and System Context	7
1.1 Relevant Business Information	7
1.2 System Context	9
2. Stakeholders and Requirements	10
2.1 Stakeholders	10
2.2 Requirements	12
2.2.1 Use Cases	12
2.2.2 Quality requirements	18
2.3 Key Driver Analysis	19
3. Acceptance Tests	23
3.1 User Personas	23
3.2 Behaviour-Driven Development Tests	23
3.2.1 Implemented in Cucumber	24
3.2.2 Implemented in Manual Tests	25
4. GUI Prototypes	27
4.1 Key GUI Design considerations	27
4.2 Mock-up software choice	28
4.3 Mock-up inspiration	28
4.4 Mock-up Feedback	28
Design Updates	31
5. Deployment Model	32
6. Detailed UML Class Diagram	34
7. Testing procedures	39
7.1 Functional Testing	39
7.2 Quality (NFR) Testing	44
7.3 Unit Test and Cucumber Test Coverage	48
8. Current Product Version	49
8.1 SafeTrip Version 1.0 – Deliverable 2 Snapshot	49
8.2 SafeTrip Version 2.0 – Finalised Product for Deliverable 3	49
8.2.1 Data Storage and Manipulation	49

8.2.2 Road User Routing.....	51
8.2.3 Analysis of Data.....	53
9. Risk Assessment	56
9.1 Generic Risks	56
9.2 Product Specific Risks.....	60
10. Project Plan	61
10.1 Trello	61
10.2 Major Milestones.....	61
10.3 Minor Milestones.....	62
10.4 Development Strategy	64
10.4.1 Feature Branching	64
11. Lessons Learned	66
Angelica.....	66
Chris.....	66
Neil.....	67
Todd.....	67
Will.....	68
Zipporah.....	68
12. References	69
13. Appendix	70
Appendix A – Sections Worked on by Members	70
Appendix B – Excel Sheets	71

Change Log

Section No	Section	Change	Person Responsible
<u>Deliverable 2</u>			
1	Business and System Context	Added screenshots from competitor apps, elaborated on unique selling points and added more detail to system context diagram.	Zipporah Price
9	Risk Assessment	Removed most product specific risks and added to quality requirements or general risks.	Angelica Silva
2.2.2	Quality Requirements	Removed the 7 th QR, collected usability related QR's together, reordered them based off the new KDA	Christopher Wareing
2.3	Key Driver Analysis	Removed the 7 th QR, changed some values to zero to better represent what is happening. Re considered why each value was chosen as so.	Christopher Wareing
7	Testing Procedures	New section	Zipporah Price, Angelica Silva
3	Acceptance tests	Split into acceptance tests to be implemented and implemented	Neil Alombro
<u>Deliverable 3</u>			
7	Testing Procedures	Proliferated manual tests to cover all acceptance tests.	Neil Alombro, Angelica Silva
11	Lessons Learned	New section	Angelica Silva
7.2	Quality (NFR) Testing	Re-made the entire Quality testing, updates the old tests with newer methods and added more tests.	Christopher Wareing
3	Acceptance Tests	Updated the acceptance tests by removing and adjusting certain tests based off of final product decisions.	Christopher Wareing and Neil Alombro
1	Business and System Context		Zipporah Price
4	GUI Prototypes	Added screenshots of the final version product, discussion of added features.	Zipporah Price
10	Project Plan	Updated project plan with a development strategy section detailing our use of git feature branching to increase workflow efficiency	William Thompson
2	Stakeholders and Requirements	Updated stakeholder reasoning with priority. Updated use cases with	Zipporah Price and Neil Alombro

		discussion on removal and addition of use cases since beginning of project.	
7.3	Testing Discussion	Re writing discussion based on updated testing. New screenshot and new discussion talking about changes since deliverable 2	William Thompson
8	Current Product Version	Updated so that the previous version was a short summary	Neil Alombro
5	Deployment Model		Todd Vermeir
6	UML class diagram		Todd Vermeir
7.1	Functional Testing	Updated with manual testing table	Angelica Silva
9	Risk Assessment	Reviewed the relevance of the risk assessment with our final project.	William Thompson
13.2	Appendix B – Excel Sheets	Updated and added links to excel spreadsheets	Angelica Silva

1. Business and System Context

1.1 Relevant Business Information

Road accidents are an unfortunate but seemingly unavoidable aspect of everyday life, and for cyclists and others who are more vulnerable on the road, this can be a scary fact to live with. But what if these road-users had a way to potentially lower their chances of experiencing an accident while still getting to their desired destination in a timely manner?

The proposal is a journey-planning software application that utilises data from records of previous crashes to inform users of the safest route to a given destination, equipping concerned road users with the necessary information to feel confident about their safety while using the road.

The key services provided to users will be:

- Presenting records of local road accidents in a user-friendly display: map view choices include heatmap, pinpoint locations, and a plain map.
- Filtering these crash records based on parameters such as time range, severity, weather, holiday, transportation mode and region boundaries.
- Generating routes based on user choices of locations transport mode.
- Presenting multiple route options to the user with relevant safety information, allowing user to compare the relative safeties and select which route they want to take.
- Allow users to add and delete multiple stops to the journey.
- Allow users to save a frequently used 'favourite' route and give the route an alias.
- Allow users to load and delete 'favourite' routes.
- Providing a pie chart graphical representation of key crash variables.
- Allow user to update these pie charts based on selected variable filters.
- Allow user to draw a boundary around an area and calculate a safety rating for the given area.
- Allow user to clear the database of crash records and import their own compatible records.

It must be acknowledged that there already exist many journey-planning apps that are currently used by the public; some of New Zealand's most popular travel apps include Google Maps and AT Mobile. Figure 1 shows an example of the routing functionality of Google Maps; clearly displaying a user-friendly, intuitive design which allows users to easily plan their journey. Its unique selling points are the fact that it is a popular and widely accessible journey-planning/mapping app, which has stood the test of time and has a reputation for being accurate and reliable. (Apple NZ, 2023)

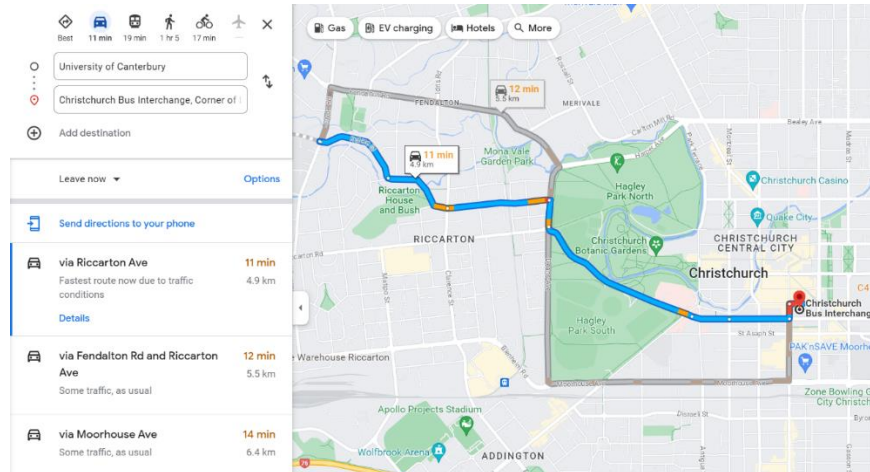


Figure 1: Google Maps Routing Functionality

Some applications were also found that implemented a crash analysis system such as Crash NZ and the Waka Kotahi Crash Data Map. Both these apps contain points located on a map, each one with specific data about a crash. Crash NZ also has the function of graphing crash data based on different characteristics, as shown in figure 2. This is a unique selling point that allows for a clear display of different crash statistics that users may have an interest in, such as graphing crashes by vehicle type, area, and others. As well as the general public, possible user demographics includes real estate agencies that may have a vested interest in which areas could be deemed as 'safer' according to crash statistics. (PhoneMe.Money Ltd, 2019)

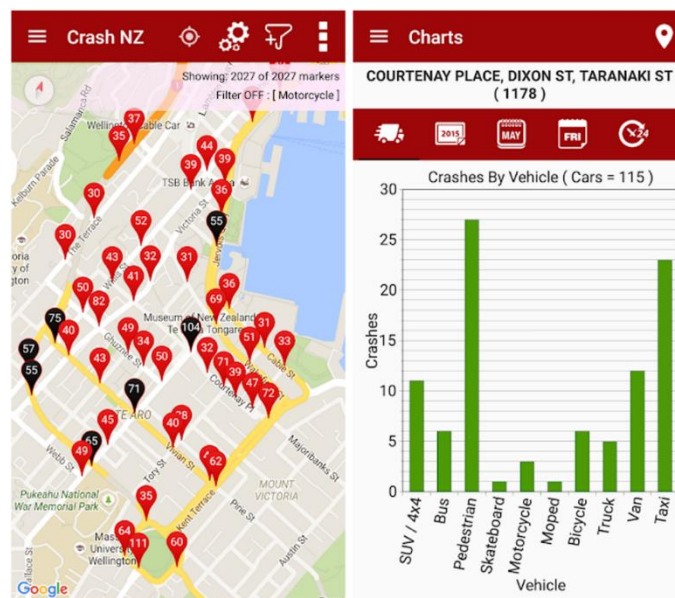


Figure 2: Crash NZ Mapping and Graphing Functionality

These competitors were taken into consideration when planning the SafeTrip intended demographic and app functionality. Both Google Maps and Crash NZ apps occupy areas of the market, but SafeTrip will bridge the gap. Google Maps has no data relating to crashes and safety in regard to routing capabilities, and Crash NZ has no journey/route planning aspect, which is an integral part of SafeTrip and

thus serves as a point of difference. The application will take aspects from both types of map applications as inspiration and implement them in a unique way. From the Crash NZ app, aspects of visualising the crash data as individual points on a map will be included, and from the Google Maps app, similar map navigation and routing capabilities will be utilised. Users will have access to past crash data and will be able to take this into account when planning a journey.

This gives the SafeTrip app a unique selling point and an advantage over competitors, as the many potential customers who have concerns about their safety on the road while still wanting to keep these journey-planning capabilities will be inclined to choose this application over those that do not offer all these services.

Not only will this application be exploring a new and profitable business territory, but it will have a positive impact on New Zealanders' lives as a whole. Since the local population of the Canterbury region has the largest percentage of cyclists throughout New Zealand, this app will aim to improve the lives of local residents as well as the overall New Zealand population. Cyclists and other road-users will be equipped with the knowledge of how accident-prone different areas are, thus allowing them to take potentially safer routes, in turn giving them peace of mind when it comes to their road safety. (Waka, 2023)

1.2 System Context

As shown in Figure 3, the envisioned application will be used by everyday road-users concerned with their safety on the road. These users will be able to choose their desired destination and other stated application functions, and then receive the different route options they can take based on safety ratings from the application system.

After selecting their destination, the application will use crash data retrieved from a csv file from the Waka Kotahi Crash Analysis System (CAS) to calculate a safety rating for each of the possible route options and display these ratings to the user. The application will interact with API data from a pre-established map system to present the routes in a user interface display.

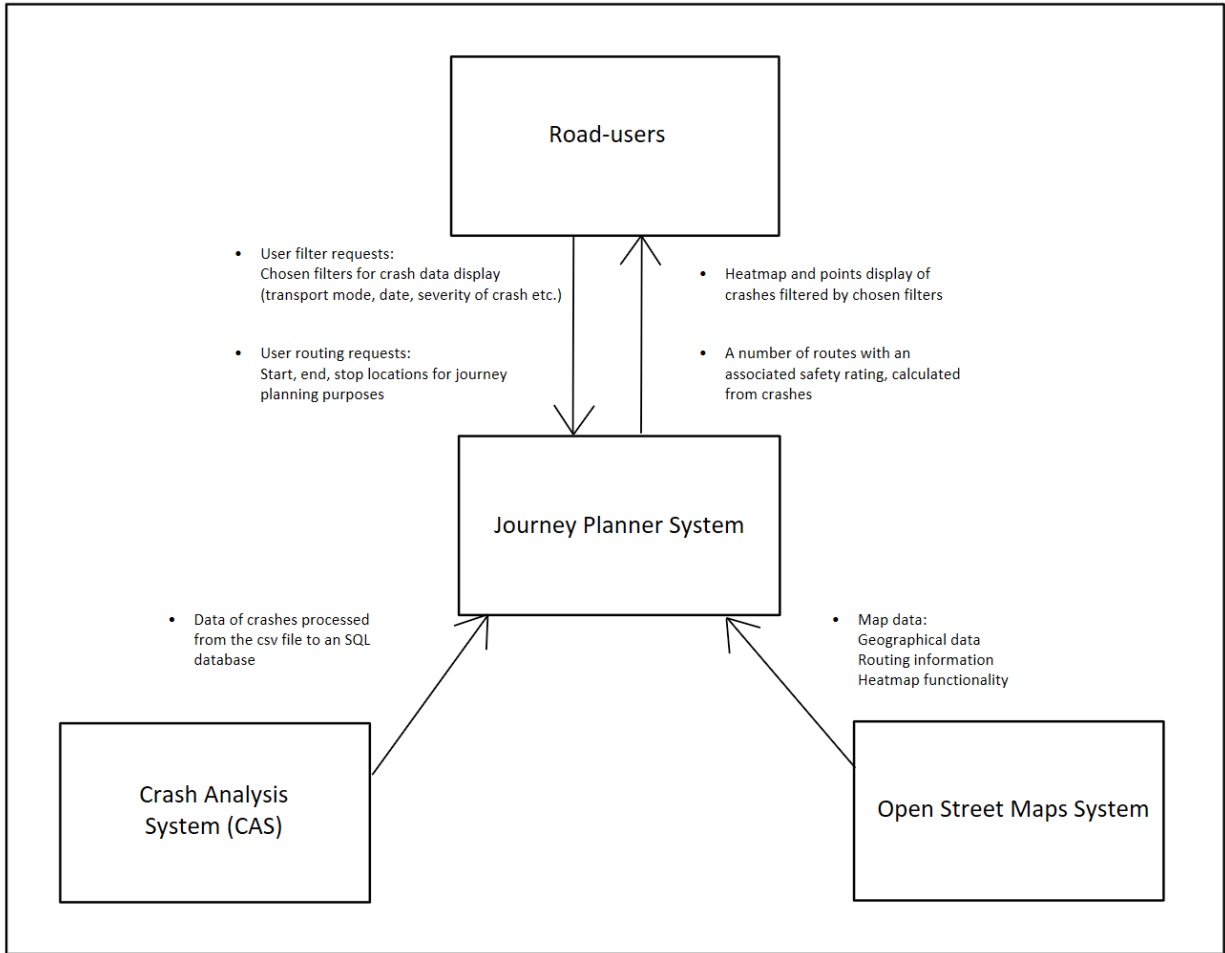


Figure 3: Application System in Context

2. Stakeholders and Requirements

Our software solution has many stakeholders and requirements with varying priorities that will affect how our software is designed and developed. The following cover these and the effects they have in how our product is shaped.

2.1 Stakeholders

These stakeholders cover direct users of the product, those who impact the development and design shaping, and those indirectly affected by the product's usage. Table 1 describes each stakeholder with their accompanying needs and concerns regarding the product. The stakeholder is then given a priority rating regarding the "power" (or influence) they have on how the product is designed and the "interest" they have in using the software. This rating scale goes from none, low, medium, and high. The way we interact with our stakeholders is based off the Power-Interest Grid. Those with High Influence and High Interest are those who we want to keep happy and satisfied with the progress. Those with High influence and Low Interest are those who we still want to keep happy, but we don't want to overcommunicate the details with them, as it may turn them off. The stakeholders with Low Influence

and High Interest are those who can provide invaluable suggestions and feedback, but we do not need to say yes to them. Finally, those with both Low influence and interest will require the least interaction with for our project. Medium is in between low and high. (Stakeholder Analysis, 2023)

In our analysis, we identified a specific group of stakeholders with low influence and/or low interest. This led us to question whether their needs and concerns were aligned with the goal of improving our product. Among these stakeholders were the New Zealand Transport Agency (Waka Kotahi), roadside business owners, police, rideshare companies, and insurance companies. Due to their minimal impact and interest in our project, we have chosen not to include them in the table below.

Table 1: Stakeholder descriptions and priorities

ID	Stakeholder (Description, needs, and concerns)	Priority	
		Influence	Interest
SH.1	<u>Everyday road users</u> The direct product user that our team would design towards. These would include parents, students, workers, pedestrians, etc. Needs of these users include having safe and fast routes, a GUI design that has ease of use and functionality, and data that is accurate. The main concern with these users would be acquiring and retaining their use of the app.	High - Highest frequency of use for our product.	High - Benefit to their daily journeys.
SH.2	<u>SENG202 Lecture Team and Student Cohort</u> The product created is a project prescribed by the university's SENG202 course. The SENG202 lecturing team and student cohort have a large influence in how the product is shaped as they are our main feedback as we progress in designing and programming the software.	High- The main source of feedback and marking.	High - Involvement in similar projects and marking.
SH.3	<u>Developer Team</u> The developer team needs to create a solution that completes the SENG202 requirements for academic performance and are concerned about making an app that solves a real problem rather than software without a need. This stakeholder is also influential as the skills and time available of this team constrains the development of the software.	High - Responsibility in programming app.	High - Responsibility in programming app.
SH.4	<u>Property owners and buyers</u> Property owners and buyers (primarily real estate and businesses) have an interest in the safety of the roads in their neighbourhood. Location is a main contributor in the decision-making process of buying or continuing to live in a house. The main concern would be showing misleading data and allowing real estate workers to influence the safety ratings with false reporting via the police or, if implemented, in the user account functionality. (Struyk, 2023)	Low - Not interested in routing so low influence overall.	High – Interest in impact on traffic and neighbourhood safety.
SH.5	<u>New Zealand Transport Agency (Waka Kotahi)</u>	High –	None -

	The NZTA provide us with the data that the app will be primarily using to recommend routes. If the NZTA ceased to release up-to-date or make the use of the data illegal, then the app would have old data to recommend routes with. However, since the current data to date has been made locally available, the stakeholder has no interest in the app.	Data source so influential in plausibility of product creation.	Product has no effect on Waka Kotahi.
SH.6	<u>Competitors/similar apps (Waze, Waka Kotahi, etc.)</u> Other apps already offer functionality that allow users to take into consideration variables other than the speed of the route. For example, the app “Waze” allows users to use live traffic data on certain roads to shape the route they take. Another example, the Waka Kotahi App allows users to view specific crashes on a map and view all the data relevant to that crash. Development would consider this to differentiate from these existing solutions and to create more meaningful suggestions to users.	High – Influence since affects differentiability of product in market.	Low – Start up and low experience so no interest in our product.
SH.7	<u>University of Canterbury</u> The team of developers are current students at the University of Canterbury, specifically the Computer Science and Software Engineering Department. The product created would socially influence the reputation of both entities and that directly affects student acquisition.	High – Guidelines and standard influence app development.	Low – Oversee many student projects.
SH.8	<u>Environment</u> With adding another variable to choosing the route in which drivers take, this could create longer routes than what they currently use. With the most recent household travel data, 79% depend on car travel. This could increase fuel use and, therefore, carbon emissions that would concern the environment. This might impact the governments promise to be net zero by 2050 as it would increase the number of emissions.	Medium – Scalability, environmental, and social impact shapes the product.	None – Non-Human entity, therefore, they have no interest.

2.2 Requirements

The following requirements of our software are split into use cases (please refer to Figure 4 below) and quality requirements with regards to the previously defined stakeholders.

2.2.1 Use Cases

The software solution’s high level use case is to be a road journey planner that considers both the safety and the speed of the route.

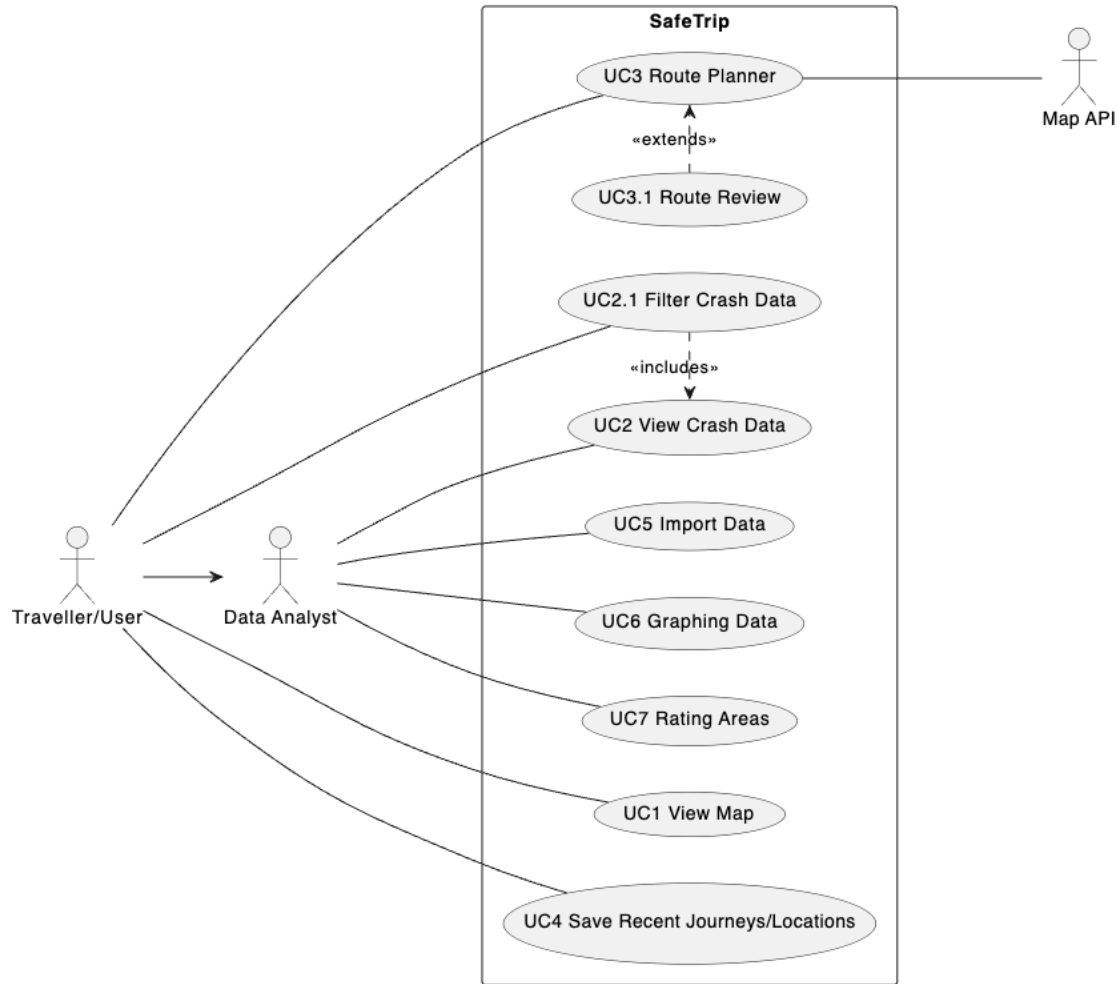


Figure 4: SafeTrip Use Case Diagram

Table 2 describes the actors involved in our diagram in further detail.

Table 2: Description of key actors

Actor	Description
Traveler/User	An everyday user of our app. Someone who wants to use the app for the route planning and map features.
Data Analyst	A person who is an instance of a Traveller/User, but also has a keen interest in the crash data for either personal or company reasons.
Map API	Provider of the Mapping and Route Planning system. Also, important to keep up to date.

Table 3 shows the textual use cases that cover the normal, alternative, and exceptional flows of each use case. Unless stated, the alternative and exceptional flows have the same pre- and post-conditions of the normal flow.

Table 3: Textual use cases

ID	UC.1	Name	View Map	Actor(s)	Traveller/User
Precondition(s)		App had been downloaded and jar file is on local machine.			
Normal Flow (and Alternative and Exceptional Flows)		<p><u>Normal Flow</u></p> <ol style="list-style-type: none"> 1. The user opens the app with the loading process starting and finishes successfully. 2. The system shows the landing page with the interactive map is shown on the screen with the default view of the whole New Zealand map visible. <p><u>Alternative flow – Map double click zoom interaction.</u></p> <ol style="list-style-type: none"> 1. The user follows the normal flow but does not move into journey planning or data analysis. 2. The user double clicks on the map and this results in the map view zooming into the clicked location. If the click is dragged, then the view will move towards the same direction. <p><u>Exceptional flow – User zooms out while whole New Zealand map is visible.</u></p> <ol style="list-style-type: none"> 1. The user will see zero effect from their action and see that the app currently only covers New Zealand. 			
Postcondition(s)		The app displays the map with interactive heatmaps relative to the zoomed view.			

ID	UC.2	Name	View Crash Data	Actor(s)	Data Analyst
Precondition(s)		The app either shows the map with heatmaps of all the crashes in the relevant data or a map with the recommended routes of a journey.			
Normal Flow (and Alternative and Exceptional Flows)		<p><u>Normal Flow (heatmaps)</u></p> <ol style="list-style-type: none"> 1. The user views the map with overlaying heatmaps with corresponding colours regarding the crashes in the area. <p><u>Normal Flow (crash locations)</u></p> <ol style="list-style-type: none"> 1. The user views the map with overlaying crash locations markers with corresponding severity colours. 2. The user clicks the pop up to see details about the specific crash. <p><u>Normal Flow (automatic changing view)</u></p> <ol style="list-style-type: none"> 1. The user views the map with overlaying crash locations markers at a high zoom in and an overlaying heatmap at zoomed out with corresponding severity colours. <p>Precondition – The app has journey information entered and processed with routes calculated and showing on the map.</p> <p><u>Alternative flow – Journey information entered and confirmed.</u></p> <ol style="list-style-type: none"> 1. The user clicks on any of the recommended routes. The initial view will show crash locations but can be toggled. 			
Postcondition(s)		The app displays the crash data in a way the user can digest.			

ID	UC.2.1	Name	Filter Crash Data	Actor(s)	Traveller/User
Precondition(s)	App has been opened and the map has successfully opened with crash heatmaps shown to the user.				
Normal Flow (and Alternative and Exceptional Flows)	<u>Normal Flow</u> <ol style="list-style-type: none"> 1. The user clicks the hamburger panel to show other filters available to the user. These filters include recency, severity, modes of transport to consider, holiday, region, and weather. These are toggled by sliders and dropdowns. 2. The user changes these filters sliders and dropdowns to change the data. 				
Postcondition(s)	The app's crash point data is filtered based on the given filtering information from the user.				

ID	UC.3	Name	Route Planner	Actor(s)	Traveller/User
Precondition(s)	The app has been opened and loading has been successful. The app has valid filter information with the relevant data used to visualize the heatmaps on the map.				
Normal Flow (and Alternative and Exceptional Flows)	<u>Normal Flow</u> <ol style="list-style-type: none"> 1. The user enters a source and destination for their journey. 2. The user has the option to enter extra locations as 'stops' along their route. 3. The user clicks on a 'generate route' button to find a route from the given source and destination. Based on the data provided, the user is given up to three possible routes with a review based on the route. <u>Exceptional Flow – Invalid source and/or destination.</u> <ol style="list-style-type: none"> 1. The user types in an invalid source and/or destination into the entry box. 2. The user clicks the route confirmation button. Postcondition – The app displays an error message to indicate that the given address is invalid. No route is generated.				
Postcondition(s)	The app displays a chosen route's directional information and a button to save the route.				

ID	UC.3.1	Name	Route Review	Actor(s)	Map API
Precondition(s)	The app has journey information about the source and destination and relevant crash data filters. The app has processed the route.				
Normal Flow (and Alternative and Exceptional Flows)	<u>Normal Flow</u> <ol style="list-style-type: none"> 1. The app shows a review with relevant data such as most dangerous segment, directions to and from, length of the route, etc. 2. The user can select between an x number of routes, dependent on the routes the router can find. 				
Postcondition(s)	The user is informed about the routes and can choose which route to take based on both safety, speed, and other relevant information deemed.				

ID	UC.4	Name	Save Recent Journeys/Locations	Actor(s)	Traveller/User
Precondition(s)	The user has been presented with recommended routes.				
Normal Flow	<u>Normal Flow</u>				

(and Alternative and Exceptional Flows)	<ol style="list-style-type: none"> 1. The user clicks the “Save Route” button located on the map screen. <p><u>Exceptional Flow – No route generated.</u></p> <ol style="list-style-type: none"> 1. The user clicks the “Save Route” button located on the map screen. 2. With no route, the user sees an error pop up.
Postcondition(s)	The user can now access the saved route for easy access instead of going through the journey planning use case.

ID	UC.5	Name	Import Data	Actor(s)	Traveller/User
Precondition(s)	The app has been opened and loaded. The initial crash information has been initialised and imported successfully in the loading process.				
Normal Flow (and Alternative and Exceptional Flows)	<p><u>Normal Flow</u></p> <ol style="list-style-type: none"> 1. The user clicks on the “Import Data” menu panel to reveal the options. 2. The user clicks the “Import” button. 3. The user, with the local file system pop up, clicks on a .csv file to import. <p><u>Exceptional Flow – Incorrect formatted crash information.</u></p> <ol style="list-style-type: none"> 1. The user follows either the normal or alternative flow to choose a file that has a format inconsistent with the Waka Kotahi’s data. <p>Postcondition – The app only has the initial data file supplied and not the chosen file’s data.</p>				
Postcondition(s)	The app has loaded and imported into the database the chosen .csv file’s data.				

ID	UC.6	Name	Graphing Data	Actor(s)	Traveller/User
Precondition(s)	The app has been opened and loaded. The initial crash information has been initialised and imported successfully in the loading process.				
Normal Flow (and Alternative and Exceptional Flows)	<p><u>Normal Flow</u></p> <ol style="list-style-type: none"> 1. The user clicks on the “Graphs” menu panel to open the graphing window. 2. The user selects a crash variable such as region, weather, severity to graph the data by. <p><u>Exceptional Flow – Graphing a region with no data.</u></p> <ol style="list-style-type: none"> 1. The user drags and zooms the map to the ocean. 2. The user clicks on the “Graphs” menu panel to open the graphing window. 3. The user selects the ‘map bounds’ option. 4. No graph will show, and an error message will appear in its place. 				
Postcondition(s)	A pie graph displays the crash data, graphed by the chosen variable.				

ID	UC.7	Name	Rating Areas	Actor(s)	Traveller/User
Precondition(s)	The app has been opened and loaded. The initial crash information has been initialised and imported successfully in the loading process.				
Normal Flow	<p><u>Normal Flow</u></p> <ol style="list-style-type: none"> 1. The user clicks on the “Rate Area” menu panel. 				

(and Alternative and Exceptional Flows)	<ol style="list-style-type: none"> 2. The user selects either the square or circle drawing button on the top right of screen. 3. The user clicks and drags over an area on the map to draw a boundary. 4. The user clicks the “Rate Area” button to rate the selected area’s safety. <p><u>Alternative Flow – Drawing over an area with no crash points.</u></p> <ol style="list-style-type: none"> 1. The user clicks on the “Rate Area” menu panel. 2. The user selects either the square or circle drawing button on the top right of the screen. 3. The user draws a boundary over an empty area of the map (e.g. ocean area). 4. The user clicks the “Rate Area” button. <p>Postcondition – The displayed danger rating will be 0, and the total number of crashes will be 0.</p> <p><u>Exceptional Flow – Not drawing a boundary.</u></p> <ol style="list-style-type: none"> 1. The user clicks on the “Rate Area” menu panel. 2. The user clicks the “Rate Area” button without drawing a boundary. <p>Postcondition – An error message will pop up, informing the user to draw a bounding area before rating the area.</p>
Postcondition(s)	The informational pane displays a danger rating for the area calculated out of 10, as well as the total number of crashes in the selected area.

2.2.1.a Use Case Discussion

During the application development process, the preceding set of use cases have been altered to include new use cases and to discard any cases that were not implemented. Such new use cases include UC.6: graphing data and UC.7: rating areas. Graphing data was added in the case of the main stakeholder everyday road users wanting to view an intuitive and user-friendly display of the crash data for all New Zealand, filtered by their choice of crash variables. The use case of rating areas was added as an interactive and easy way for users to find out the relative safety of any area on the map that they wish, as previously crashes could only be filtered based on region. This use case particularly pertains to the stakeholder of property owners and buyers who are likely to have a particular interest in the safety of specific areas.

The previous use case of exporting data has since been discarded as it was not seen to fit with the app stakeholders. Since disregarding data analysts as a high priority stakeholder, it seemed unnecessary to include exporting as a functionality since the main stakeholder of everyday road users were likely to only want to view the data within the context of the app. Another previous use case included the user being able to click on an area of the ‘Heatmap’ view and get a pop-up window with information pertaining to the clicked area. Since there is already an informational pop-up that occurs when a user clicks on an individual crash in the ‘Crash Locations’ view, it seemed redundant to also include this for the heatmap view since this view was meant for more holistic viewing purposes.

2.2.2 Quality requirements

As mentioned in section 2.2, a key driver analysis was undertaken to determine which of the following eleven quality requirements were most important and which stakeholders drive this priority. To keep consistent, the top three drivers for a particular requirement were chosen to be displayed within Table 4 below and a priority value was derived from the value in the key driver analysis. If one or more of the top three drivers have very little impact or interest and then they are disregarded, leaving only two stakeholders.

Table 4: Quality requirements with priority based on key driver analysis.

ID	Description	Stakeholders	Priority
QR.1	<u>Maintainability</u> The app must be able to be maintained easily. This includes keeping code modular and easily understandable as bugs appear over time. Along with this, it should be easy to update the information through the CSV files as this will happen yearly. Ensuring scalability means code is not hardcoded or dependent on a local machine, so that there is versatile use of the application as intended.	Development team, SENG202 team.	High
QR.2.1	<u>Usability - Responsive</u> The app should provide feedback to users whenever an operation is in progress. Examples of this include when filtering data the map should update to indicate the changes on the filters, when buttons are pressed, they should be depressed etc...	Everyday road Users, Property owners, SENG202 team.	High
QR.2.2	<u>Usability – Intuitive</u> The user interface should be intuitive and user-friendly, ensuring that new users can navigate and understand features without extensive training. This includes users being able to apply filters easily, navigate through the different windows without extra direction from the development team.	Everyday road Users, Property owners, SENG202 team.	High
QR.2.3	<u>Usability - UI Consistency</u> All user interface elements, such as buttons, menus, and dialog boxes, should have a consistent design and layout. For example, we are placing the hamburger on the top left of the screen as is consistent with other websites.	Everyday road Users, Property owners, SENG202 team.	High

QR.3	<u>Performance</u> The application must be responsive and efficient, ensuring quick load times and immediate feedback when users interact with it. Filters should update maps promptly, and the initial data processing should not result in long waiting times for the user.	Everyday Road users, Property owners, competitors.	Medium-High
QR.4	<u>Data Integrity</u> As the system deals with accident data, it should ensure that the data integrity is maintained. Even though personal data is not involved, the application should have mechanisms in place to prevent unauthorized data manipulation or access. Any data calculations in the application should be double checked and reasonable, and data should be imported properly.	Academic researchers, competitors.	Medium-High
QR.5	<u>Robustness</u> The app should handle all edge cases leaving no bugs in the final product. The user should be able to input whatever they want in a field and get appropriate feedback on whether what they have done is accepted. For example, when the user applies filters there should be no issues with displaying specific combinations i.e. high severity for busses.	Property Owners, Everyday Road users, SENG202 Lecture team.	Medium
QR.6	<u>Licensing</u> Our product must adhere to copyright laws and other relevant implications. We must ensure that NZTA gives permission to use the data from the CAS. We need to ensure the libraries that are used, give permission or are in the creative commons.	University of Canterbury, Development team, Academia and researchers.	Low-Medium

2.3 Key Driver Analysis

To discover which requirements are most important a key driver analysis was indispensable in understanding which factors have the largest influence on a particular outcome or objective. By pinpointing the primary drivers behind certain quality requirements, it was easily discovered which quality requirements had the highest priority in the development process.

A decision was made to exclude two Stakeholders from our key driver analysis, the Environment, and Waka Kotahi. The reason behind this is that the Environment and Waka Kotahi do not provide additive value to our key driver analysis. It is also important to note that the usability requirements were put together in the analysis as they will have very similar results which would only skew the results.

Table 5: Key driver analysis with quality requirements.

Stakeholders	Influence	Interest	Weight	Licensing	Performance	Usability	Robustness	Data Integrity	Maintainability	Total
Everyday road users	3	3	0.18	0	30	35	25	10	0	100
Property owners	1	3	0.12	0	25	35	25	15	0	100
Competitors/similar apps (Waze, etc.)	3	1	0.12	15	20	15	15	20	15	100
University of Canterbury	3	1	0.12	25	10	10	20	15	20	100
SENG202 Lecture Team and Student Cohort	3	3	0.18	15	10	25	10	15	25	100
Developer Team	3	3	0.18	20	10	10	10	15	35	100
Academia and Researchers	1	2	0.09	30	5	10	5	30	20	100
Total			1.00	2.97	4.11	4.36	3.95	4.03	4.48	100

Below is a brief discussion about the results of the key driver analysis.

Everyday road users – This stakeholder has an important role in being our largest stakeholder and also one of the least technical. This has led to QR's like licensing and maintainability to be zero while usability is very high. Data integrity remains at 10% because there is no personal data stored in the system.

Property owners - Property owners have very similar results to everyday road users however it was decided that they care less about performance and more about data integrity as potential misuse could lead to changes in property values.

Competitors – Competitors have an even spread of interest in most QR's as each strong point of our product they would have to either replicate or substitute. ie competitors care about the robustness of our product because it sets the bar for how robust their product must be.

University of Canterbury – The university particularly cares about licensing as we are representing them through this project, although they don't care about how great our product is they certainly care about whether it breaks the law. Aside from this they generally care less about usability, performance and robustness as they are not going to be using the app very often.

SENG202 Lecture team – It has been decided that the lecture team would care a reasonable amount about maintainability as they must analyse and mark the codebase. It would be very difficult to grade the system if the code was unmaintained. Other than this they have a general spread of importance on the remaining QR's however they care around 25% about usability because they will be using this app semi-often when grading.

Developer team – The developer team has placed key importance on maintainability and licensing, maintainability is important as it makes it much easier to develop the project in the long run. Licensing is important because if the team does not adhere to copyright laws then there is potential for fines.

Academia and Researchers – This group similarly cares about licensing for the reason that if any data was exported or systems used in potential reports they must ensure that everything is licensed. They care about data integrity also as that is the key thing they would take away from the application, and care a reasonable amount about maintainability as they would not want to reference the data from the application and then only a month later the application is gone.

3. Acceptance Tests

3.1 User Personas

Following on from the actors and use cases developed, personas have been identified following on from the different types of actors – please refer to Table 5. These personas allow the developers to create an app designed to all the different types of users.

Table 6: Description of personas

Persona	Description
Fixed Traveller	Bob travels to his building work on weekdays and always only uses his car for this journey. He is a 25-year-old male without much time, so he likes time-efficient tasks. Bob has found that his current route has many cars that often travel over the speed limit and makes his travels feel unsafe. His goal is to find a safer route to work without sacrificing too much travel time.
Flexible Traveller	Ash is a training student with flexible start and end times to his daily going-on. He has many modes of transport with a bike, a car, and a bus card to take the bus. Ash cares for his safety, especially when vulnerable on a bike, due to many animal-like creatures depending on him. His goal is to find a route from his house to university, gyms, training centres, and other commitments.
Pedestrian	Peppa lives close to work and walks to save money on transport. She has found that there is a lot of car traffic on the roads she walks to work on and is interested in the possible crashes that have happened along these roads. Her goal is to look at the safety of these roads in a visually appealing way for her to digest the data.
Home Enthusiasts	Marge is in the market to purchase a home. She has a family of five and is looking for considerations in buying her new home. She has noticed that the current neighbourhood she is looking to buy the house in has narrow roads and bends. Marge's goal is to find out how safe the roads that her children will be around will be, if she was to purchase this house.
Data Analyst	Phineas is looking at building a new rollercoaster. He wants to build this rollercoaster over roads with minimal crashes, so the building process is not affected. However, the roads being considered have been rebuilt to be wider in the past two years. Phineas' goal is to gather data and analyse it with regards to filtering by recency of the data and other possible filters.

3.2 Behaviour-Driven Development Tests

The following acceptance tests shown in Table 6 have been designed with behaviour-driven development in mind. The descriptions follow a “Given-When-Then” pattern for a tester to easily interpret where, what, and why we test. These are then related to the specific personas, given a criticality level ranging from low, medium, and high, then related to the use case.

Within the context of this app, all these personas will have access to all use case functionality and the following personas chosen dissects which use cases they are to have a likened interest to and would be testing.

3.2.1 Implemented in Cucumber

We have a limited number of implemented tests. These are detailed below.

Table 7: Given-When-Then Unimplemented Acceptance Tests

ID	Acceptance Criteria (Given, When, Then)	Personas involved	Criticality	Use Case(s)
AT.1	<u>Change the view of the map from None to heatmap.</u> <u>Given</u> the user has opened the app and the database is loaded, <u>when</u> the user changes the view, <u>then</u> the current view should be changed to the user selection.	Ash, Marge, Phineas	Medium	UC.2
AT.2	<u>User wants to filter by a column.</u> <u>Given</u> the user wants to filter crashes by a column, <u>when</u> the user deselects a number of checkboxes or sets the years for in between, <u>then</u> the user will see crashes without the filters deselected.	Bob, Ash, Peppa, Marge, Phineas	High	UC.2 UC.2.1
AT.3	<u>Review a given route for its safety based on crash data.</u> <u>Given</u> the user has a set of coordinates, roads, and distances, <u>when</u> the user generates a route, <u>then</u> the user should receive a review containing relevant metrics.	Bob, Ash, Peppa	Medium	UC.3.1
AT.4	<u>User wants to save the generated route as a favourite.</u> <u>Given</u> the user has a start and end location entered on the routing menu, <u>when</u> the user clicks save route and enters a unique name, <u>then</u> the named route is saved in the database.	Bob, Ash, Peppa	Medium	UC.4
AT.5	<u>User wants to load a favourite route.</u> <u>Given</u> there is a route saved called a unique name, <u>when</u> the user selects the saved route and clicks load route, <u>then</u> the route has a start location matching the selected favourite.	Bob, Ash, Peppa	Medium	UC.4
AT.6	<u>User wants to delete a favourite route.</u> <u>Given</u> there is a route saved called a unique name with a starting location, <u>when</u> the user selects the favourite route and clicks delete route, <u>then</u> the favourite route is deleted from the database.	Bob, Ash, Peppa	Low	UC.4
AT.7	<u>Import a CSV file to the app.</u> <u>Given</u> the user has a CSV data file saved on the device running the app, <u>when</u> the user imports the CSV file, <u>then</u> the database should be populated with data from the CSV file.	Phineas	Medium	UC.5

AT.8	<u>Show a pie graph filtered e.g by severity</u> <u>Given</u> the user has opened the app and has opened the pie chart, <u>when</u> the user changes the filter to a column name, <u>then</u> the graph updates to show data from the column name.	Marge, Phineas	Low	UC.6
AT.9	<u>Drawing and Rating an Area</u> <u>Given</u> the bounding box information for a rectangle or a circle, <u>when</u> the area is rated, <u>then</u> the rating is calculated for the area.	Marge, Phineas	Low	UC.7

3.2.2 Implemented in Manual Tests

These tests were implemented and tested manually. These are due to being more visually based and harder to implement on cucumber.

Table 8: Given-When-Then Unimplemented Acceptance Tests

ID	Acceptance Criteria (Given, When, Then)	Personas involved	Criticality	Use Case(s)
AT.10	<u>Map shows all New Zealand.</u> <u>Given</u> the app has loaded successfully, <u>when</u> the user looks at the map, <u>then</u> the map will show a map of New Zealand.	Bob, Ash, Peppa, Marge, Phineas.	High	UC.1
AT.11	<u>Map is interactive with zooming.</u> <u>Given</u> the app has loaded successfully, <u>when</u> the user zooms in or out, <u>then</u> the map respectively shows the map zooming in or zooming out from that location.	Bob, Ash, Peppa, Marge, Phineas.	High	UC.1
AT.12	<u>Map zoomed out past New Zealand does not zoom out.</u> <u>Given</u> the app has loaded successfully, <u>when</u> the user zooms out past all of New Zealand, <u>then</u> the map still shows New Zealand and does not zoom out.	Bob, Ash, Peppa, Marge, Phineas.	High	UC.1
AT.13	<u>Map has heatmap, crash locations, and automatic changing view.</u> <u>Given</u> the app has loaded successfully, <u>when</u> the user changes the view to heatmap, crash locations, or automatic, <u>then</u> the user will see the chosen view.	Bob, Ash, Peppa, Marge, Phineas.	High	UC.2
AT.14	<u>Map shows crash locations along routes.</u> <u>Given</u> the app has loaded successfully, <u>when</u> the user sends a start location and end location and selects	Bob, Ash, Peppa.	High	UC.2 UC.3

	<u>“GO” for a route, then the route will show with crash locations showing.</u>			
AT.15	<u>Route production shows a review with relevant information about the route in terms of safety and other information.</u> <u>Given</u> the user has produced a route, <u>when</u> the user looks at the route, <u>then</u> the user will see relevant information, including safety parameters, in a review box.	Bob, Ash, Peppa.	Low	UC.3.1
AT.16	<u>Route production with invalid source and/or destination shows error.</u> <u>Given</u> the user leaves the source and/or destination is empty, <u>when</u> the user selects “GO”, <u>then</u> an error shows saying invalid selection of locations.	Bob, Ash, Peppa.	Low	UC.3
AT.17	<u>Saving a route as a favourite.</u> <u>Given</u> the app has generated a route, <u>when</u> the user clicks save route, <u>then</u> the user sees the favourite saved in the saved list and can load it later.	Bob, Ash, Peppa.	Medium	UC.4
AT.18	<u>Failing to save a favourite when a route is not generated.</u> <u>Given</u> the app has not generated a route, <u>when</u> the user clicks save route, <u>then</u> no favourite is saved, and an error shows up.	Bob, Ash, Peppa.	Low	UC.4
AT.19	<u>Importing an invalid csv file into the app.</u> <u>Given</u> the user has an invalid csv file in terms of format, <u>when</u> the user clicks import and choosing the invalid file, <u>then</u> the app will not import the csv file and show an error.	Marge, Phineas.	Medium	UC.5
AT.20	<u>Show graphing with a given column.</u> <u>Given</u> the app has loaded, <u>when</u> the user enters graphing and chooses their chosen column, <u>then</u> the app will provide a pie chart of the data.	Marge, Phineas.	Medium	UC.6
AT.21	<u>Show an exception with graph that has no data points.</u> <u>Given</u> the filtering has no data points associated with query, <u>when</u> the user clicks the column name with no data points, <u>then</u> the app will show a message saying no points are shown.	Marge, Phineas.	Low	UC.6
AT.22	<u>Error handling with rating an area without drawing an area.</u>	Marge, Phineas.	Medium	UC.7

	Given the user not drawn a bounding area, <u>when</u> the user selects the rating area button, <u>then</u> the user will see an error message saying to draw an area before rating an area.			
--	---	--	--	--

4. GUI Prototypes

Before getting started with prototyping, a list with key considerations to think about was made.

4.1 Key GUI Design considerations

Color Palette:

- Choose a colour scheme that effectively communicates the safety levels of different areas or roads on the map. Consider using colours that intuitively convey risk, such as green for low risk and red for high risk. We also considered accessibility for color-blind people, making sure to use contrasting colours.

Data Visualization:

- Decide on the most appropriate way to visualize safety data on the map (heatmap, colored roads, etc.) based on the readability and clarity of the information.

Iconography:

- Design clear and meaningful icons to represent crash data points on the map, such as accidents, traffic incidents, etc.

Map Interaction:

- Consider the interaction design for users to access detailed safety data. Decide whether clicking on a specific area or road should display relevant crash information.

Typography:

- Select fonts that are easy to read and suit the app's overall aesthetic.

Information Hierarchy:

- Organize the app's interface to prioritize the display of safety-related information while ensuring a clean and uncluttered design.

Map Labels:

- Determine whether to display road names, landmarks, or other relevant information on the map while avoiding visual overload.

Legend and Key:

- Include a legend or key to explain the color coding or symbols used to represent safety levels, ensuring users can interpret the information correctly.

Responsive Design:

- Ensure the app's design is responsive and adapts to different screen sizes and orientations, providing a consistent experience across devices.

Onboarding:

- Design an informative and engaging onboarding process to introduce new users to the app's safety features and functionality.

4.2 Mock-up software choice

For making mock-ups of the GUI, a program called Figma was deemed the most suitable; this software was chosen due to its ease of use for making good-looking prototypes and the ability to make the prototype interactive. Another key feature of Figma is its ability to work with JavaFX, and it can provide code snippets of elements created in Figma; for example, if a nice-looking button was made in Figma, it can provide the code snippet for how this can be done in Java. There is also the option to export the asset, such as a button as an image which can then be used in JavaFX also.

4.3 Mock-up inspiration

Before starting any mock-ups, a Word document was created with lots of images of similar applications to get ideas for the application design. When trying to find images for inspiration, we found that apps designed for mobile phones tended to have much nicer designs and more user-friendly, even though our final app will be made for desktop, we want to model it around a phone interface as it is nicer to look at and more intuitive. To get the images, a design website called dribbble.com was searched with relevant terms to the project, for example, 'JourneyPlanner' and 'Maps'. (Dribbble, 2023)

Figure 5 displays a collage of some of the images that were in this design inspiration document.

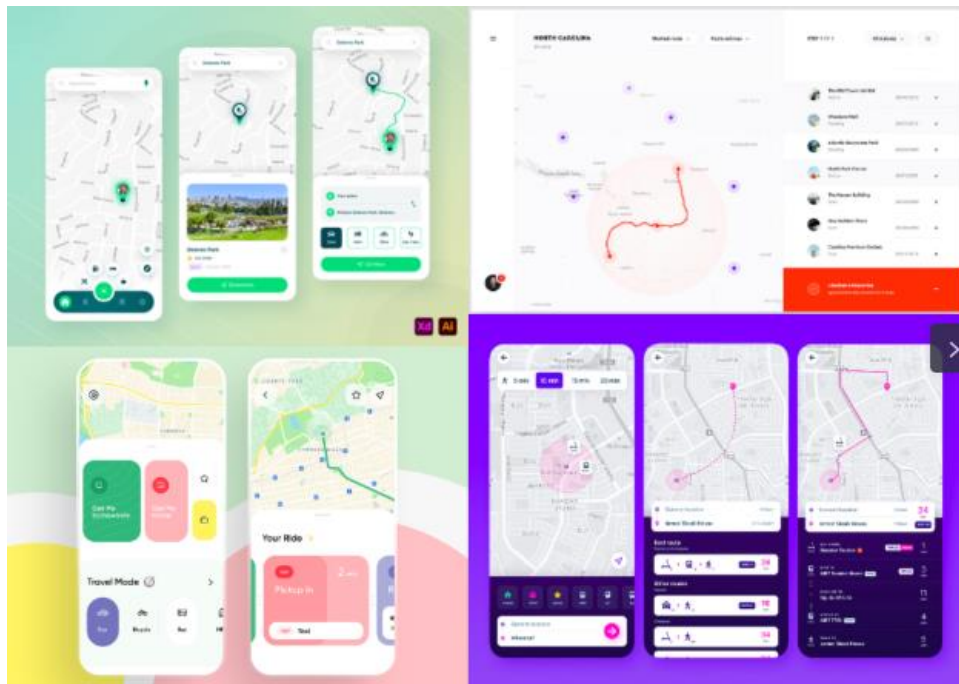


Figure 5: Existing GUI for inspiration

4.4 Mock-up Feedback

Before finalising any designs, the team reviewed certain design choices. This meant everyone was on the same page with how the mock-ups and the final outcome would look like. An example of this was

deciding how to show the safety of routes on the map. Figure 6 is a screenshot of the feedback document.

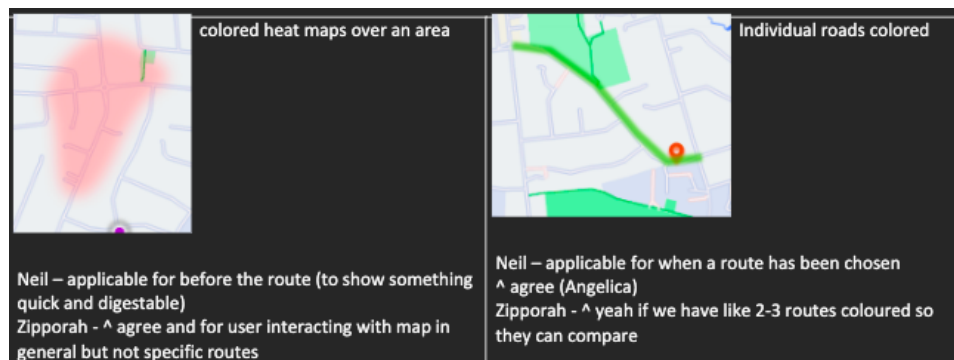


Figure 6: Feedback on how to visualise data

The feedback given in the document was then taken and a mock-up was created. Figure 7 is the final mock-up of the data display, after feedback was given.

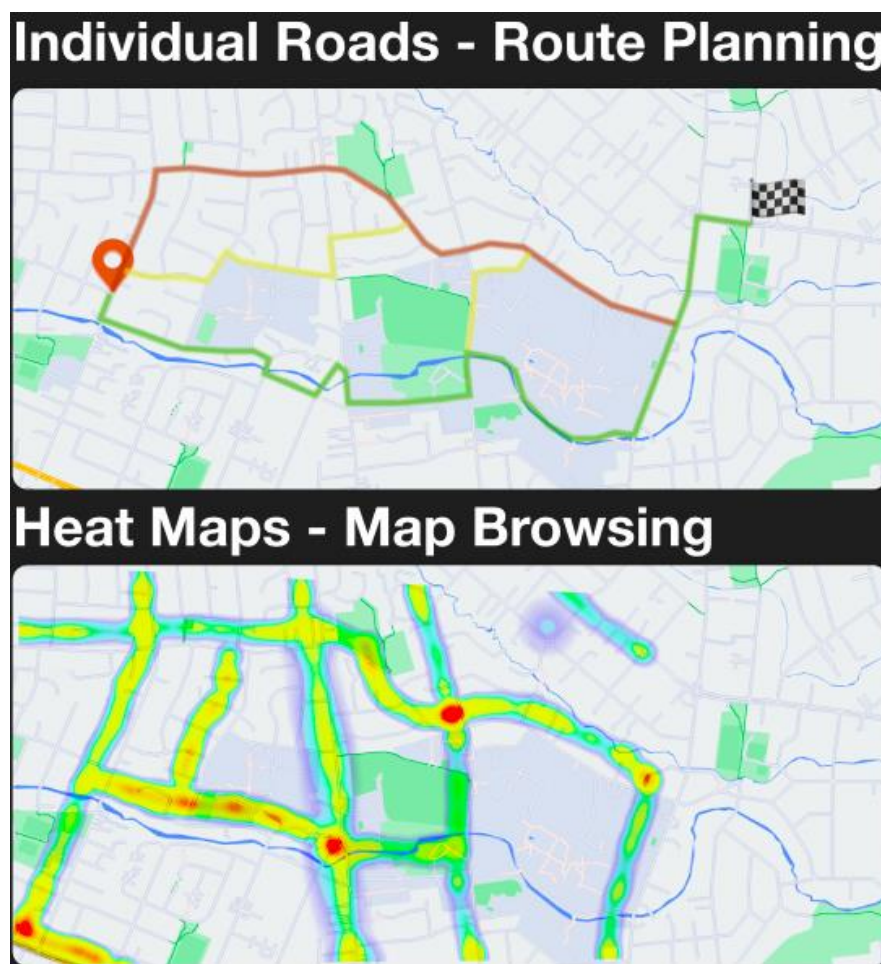


Figure 7: Implementing the two different visualisations in different contexts

Figure 8 displays what the final app would look like on the landing page.



Figure 8: Current GUI prototype for the app

Figure 9 displays the current prototype for clicking on the hamburger menu to filter the data.



Figure 9: Current prototype for hamburger menu

Design Updates

Throughout the development stage of the application, considerable changes have been made to the GUI design – features are now perfected to create a cohesive final product. Displays of the final version's menu screens are shown below in figures 10, 11, and 12.

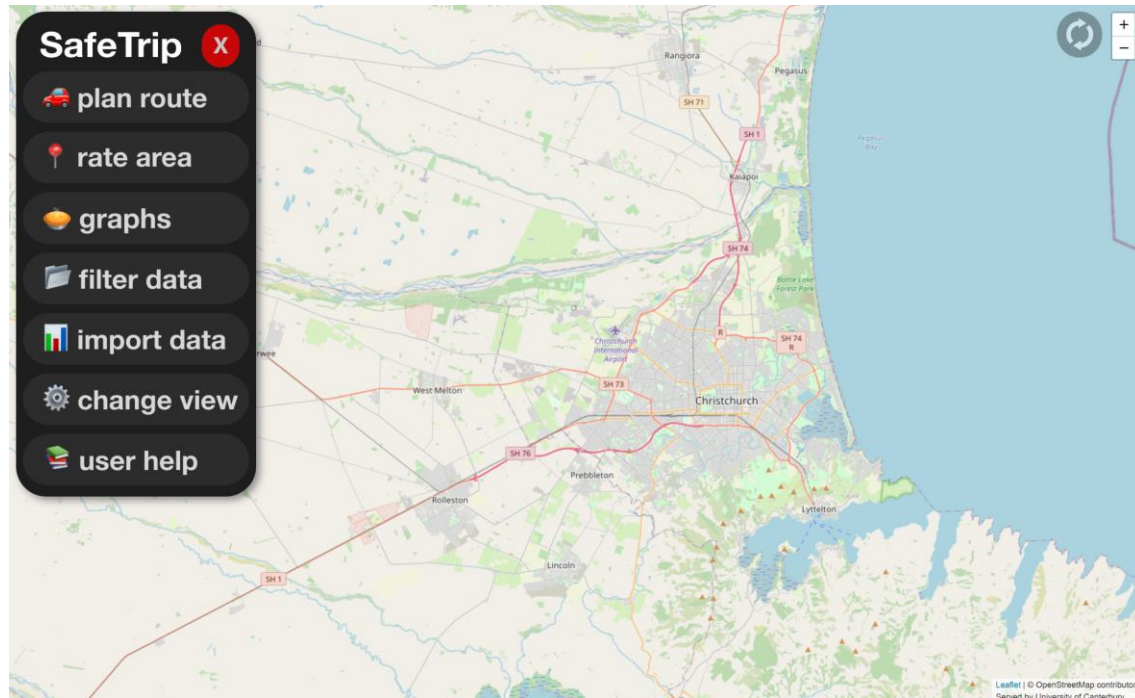


Figure 10: Screenshot of SafeTrip main screen.

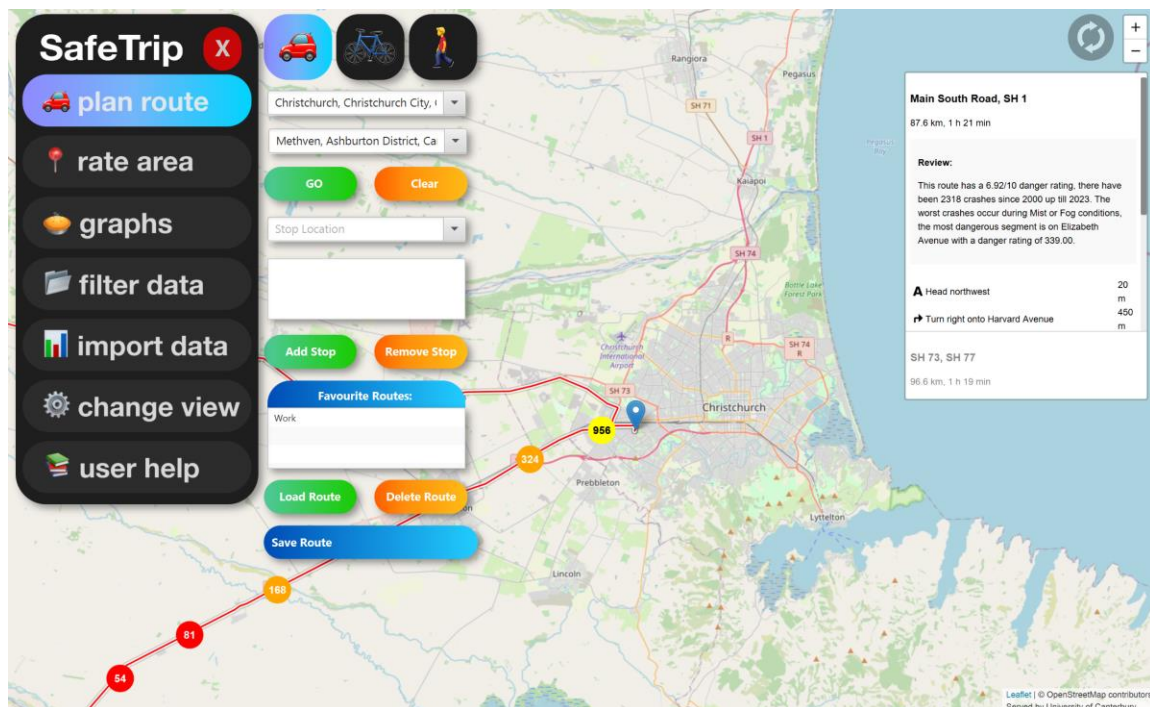


Figure 11: Screenshot of routing functionality.

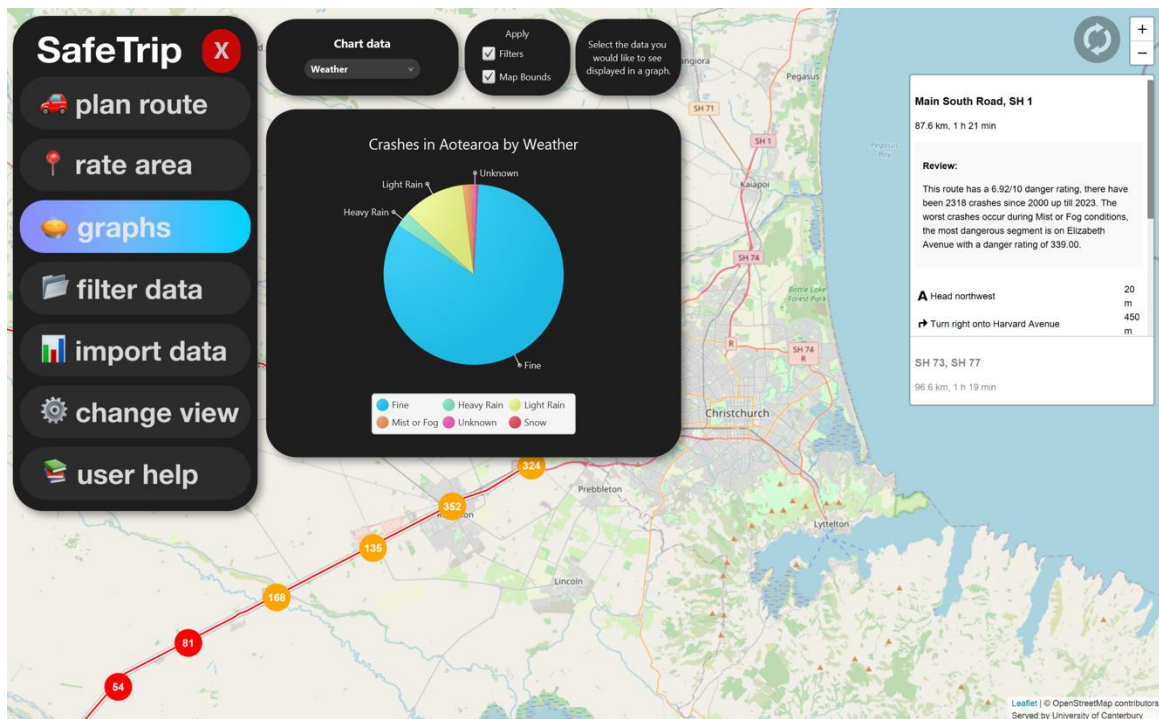


Figure 12: Screenshot of graphing functionality.

A major selling point of the app is its unique routing functionality. The ability for users to save routes under a chosen alias and select the route from the favourites list allows for ease of access and reduces redundancy in the scenario of a user having a frequently used route. Routes display an informational panel conveying the relative safety rating calculated for the route, as well as displaying crash events that overlap with the generated route.

As shown in figure 12, another major functionality that has been implemented is the displaying of crash data in the form of a pie chart. Users can choose crash variables and filters and the pie chart will display these changes accordingly. This design feature adds a more concise way to visualise the data, allowing users to gain a more insightful visual of the data.

5. Deployment Model

Given that the app is made for desktop, a local PC (for example, a lab computer) is needed for the client to run the software through the application server. Deployment must be made with the Jack Erskine software suite in mind, so in terms of software artifacts, a suitable operating system (Linux Mint 21.1), JRE (Java 17), the source code and the app's executable file are required (all contained within the local device). The use of Gradle 7.5.1 ensures maintainability and makes it easier for the team to build the app. Importing the JavaFX library and having the CAS data csv file on hand are also required.

While the application is designed for a single user, the use of Java and JavaFX allows the app to be web-based or Android in the future, making the app scalable. To connect the app server to the map server, a suitable API and internet access is required. The aspects of the map that our product uses includes routing, tilemaps, clustering and heatmaps. All require a valid internet connection. Because the user

only needs a local PC to run the app, the software is portable and accessible from devices such as laptops.

Figure 13 displays a UML representation of the deployment model, from a client perspective.

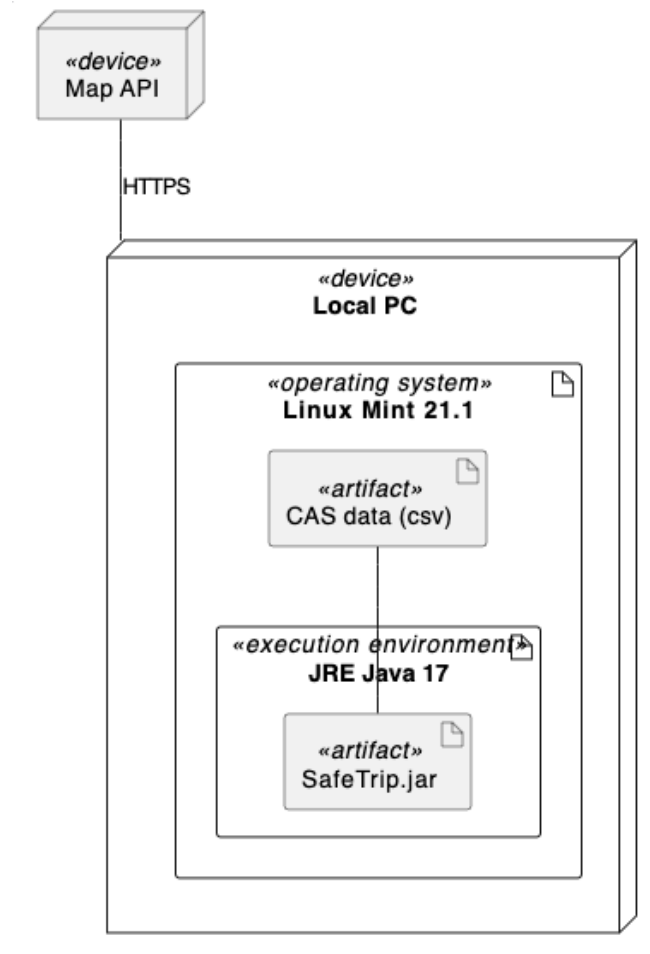


Figure 13: Deployment Model Diagram

6. Detailed UML Class Diagram

The UML class diagram uses the MVC software design pattern. The figure below is a brief overview of how our packages interact with one another in the MVC design pattern. Our view package sends user input to our Controllers, our controllers manipulate our models before the models update the views. The smaller UML diagrams below show the more intricate interactions between our different packages and classes. The business layer handles the main logic for the controller classes, such as setting and getting useful variables.

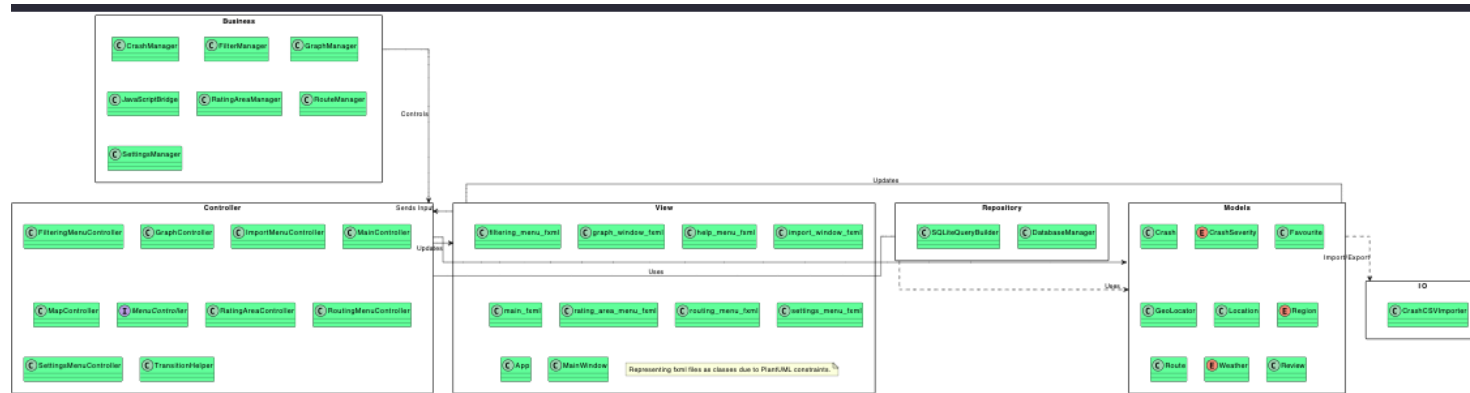


Figure 13: UML Class Diagram

In *Figure 15* below, the process for filtering data is shown via the interactions between the Model, View and Controller packages. The FilterManager class in the Business package manages the filter operations by keeping track of the various filtering parameters such as the Severity of the crashes. The FilteringMenuController class manages the user interactions that get returned from the View Package, before manipulating /updating the data stored in the models when necessary. The relevant data is then presented to the fxm file so the user can see it.



Figure 15: UML Class Diagram for filtering.

In *Figure 16* below, the process for updating the map settings/changing the views is shown via the interactions between the Model, View and Controller packages. The SettingsMenuController takes input from the View package such as removing and adding the Crash Location layers. The input allows the SettingsMenuController to display the right layer, by using aspects of the Crash class in the Models packages. The crash is then displayed in the view, depending on the map view selected.

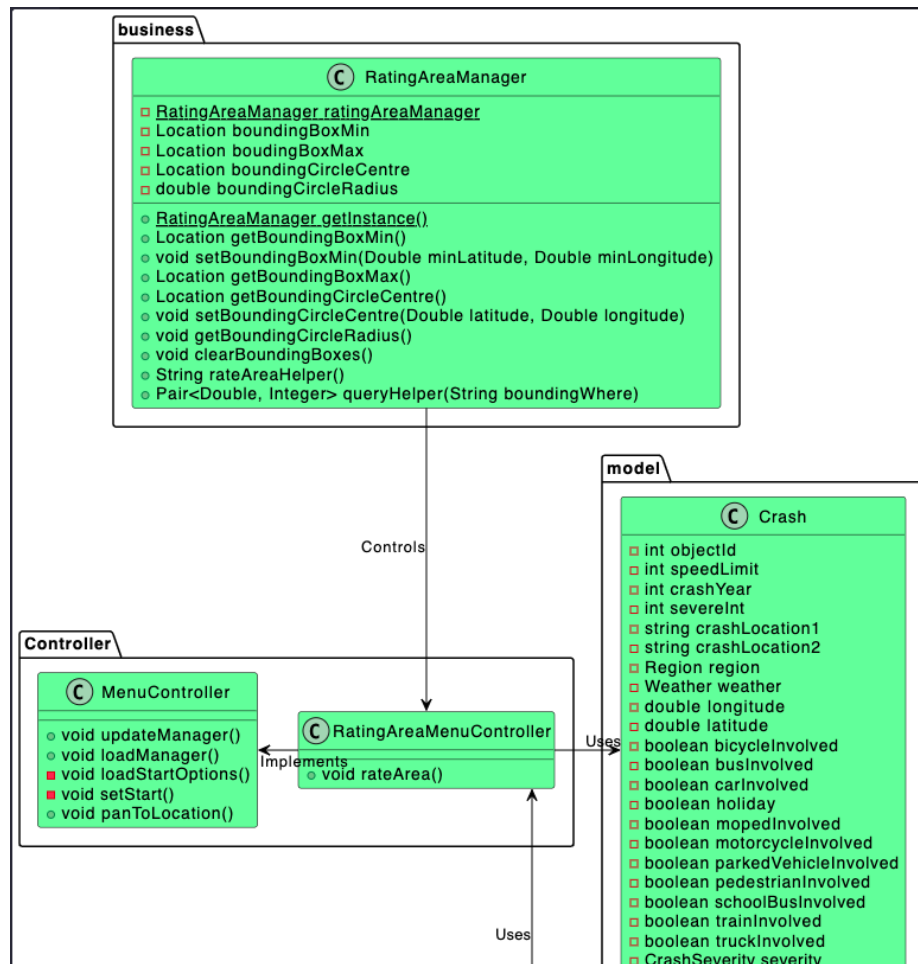


Figure 1614: UML Class Diagram For Changing the Views

In Figure 17 below, the process for managing and displaying the crash data is shown. The main.fxml is the primary view for the app. The MainController contains various methods to manage and respond to user interactions with the fxml file. It does things such as loading the g aph view, toggling menus and interfacing with the CrashManager class to handle logic and maintain crash-related states. The CrashManager class interfaces with the MainController to handle the logic related to managing crash data. The main controller uses the Crash class to accurately display and use the data for things such as viewing on the map and routing.

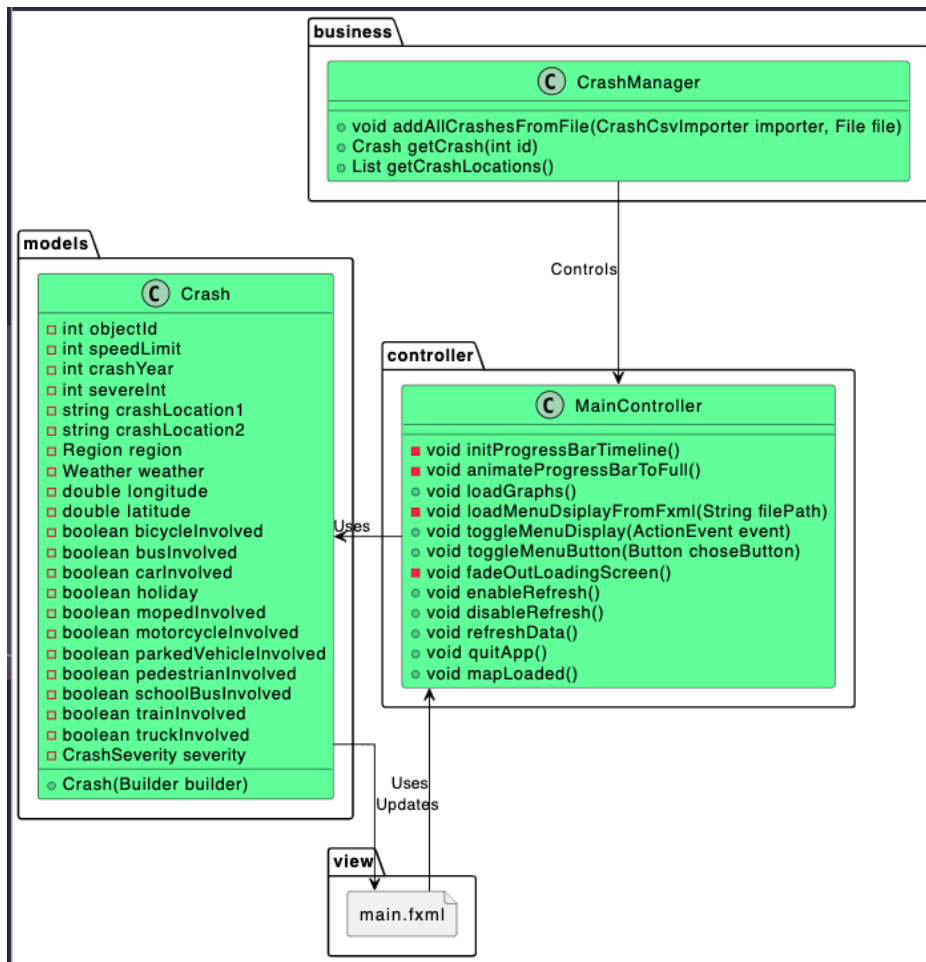


Figure 1715: UML Class Diagram For Managing and Displaying the Crash Data

In Figure 18 below, the process for querying a route and displaying it on the application is shown. The routing_menu.fxml file is the view package in this case. It gets the input from the user regarding addresses and sends it to the controller, which queries the modes, for example getting the latitude and longitude from the Geolocator class. The updated variables are then sent to the fxml file, where they are displayed correctly for the user to see. The RoutingMenuController is controlled by the RouteManager class, which contains methods to find the safety and crashes along a route.

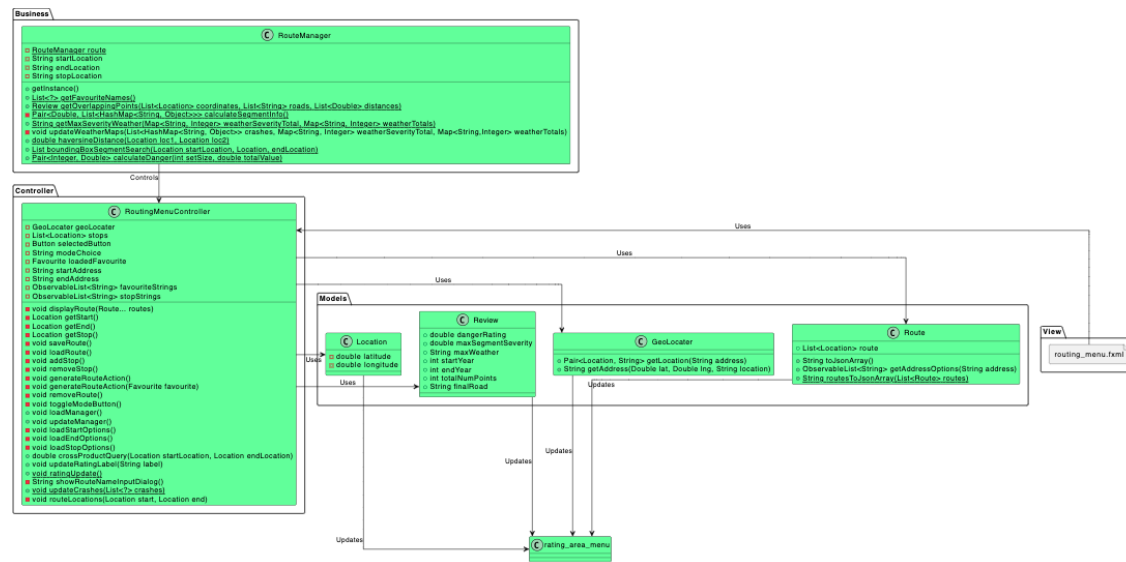


Figure 18: UML Class Diagram For Finding and Displaying a Route

7. Testing procedures

7.1 Functional Testing

Manual testing was undertaken throughout the coding process, with a high priority placed on making sure the database was being created and read from correctly, and that routing and map was working as expected. The tests mentioned below are selected for their relevance; for example, smaller tests such as checking the help window opened are not documented in the table below. The developers also used a testing log to document the latest outcome of any manual tests and add screenshots, actual result and the commit number tested.

Table 8: Functional Tests

ID	Description	Steps	Acceptance Test(s)	Criticality	Expected Result	Latest Tester	Latest Date	Latest Pass/Fail
MT.1	The map is successfully displayed to the user.	1. Load app. 2. Look at the map.	AT.10	High	Map shows all of New Zealand to the user.	Neil Alombro	15/10/2023	Pass
MT.2	The map is clickable and double clicking zooms into the map.	1. Load app 2. Double click a location on the map displayed	AT.11	Low	Map zooms into the selected location.	Neil Alombro	15/10/2023	Pass
MT.3	The map is zoomable by drag with 2 fingers	1. Load app 2. Mouse over a location 3. Pinch inwards on the trackpad and/or the touchscreen 4. Then pinch outwards on the trackpad and/or the touchscreen	AT.11	Medium	Inwards pinch should zoom out from the map, while the outwards pinch should zoom into the map on the selected location.	Angelica Silva	16/10/2023	Pass
MT.4	The map shows heatmaps when zoomed out and crash	1. Load app 2. Ensure the map is on automatic view	AT.13	Medium	Past 70% zoom out, the map shows heatmap, and zooming in	Neil Alombro	15/10/2023	Pass

	pin location clusters when zoomed in.	3. Zoom out past 70% 4. Zoom in past 70%			past 70% means the map should display relevant crash pin clusters.			
MT.5	The map does not zoom out past showing all of Aotearoa	1. Load app 2. Keep zooming out as far as you can	AT.12	Medium-High	When the user tries to zoom out past the view of all of New Zealand, the map display should stop zooming out and continue to show the whole map of new Zealand.	Angelica Silva	16/10/2023	Pass
MT.6	Filtering the data with all filter information provided works.	1. Load app 2. Click the hamburger panel to reveal the filter options 3. Provide transport, weather, severity, recency, region and holiday filter information - select all boxes except for Wellington region and Motorcycle boxes 4. Click 'Apply filters'	AT.2	High	The user will have a map view with the filtered data correctly based on all the information given, not showing data from Wellington and Motorcycles.	Angelica Silva	16/10/2023	Pass
MT.7	Filtering the data with some of the filter information provided works.	1. Load app 2. Click the hamburger panel to reveal the filter options 3. Provide transport, recency and severity information but leave the holiday, region and	AT.2	High	The user will have a map view with the filtered data correctly based on the information given, i.e. not showing any data	Neil Alombro	15/10/2023	Pass

		weather filters all unticked. 4. Click 'Apply filters'			since all regions are unticked.			
MT.8	Routes are clickable and show information relevant to the selected route.	1. Load app 2. Click the 'plan route' button in the sidebar 3. Enter a start location 'University of Canterbury' and an end location 'Chch airport' 4. Click the 'Generate Route' button 5. Click on a route displayed	AT.15	Medium-High	The app should display a review based on crashes on each route.	Christopher Wareing	16/10/2023	Pass
MT.9	Route planner has a start and end location entered and shows up to 3 routes to the user.	1. Load app 2. Click the 'plan route' button in the sidebar 3. Enter a start location 'University of Canterbury' and an end location 'Chch airport' 4. Click the 'Generate Route' button 5. Click on a route displayed	AT.14	High	one to three routes are displayed on the screen with a unique set of directions and a catered review to the route.	Christopher Wareing	16/10/23	Pass
MT.10	Route information being entered will show an error message if either source or destination is invalid (ie routing cannot	1. Load the app. 2. Input route information from 40 Little Oaks Drive as "source" and hjqwbq in "destination" 3. Click "Generate Route" button.	AT.16	Medium-High	Error window opens up that informs the user that the route requires a valid address for each box.	Christopher Wareing	16/10/23	Pass

	map the information provided to a coordinate on the map).							
MT.11	The user can save journeys for later usage.	<ol style="list-style-type: none"> 1. Load the app. 2. Input route information from 40 Little Oaks Drive to University of Canterbury. 3. Click "Generate Route" button. 4. Click "Save Route" button. 	AT.17	Low	Route is stored in the dropdown of favourite journeys and can be loaded up with relevant filters.	Will Thompson	16/10/2023	Pass
MT.12	The file chosen for importing is not the right format (ie not a .csv file)	<ol style="list-style-type: none"> 1. Load app. 2. Click the "Import/Export" button in the menu. 3. Click "Import file" button. 4. Import an empty text file named "text.txt". 	AT.19	Medium-High	Error window opens up that informs the user that the file format is incorrect.	Will Thompson	16/10/2023	Pass
MT.13	Inputting an invalid stop address in the route planning	<ol style="list-style-type: none"> 1. Load app. 2. Click plan route. 3. Type "jdjdjdjdjdj", "sksksksk" and "invalid" into start, end and stop locations respectively. 5. Press 'Go' and 'Add stop' 	AT.16	High	User is informed that invalid addresses were entered and is told which ones are invalid. The Stop location error popup works.	Angelica Silva	16/10/2023	Pass
MT.14	Graphing data by crash variable	<ol style="list-style-type: none"> 1. Load app. 2. Click the "Graphs" button in the menu. 3. Select the variable 	AT.20	High	The graph pane will update to display a pie chart	Zipporah Price	16/10/2023	Pass

		under "Chart data" you wish to graph by			graphing the chosen variable.			
MT.15	Applying filters to graph data changes the pie chart accordingly	<ol style="list-style-type: none"> 1. Load app. 2. Click the "filter data" button and apply some variable filters. 3. Click the "Graphs" button in the menu. 4. Select the filtered variable you wish to graph by 	AT.20	Medium-High	The graph pane will update to display a pie chart graphing the chosen variable, updated with the chosen filters	Zipporah Price	16/10/2023	Pass
MT.16	Graphing a region with no crash data shows an error message instead of a pie chart	<ol style="list-style-type: none"> 1. Load app 2. Drag the map to an area without crash data (e.g. ocean) 3. Click the "Graphs" button in the menu. 	AT.21	Low	No graph will be displayed in the graph pane, a descriptive error message will show in its place.	Zipporah Price	16/10/2023	Pass
MT.17	Rating the safety of a region	<ol style="list-style-type: none"> 1. Load app 2. Click the "rate area" button in the menu. 3. Click the square or circle button in the top right corner. 4. Draw a bounding box over desired region. 5. Click the "Rate Area" button 	AT.9	Medium-High	The information panel will update, showing a calculated danger rating as well as the total number of crashes in the area.	Todd Vermeir	16/10/2023	Pass
MT.18	Rating an area with no crash points has no safety rating	<ol style="list-style-type: none"> 1. Load app 2. Zoom and drag the map to an area with no crash data (e.g. ocean) 3. Click the "rate area" button in the menu. 	AT.9	Low-Medium	The danger rating displayed in the information panel will be 0/10, and the number of	Todd Vermeir	16/10/2023	Pass

		4. Click the square or circle button in the top right corner. 5. Draw a bounding box over desired region. 6. Click the "Rate Area" button			crashes in the area should be 0.			
--	--	---	--	--	----------------------------------	--	--	--

7.2 Quality (NFR) Testing

Quality testing occurred during the very end of the development process. The reason for this was that most of the quality requirements were not added to the project until very late as they are less important than the functional requirements. Licensing was excluded from the Quality testing as it is very low priority and difficult to quantify as was feedback from the second deliverable.

Table 9: NFR Tests

ID	Description	Steps	Quality Requirement	Criticality	Expected Result	Latest Tester	Pass Fail	Notes

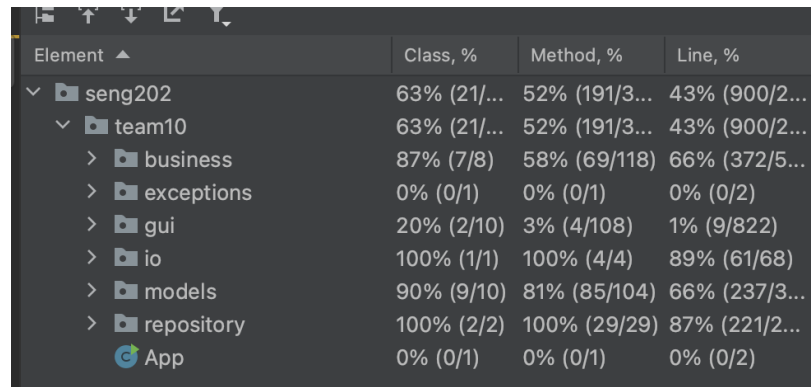
QT.1	Testing updating the SQLite database by inserting, querying and removing data from the database. Also ensure all check style is passing, meaning everything is consistent and easy to maintain. Ensure there are no hardcoded values, things should be modular.	Insert a new record into the database and check it can be retrieved. Delete a record from the database and check it cannot be retrieved. Run the check style.	QR.1	High	You should be able to add a record to the database, and when you remove one it shouldn't be in there. When you run the check style ensure there are no warnings.	Neil Alombro	Pass	
QT.2	Responsiveness, validate that the app provides appropriate feedback on user interaction, such as pressing buttons, loading a route etc...	Press a variety of buttons, combo Boxes etc... change the view to crash locations while fully zoomed out.	QR.2.1	High	When you press the buttons, interact with combo Boxes etc... it should depress, change color. When you change the view it should appear with a loading screen after a second.	Christopher Wareing	Pass	
QT.3	Ensure the navigating and using the application is intuitive.	Give the program to someone and tell them to: Navigate to the help page, generate a route and look at the review, close the app.	QR.2.2	High	The user should be able to get to the help page by pressing the help button, they should generate a route by going to 'Route' then type in a start and end address and press go, then look at the right. They should then press the X to close the app.	Zipporah Price	Pass	We handed the program to a range of Uni students, friends and family.

QT.4	Ensure UI elements are consistent across different display sizes, and sections of the application.	Navigate the app keeping note of the styling of each section, sections being the different options on the sidebar. Try to minimize the screen size to as small as possible.	QR.2.3	High	UI elements fit the same style, e.g. the black background border with similar colored buttons. When the user decreases the size of the window it stops after the screen becomes smaller than 1280x800.	Will Thompson	Pass	
QT.5	Ensure the app maintains responsiveness and efficiently processes data, for example when generating a route, it should not take any longer than 7 seconds.	Enter a start and end location, press 'go' to generate the route. When you press go start a timer, when the review / directions appear in full stop the timer. Check the time is reasonable.	QR.3	High	The time should never exceed 7 seconds for route generation, it should be noticeably faster the shorter the route, or the smaller the viewport.	Neil Alombro	Pass	7 seconds' was decided because it is the time it takes from cape Reinga to bluff (the longest route possible in terms of crashes)
QT.6	Ensure the application maintains data integrity during the imports and calculations used with the database.	Import data and perform operations such as generating routes. Compare the expected values for non-imported	QR.4	Medium	check that adding more crash points and generating a route should never decrease the number of points.	Angelica Silva	Pass	

		data with the imported data						
QT.7	Ensure the app has the ability to handle all user inputs and edge cases without crashing or producing incorrect outputs.	Use the app and try typing random things into all of the textbox or combo boxes. Then try to run the operation associated with it. Try all kinds of filters with all different features, ie routing, rating area, graphs.	QR.5	Medium	The app should not crash and give the appropriate result, for example with routing and an invalid address it should not generate a route and the popup should appear saying the address is invalid.	Todd Vermeir	Pass	

7.3 Unit Test and Cucumber Test Coverage

The overall class coverage for our project was 63%, with the overall method and line coverage at 52% and 43%, respectively. After deliverable 2, we refactored our controllers to separate concerns from GUI-related methods and the backend. This allowed us to better see how much of our business logic was being tested, as we would use manual tests for our GUI. By nature, our exceptions package has classes with minimal logic that are used for handling custom exceptions; this is why the coverage for this is 0%. See Figure 19 below for full test coverage.

A screenshot of the IntelliJ IDEA IDE showing the 'Coverage' tool window. The window displays a tree view of the project structure with columns for 'Element', 'Class, %', 'Method, %', and 'Line, %'. The project 'seng202' is expanded, showing sub-packages like 'team10', 'business', 'exceptions', 'gui', 'io', 'models', and 'repository'. Each package has its respective coverage percentages and counts displayed. For example, 'team10' has 63% class coverage (21/33), 52% method coverage (191/365), and 43% line coverage (900/2090). The 'exceptions' package shows 0% coverage for all metrics. The 'App' class at the bottom shows 0% coverage.

Element ▲	Class, %	Method, %	Line, %
▼ seng202	63% (21/33)	52% (191/365)	43% (900/2090)
▼ team10	63% (21/33)	52% (191/365)	43% (900/2090)
> business	87% (7/8)	58% (69/118)	66% (372/562)
> exceptions	0% (0/1)	0% (0/1)	0% (0/2)
> gui	20% (2/10)	3% (4/108)	1% (9/822)
> io	100% (1/1)	100% (4/4)	89% (61/68)
> models	90% (9/10)	81% (85/104)	66% (237/359)
> repository	100% (2/2)	100% (29/29)	87% (221/254)
App	0% (0/1)	0% (0/1)	0% (0/2)

Figure 19: Screenshot of our overall test coverage

Throughout our development process, we were clear on the need for a robust test strategy. Due to the diversity of functions within our codebase, we adopted a strategy that prioritized functions based on how much they were used and how complex they were. Essential functions, especially those integrated frequently or embedded with crucial logic, were positioned at the top of our testing hierarchy. This approach ensured that the most critical parts of our application were thoroughly tested and tested properly.

For behaviour-driven development, we integrated Cucumber into our testing. Since it was our first-time using Cucumber, we had to come up with a framework that would work well for our project. We eventually settled on grouping them by use cases. This decision was due to our aim to mirror real-world application uses, ensuring that tests not only validate individual components but also the actual user journey. By aligning our Cucumber tests with distinct use cases, we were able to achieve both clarity in our testing approach and assurance key workflows in our app worked well.

8. Current Product Version

8.1 SafeTrip Version 1.0 – Deliverable 2 Snapshot

In version 1.0 of our product SafeTrip, three main aspects of our product delivered were data visualization and process, the GUI design, and routing. Regarding our use cases, this version fulfilled UC.1 Map view, UC.2 View Crash Data, UC.2.1 Filter Crash Data, UC.3 Route Planner, and UC.5 Save Recent Journeys/Locations. The product was also able to fulfil UC.6 Import Data but this was implemented in a more automatic nature during initialisation rather than an enabled functionality for the user to import data. We were not able to implement UC3.1 Colour Coded Route Options and UC.4 Export Data as we did not see these functionalities as being a top priority for the product to be a minimally viable compared to the use cases we did implement.

8.2 SafeTrip Version 2.0 – Finalised Product for Deliverable 3

In version 2.0 and our submission for Deliverable 3, we were successful in implementing all use cases we set out as a team. One use case from the beginning of our project was dismissed and that was UC.4 Export Data. The discussion on why the team decided this use case was not applicable to our stakeholders can be found in Section 2.2.1 regarding our use cases.

Our final product can be summarised by the following three aspects we are most proud of.

8.2.1 Data Storage and Manipulation

Our app uses an SQLite database and singleton classes to store the data for our product. The team has used the SQLite database to keep data that needs to be persistent and the singleton classes for data that did not need to be persistent but easily accessible.

The app on initialisation checks if a database has been populated with any crashes and imports these crashes if not. The initialisation shows all New Zealand and loading all these crashes on app opening. However, the default view has no crash visualization.

The app can manipulate the data used in all the aspects of the app with filtering by various filters as detailed below.

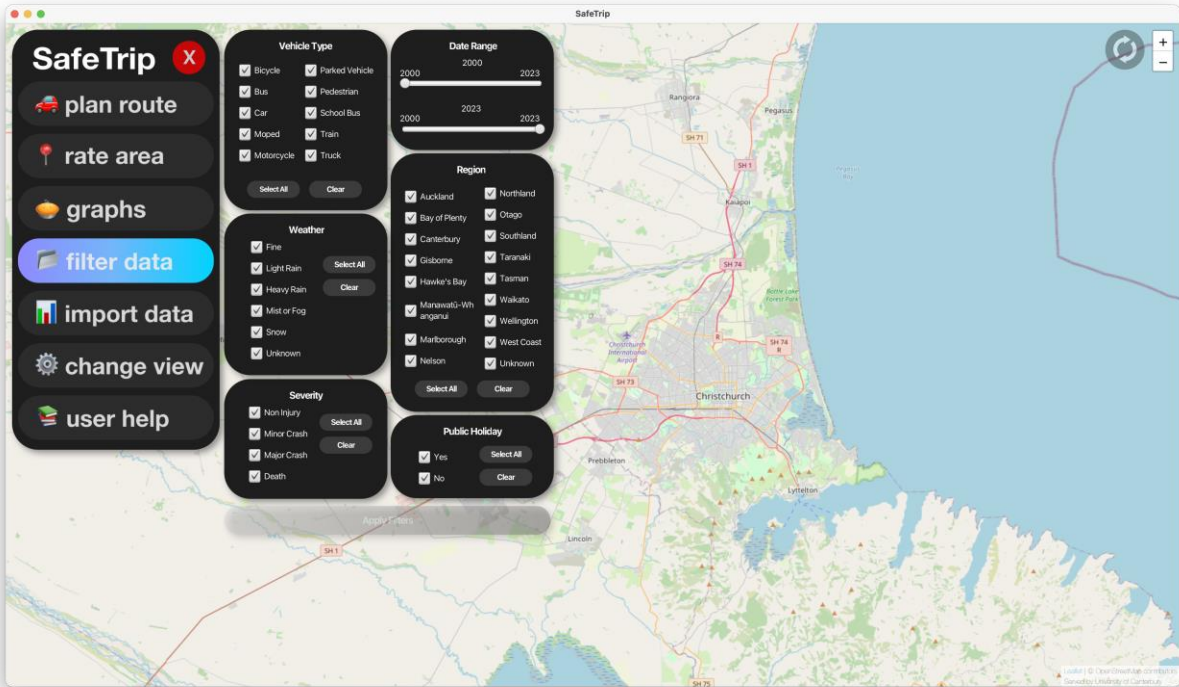


Figure 20: Main screen display

The app can import a validly formatted .csv file and reset the database.

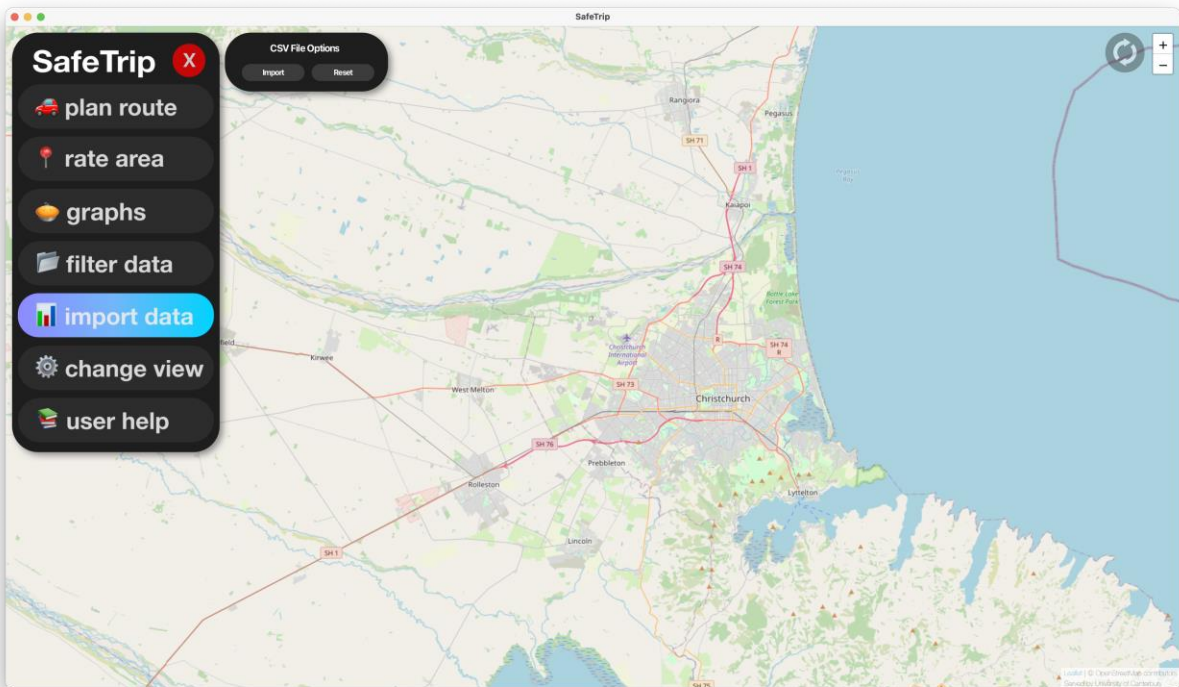


Figure 21: Import data display

This has all been done with efficiency in mind and the presentation and manipulation of data has been optimised to the best of the team's abilities.

8.2.2 Road User Routing

With road users as our largest and most frequent targeted user of our app, routing was a feature we made sure had lots of sub-functionalities within.

Our app can produce up to three routes when generating from a start and end location.

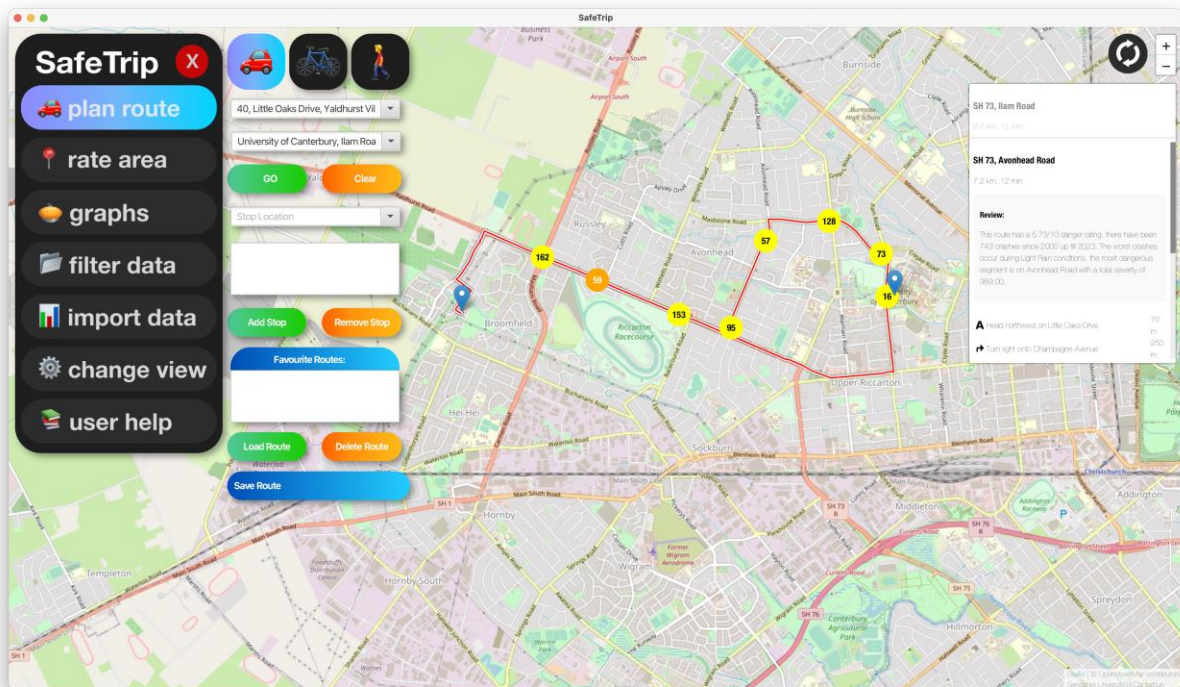


Figure 22: Routing menu display

After this route generation, the route can add and remove stops.

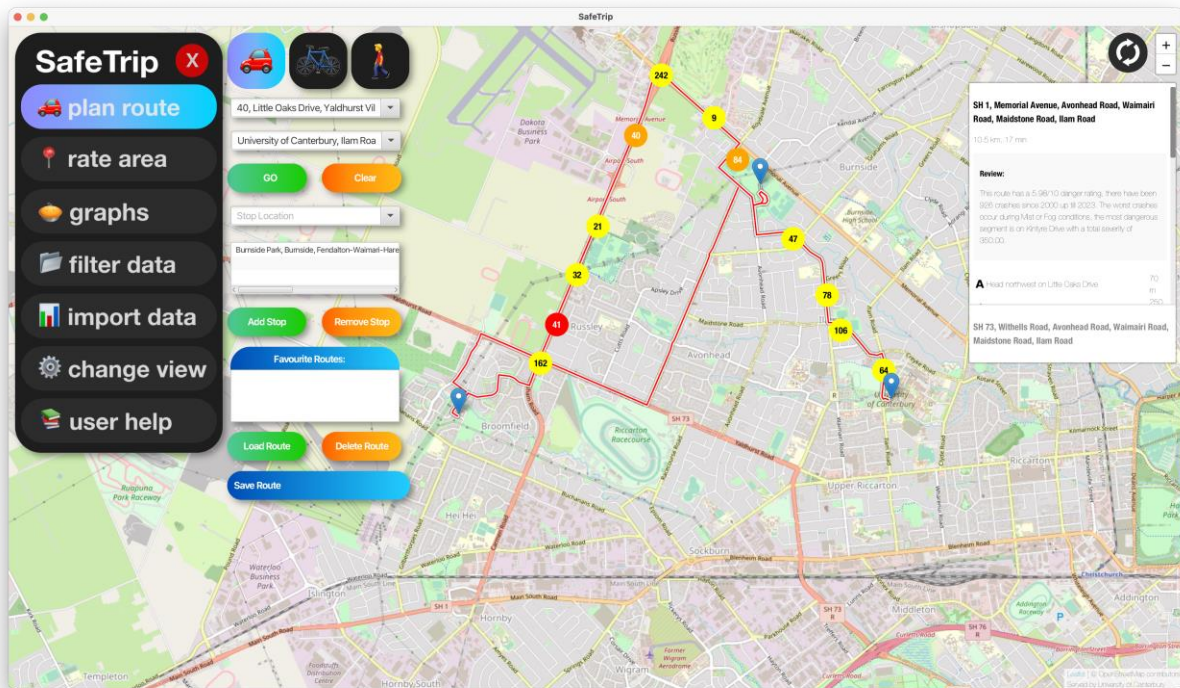


Figure 23: Route review

These routes produce a review embedded within our route information for ease of access to information.

These routes can be saved and deleted as “Favourites” with naming these routes.

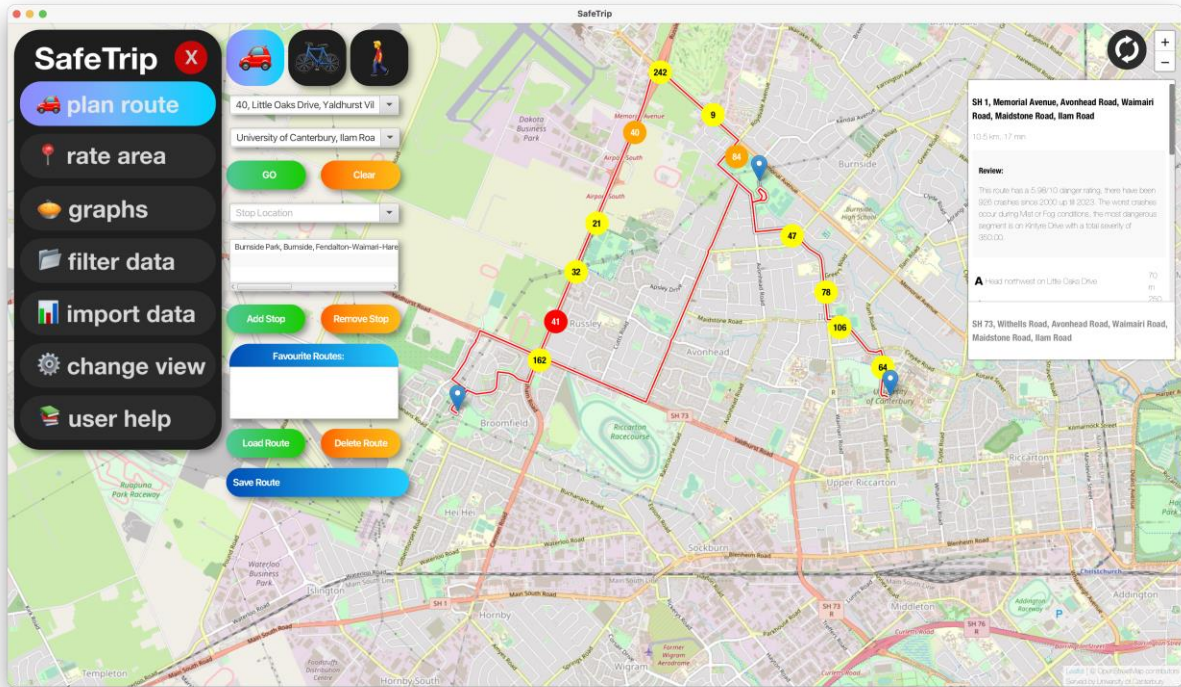


Figure 24: Saving favourites

8.2.3 Analysis of Data

To cater for our data analyst actor, we implemented different ways of analysing the crash data for the user to easily digest this information.

The app can present crashes as crash location markers or as a heatmap. Crash location markers has information regarding each specific crash.



Figure 25: Changing view

The user can graph different columns of the database as a pie chart.

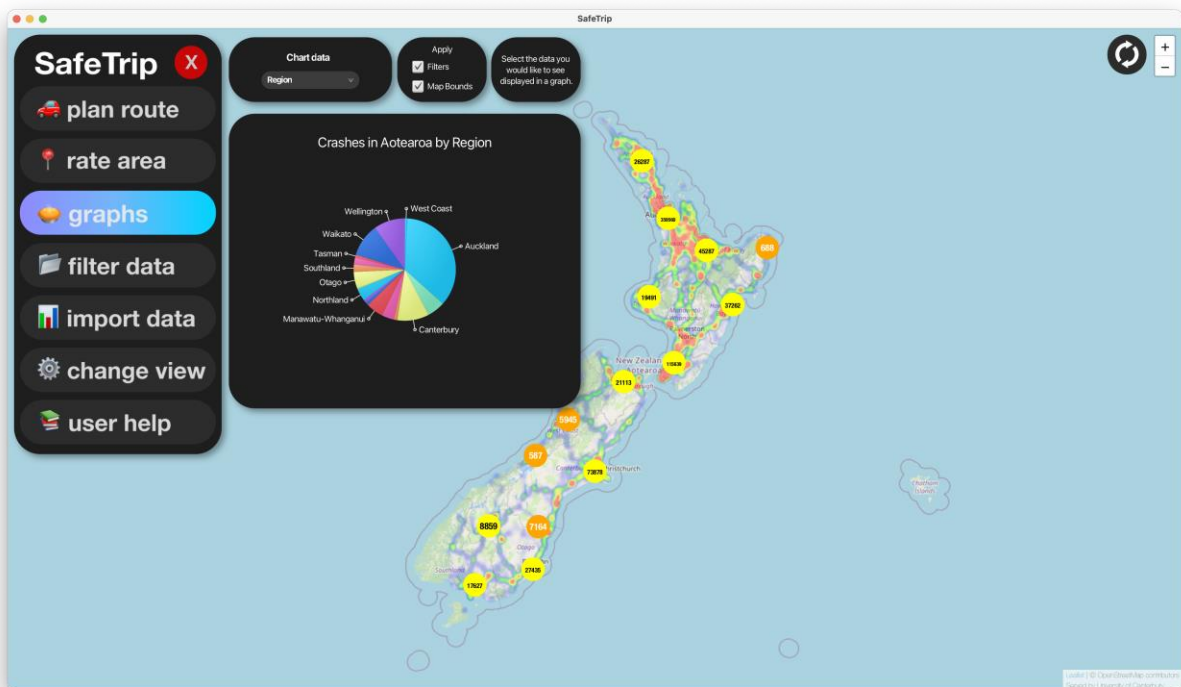


Figure 2616: Pie chart screen

The user can rate areas bounded by a rectangle or circle. The feature can also take in an address and take the user to the zoomed in location for easier area drawing.

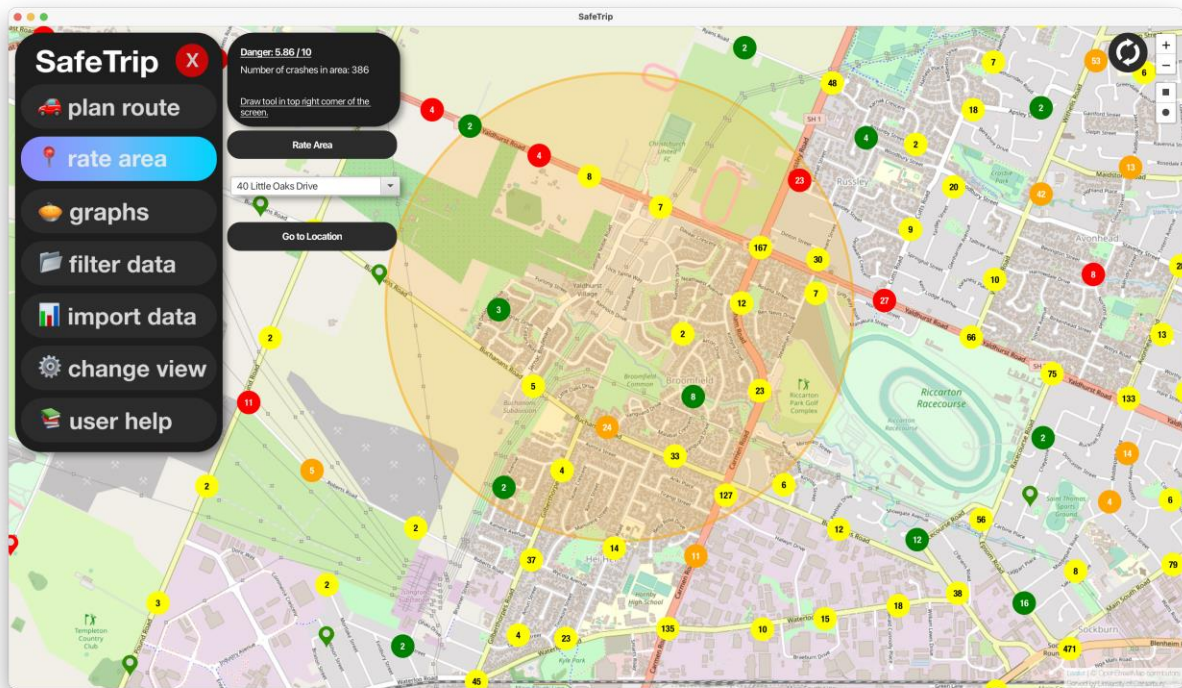


Figure 27: Rating area

9. Risk Assessment

The following risk assessment in Table 10 is based on the risk table provided in the lecture notes (see Appendix C). Generic risks are defined as risks that happen that are not specific to SafeTrip itself, but rather the team and code practices in general. Product specific risks are as described, that directly relate to how the user experiences the app.

Impact and Likelihood are measured as Low: 1, Medium: 2, High: 3. These values are assigned for the purpose of calculating the exposure such that both are weighted the same.

Exposure is calculated as Impact * Likelihood.

9.1 Generic Risks

Table 1010: Generic Risks Assessment

ID	Description	Impact	Likelihood	Exposure	Responsibility Who?	Consequences What happens when risk becomes problem	Prevention Reducing risk chance	Treated Impact	Treated Likelihood	Residual Exposure	Mitigation Reducing impact after it has occurred
GR. 1	Team conflicts / escalated disagreement	3	3	9	All team members	Unhappy work environment, lack of cooperative teamwork, low spirits and drive to succeed, slows progress	Maintain respect for all team members, facilitate discussions and welcome different <i>constructive</i> opinions/ questions, don't take it personally, be a good listener, ask teaching team to mediate if need be.	2	2	4	Escalate to teaching team if need be, call a team meeting and have a mediator involved if everyone is okay with that, ensure open communication and don't be personal or take things personally.
GR. 2	Unreadable code	3	2	6	Relevant team members, all team members	Unable to add/ improve code until the original coder clarifies it,	Create a code conventions policy, maintain open channels to ask questions rather than save it for the	3	1	3	Have the relevant team member clean up their code and comment as appropriate, have another member sanity check for

						stops/slows progress	next meeting, encourage commenting				understanding and clarity, refer to code conventions policy and continue communication.
GR. 3	Poor time management	3	2	6	Relevant member, all team members	Not completing tasks by the deadline, stressing everyone out, staying up all night	Set mini deadlines for small milestones/inch pebbles, make sure everyone knows what task they're doing, communication is key if a team member's situation changes	2	1	2	Place SENG202 as a higher priority than before, ask other team members for help rather than trying to do everything while stressed, apply for special consideration <i>only</i> if applicable under certain circumstances.
GR. 4	Unfamiliar APIs	2	3	6	All team members, relevant team member	Not knowing how to use the API to integrate with our app, using it in the wrong way, taking up time that could be better used in other ways	Making sure most if not all team members that are familiar with the usage of the API, do research in own time, discuss questions as a team	1	2	2	Familiarise yourself further with the API, ask team members or the teaching staff to explain, use further resources on the web.
GR. 5	Miscommunication with teaching team	3	2	6	Team, teaching team	Going the wrong direction as a team, answering the wrong question or confusing ourselves, working on unnecessary things, wasting time	Ask clarifying questions if you're ever unsure of anything, email teachers rather than talking to them so that there's a record we can look back on to remind ourselves	2	1	2	Resolve the miscommunication ASAP by asking the teaching team and the rest of the team for clarification, action the change and stay aware of other possible chances for miscommunication

GR. 6	Unfamiliar development tools e.g., IntelliJ	2	2	4	Relevant team members, whole team	Lack of confidence to code, more time needed to familiarise oneself with the tools	Experiment with the tools beforehand, watch explanation videos/demos and ask teammates for help if needed	2	1	2	Reach out to team members and/or the teaching team for help and resources, practice and familiarise yourself with the tools, do further research
GR. 7	Uneven distribution of workload e.g., due to various coding levels	2	2	4	All team members	Relying only on one person, while others don't contribute as much, a team member may feel useless while another feels overworked, feelings of unfairness negatively influence team dynamics	Discuss as a team when assigning roles, make sure everyone knows their abilities. Set aside time to peer program if necessary, use organizational tools like Trello, refer to meeting minutes role allocation	2	1	2	Communicate with team, make plans for the next allocation of team tasks. Continue to use Trello, set aside time for collaborative working / peer programming.
GR. 8	Unrealistic work expectations/ plan	2	2	4	All team members	Over/underestimating and doing worse than we intended	Communication with team, continuous reflection on progress and capacity to work, updating Trello	2	1	2	Work as a team and adjust accordingly, set clearer/SMARTer goals for the next plan, adjust Trello board as needed.
GR. 9	COVID/ sickness	2	2	4	The sick team member, other team members	Doesn't communicate sickness, loss of manpower, more work for other team members	Mostly uncontrollable, but try to remain healthy and don't overwork yourself, reach out to team for support, team members to be supportive	1	2	2	Everyone else on the team to collaborate and take up the sick member's workload, support each other and stay in contact with the sick person to check in.
GR. 10	Product disagrees with	2	2	4	Team members, stakeholders	Useless or near useless app,	Communicate with stakeholders, do pilot testing	2	1	2	Take feedback from test users to improve app, or

	stakeholder expectations					disappointed stakeholders, team feels like time is wasted	with users as the app is being developed, ask people for feedback continuously including the teaching team.				make sure stakeholders know what our app is about. Ask for constructive criticism from teaching team and each other, refer to personas created.
GR. 11	Copyright infringement	2	1	2	All team members	Not having the rights to use certain information /data, being penalised because of that	Make sure you know where all the files are coming from e.g., open-source code, copyright-free music, cite as appropriate	1	1	1	Replace the infringing material with open source/copyright-free equivalent, ensure team is aware of this issue.
GR. 12	Unexpected personal issues	2	1	2	Relevant team members, whole team	One less team member to work on the project, tensions or stress may rise for other team members	Agree as team that unexpected things happen and it's okay to take time off, the rest of the team will take on their share and work together to support each other. Be understanding and willing to help.	1	1	1	Everyone else on the team to work together and take up the affected team member's workload, communicate as per usual.
GR. 13	OneDrive access lost	2	1	2	All team members	Lose minutes, deliverables and other shared files that are important for the planning and development of the project, may impede progress since organization record is lost	Have backup files on each device, send files to group chats e.g., Slack, continuously back up	1	1	1	Restore files from the backup, refer to history and relevant group chats if need be, ask team members if they have any memories or things to add to the restored files.

9.2 Product Specific Risks

Table 1111: Product Risks Assessment

ID	Description	Impact	Likelihood	Exposure	Responsibility Who?	Consequences What happens when risk becomes problem	Prevention Reducing risk chance	Treated Impact	Treated Likelihood	Residual Exposure	Mitigation Reducing impact after it has occurred
PR.1	Stakeholder misuse	3	2	6	Team, stakeholders	Unfairly impacting things like house prices e.g., real estate agents with ulterior motives	Don't implement function for self-logged reports for the user	2	1	2	Know who the stakeholders are and their reasons for using the app – they shouldn't be able to influence the app's data anyway so there's no extra data to doublecheck or validate in the first place.
PR.2	Unintended consequence of "safe" classified roads	3	2	6	Team members, relevant stakeholders	Unsafe driving (users let guard down on 'safer' roads), impacts their actions when it otherwise wouldn't.	Insert warning or info box that there will always be hazards and risk, encourage road safety, don't use numerical scores, link to other road safety sources	2	1	2	Add further warnings / safety features to the app and include other information, for example a speed limit in the area/route

10. Project Plan

10.1 Trello

Here is our finished trello board. With two weeks to go until the deliverable submission, we sat down and talked through all the features we wanted to implement still. We decided that some features would be impossible to implement with the timeframe we had, so we created a new category for them. We completed all our relevant tasks on time. At every meeting, we constantly updated the Trello by coming up with new ideas, evaluating those ideas and assigning them to individual people. This meant that group member knew what they had to work on, and gave them the freedom to present their ideas to the group.

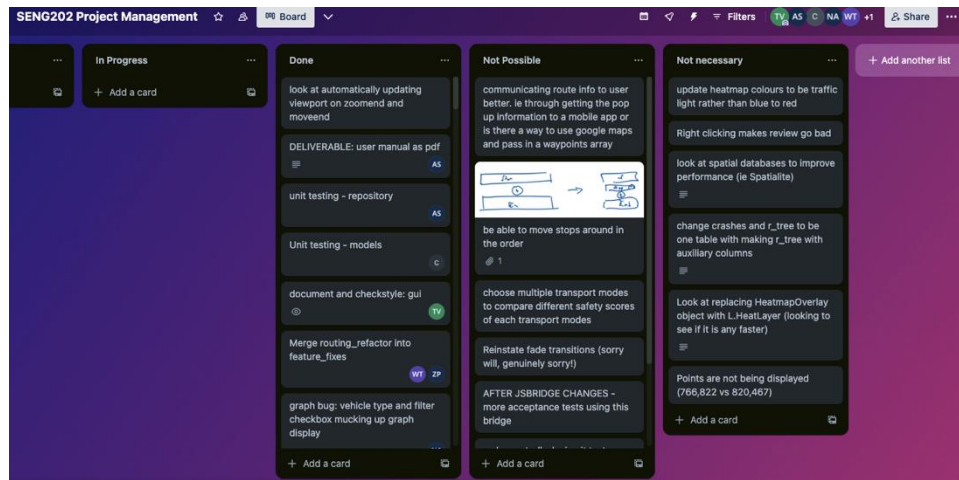


Figure 1228: Trello board

10.2 Major Milestones

Table 10 and Table 11 show the major and minor milestones respectively, which will be subject to change as the project progresses. Major milestones indicate about a third of the work required for a deliverable. Minor milestones indicate individual classes being complete, sections in the project plan etc.

Table 1213: Major Milestones

Milestone	Date	Description	Related tasks
M1	31/07/2023	Requirements document, have to around 80% completion	<ul style="list-style-type: none"> Project plan Risk assessment Stakeholders and requirements Business and system context
M2	04/08/2023	Prototyping with figma and code plan	<ul style="list-style-type: none"> GUI Prototypes Deployment model Detailed UML class diagram Acceptance tests

M3	07/08/2023 5pm	Edit and submit Deliverable 1	<ul style="list-style-type: none"> • Finish off all sections • Edit and proofread for submission
M4	9/08/23	Present Deliverable 1	<ul style="list-style-type: none"> • Completion of Deliverable 1
M5	23/08/2023	Have the first minimum viable product complete	<ul style="list-style-type: none"> - Rough controller class complete - Homepage window and Map window functional - Hamburger window also complete - Get the data class to about 50% completion
M6	19/09/2023	Complete sections excluding the graphing display	<ul style="list-style-type: none"> • Complete the filter data class and data class • Complete the Heatmap, Route and Point classes. • Complete the heatmap overlay and crash window overlay
M7	20/09/2023 5pm	Edit and submit Deliverable 2	<ul style="list-style-type: none"> • Finish off all sections • Edit and proofread for submission
M8	22/09/23	Present Deliverable 2	<ul style="list-style-type: none"> • Completion of Deliverable 2
M9	09/10/2023	Have the final product complete	<ul style="list-style-type: none"> • Complete both the about page window and help window • Complete the graph display window • Complete the builder pattern graph creator • Complete the favourite functionality
M10	16/10/2023 5pm	Edit and submit Deliverable 3	<ul style="list-style-type: none"> • Finish off all sections • Edit and proofread for submission
M11	18/10/23	Present Deliverable 3	<ul style="list-style-type: none"> • Completion of deliverable 3

10.3 Minor Milestones

Table 1413: Minor Milestones

Milestone	Date	Description	Related tasks
M1.1	31/07	Project plan.	Complete the project plan.

M1.2	31/07	Risk assessment.	Complete the risk assessment.
M1.3	31/07	Key driver analysis.	Complete a key driver analysis.
M1.4	31/07	Stakeholders & requirements.	Find all the stakeholders and requirements.
M1.5	31/07	Business & system context.	Complete the business and system context.
M2.1	04/08	GUI prototypes.	Complete the GUI prototypes.
M2.2	04/08	Acceptance tests.	Complete the acceptance tests.
M2.3	04/08	Deployment model.	Complete the deployment model.
M2.4	04/08	Detailed UML class diagram.	Complete a full UML class diagram.
M3.1	07/08	Proofread/edit deliverable 1.	Ensure formatting, grammar, and structure is all correct.
M3.2	09/08	Presentation for deliverable 1.	PowerPoint slideshow must be complete, and sections divided.
M4.1	13/08	Setup all software such as git and Gradle for version control.	Setup git, Gradle the files and any other software.
M4.2	23/08	Work on controller class and test.	30% complete controller class with all methods tested.
M4.3	23/08	Homepage and map GUI completed.	Create the GUI and test the code that connects them.
M4.4	23/08	Hamburger GUI complete.	Create the hamburger GUI and test the code
M4.5	23/08	Data class complete	Get the functionality of importing the data and processing it complete. Test every method.
M5.1	19/09	Filter data class and data class complete.	Test and complete all methods within the filter data and data classes.
M5.2	19/09	Complete the display related classes.	Test and complete the Heatmap, Route and Point classes.

M5.3	19/09	Complete the display related GUI	The heatmap overlay and the crash overlay must be complete.
M6.1	20/09	Presentation for deliverable 2.	Complete PowerPoint slideshow and organise roles.
M7.1	09/10	Complete the front-end windows.	Complete the about and help pages.
M7.2	09/10	Complete the graphing GUI	Complete the graph display window class
M7.3	09/10	Complete the builder pattern related graphing classes.	Complete GraphCreator, GraphBuilder, ConcreteGraphBuilder and Graph.
M7.4	09/10	Complete the favourite functionality	Add the ability to favourite routes and keep this in permanent storage.
M8.1	16/10	Presentation for deliverable 3	Setup PowerPoint slideshow for deliverable 3 and decide on who speaks on wha

10.4 Development Strategy

10.4.1 Feature Branching

Definition: Feature branching is a development methodology where a new branch is created in the version control system for every new feature or bug fix. This ensures that the main branch or dev branch remains stable and usable at all times.

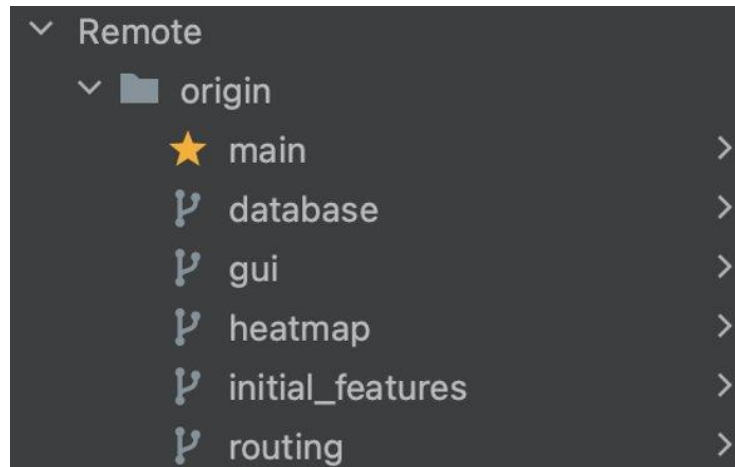


Figure 1529: Screenshot of branching inside IntelliJ

Given the size of our team, feature branching has been particularly advantageous. With multiple developers working simultaneously and sometimes on the same features, it's essential to have isolated environments to prevent overlap, potential code conflicts, and ensure smoother integration. Feature branching has provided each team member with their own workspace, allowing for side-by-side development without hindering others. This has been a great help with the speed of which we implement new features.

11. Lessons Learned

Angelica

Throughout this project I learnt of the importance of time management and planning. I had never used tools like Trello before and found it really helpful when the team implemented it. I now also use Trello in my personal life. Reflecting on the semester, I can definitely think of moments when I should have managed my time better. I spent too much time stressing over little things in the app and not actually implementing major functionality. In the future, I now know that it's better for me to ask questions even at the risk of appearing less smart.

A technical lesson I learnt was the importance of Javadoc. There were many times throughout the project when I found that I couldn't understand other people's code and found that it took me too long to trace back. This then led me to realise that I should also document my own code properly and think about how other people might see my code with zero context. This helped to ensure that everyone had an idea of the codebase and what everything was for.

I'd say the most valuable lesson I learnt from this project was using empathy and communication. This helped our teamwork and made sure that no one felt uncomfortable raising any questions or issues. This also contributed to a friendly team environment where people cared about the product, and all wanted to do our best to help each other succeed. Overall, I really appreciated the openness we had with each other and how I was able to 'rubber ducky' off various team members. There were times when the course was unable to be my main priority, and being honest about this with the team enabled us to not fall behind and make sure everyone was on the same page.

Chris

My first key lesson learned is that I should perform more frequent commits for more single-purpose functionality. During development, I noticed that my commits were significantly larger and much more multipurpose than my teammates. I realised that this was very bad as it made it very difficult to rollback and find what specific parts caused the branch to no longer function. Another issue is that it is much more difficult to look back on what I have done for clockify and testing.

My second key lesson learned is that it is much better to get feedback from tutors before the submission rather than through the submission. In the future, I need to make sure to dedicate some period of time before the submission to go over the submission with a tutor. The issue we faced was that because the submission was on Monday we couldn't get help from tutors over the weekend, hence it is important to properly schedule a session.

My final key lesson learned is to keep much more consistent during the entire project. I regret taking breaks after the submissions of the deliverables as it put me a whole week behind schedule. In the future I would set weekly time goals, such as needing to at least do 10 hours a week rather than object related goals, as they can be too easy to procrastinate.

Neil

My biggest takeaways from this course are that the team is only as good as its communication, consistent contributions help with understanding, and that my most efficient debugging strategy is the “rubber ducky”.

Communication was strong from all team members, and I felt that this is what made our project fun to work on. This also helped make sure everyone was clued in on what was happening and what was being worked on so that there was no overlap on the features being worked on. There were certainly times where I or another member were less communicative and the effect this showed on our teamwork was immense. This certainly taught me that communication is a priority for team projects.

Contributing consistently helped with keeping up with understanding of the project in both a low level in terms of programming and high level with how our product was shaping up to be. I made sure to read people’s messages on the group chat and on Trello to make sure I had a good grasp on everyone else’s headspace. Making sure I was consistently on top of the project and how I was contributing helped with not falling behind and having to be caught up by my teammates.

I also found that my favourite debugging strategy was talking to a “rubber ducky”. Working on such a complex and large-scale project that I have never been exposed to made running into bugs very frequent. I found that as I talked the problems out loud to my drink bottle or a team member that was not fully paying attention that this helped me decipher what the core problem was.

Todd

One thing I took away from this course, is to constantly ask for help from the tutors and my fellow group members. Normally when I code, I like to try do everything by myself, without asking for help, and since this was my first major group project, I attempted to do the same without realising how hard it would be. I learnt to depend on others in my group and ask them for help whenever I get stuck. Over time, I began to ask for help more and more whenever I needed it. I found it is really important to ask others for help, and not get bogged down on the same task for many hours.

Another thing I took away was the experience of working together as a team. Like I said above, I haven’t had experience being a part of a group project of this magnitude before. Communication between us was very effective, and we all got on very well, which made the course more enjoyable. I have learnt a lot about empathetic listening, by letting others speak and voicing my opinions when it matters. The time commitment was massive, but having a highly talkative team meant that the meetings and the peer programming was a lot more fun.

The final takeaway from the course for me was time management. As shown by my clockify, I started really consistent, but during the midway point of the course, I didn’t put in as many hours as I should have. This led to a very hectic last couple of weeks. I learnt how important it was to set small goals, such as ‘I want to work for 10 hours this week’ or ‘I want to implement this feature today’. By setting small goals, I was able to greatly improve my productivity and get more tasks done. This is a big takeaway that I will utilize in future projects.

Will

During the development of our project, I often found myself stuck on coding tasks. At the start I would often spend many hours trying to fix bugs/errors but closer to the end of the project I started asking for help and I found that often bugs that took me hours to fix took a couple minutes with another pair of eyes spotting stuff I had previously overlooked. I believe asking for help increase the efficiency at which I did lots of tasks mainly coding ones and this is something I will continue to do in future.

My second key lesson that I learned is to do frequent and small commits when using git with a codebase. At the start of development, when working on features we would use feature branching which was useful for each member working without clashing with others. I often found myself working for ages and getting to a point where I would break my branch, doing lots of small commits was very useful as it allowed me to roll back to working versions without losing lots of my work.

The final lesson that I learnt was importance of good time management, at the start of this project I was unfamiliar with starting work early on a project as opposed to the week before the due date. I found it lots easier to manage my time by committing a couple times a week where I would work on the project, meaning that we all made good consistent progress. This helped as it meant that when it was time to submit, we did not have too much to do and It was much less stressful than times I have had in the past, having consistent time management is definitely something I will continue to do.

Zipporah

One major lesson I've learned is the importance of having strong communication within the team and making sure everyone is always on the same page. As a team we were all frequent communicators, however our methods have improved greatly as the project duration progressed. One example of this can be seen in our Trello board: we initially set it up at the beginning of our project but then barely used it for the entire first deliverable, and much of the second. Closer to the project due date, we began to utilise our Trello much more - adding all tasks that are yet to be done into a list, where we could each see everything that needs doing immediately and allocate ourselves to tasks accordingly. I noticed this made a considerable improvement to our team collaboration as we could each see the tasks being worked on at any given time, in turn reducing the frequency of making conflicting and/or redundant code.

Another lesson I've learnt is the necessity of pacing ourselves when it comes to completing our planned tasks. Consistency with commits was a big issue I personally faced, as I noticed I made far fewer commits to the code repository at the start of the project compared with the end. Because of this, I was not on track to completing my designated 150 hours of work and found myself having to spend a large amount of time catching up in the final two weeks before submission. This led to an unnecessary amount of stress before the final submission, not just for me but the entire team. It has been a hard lesson learnt and will become a priority for me to avoid this situation happening in the future, through more frequent commits and thorough project planning.

Overall, this project has been an unparalleled learning experience for me, and I am grateful for the valuable experience I will take with me for future pursuits.

12. References

Apple NZ. (2023, Aug 7). *Top iPhone Navigation Apps on the App Store*. Retrieved from App Store:
<https://apps.apple.com/nz/charts/iphone/navigation-apps/6010>

Dribbble. (2023, Aug 7). Retrieved from Dribbble: <https://dribbble.com/search/route-planner>

PhoneMe.Money Ltd. (2019, May 25). *Crash NZ (1.21) [Mobile application software]*. Retrieved from Google Play:
https://play.google.com/store/apps/details?id=nz.co.omega.omegacrashnz&hl=en_NZ&gl=US

Stakeholder Analysis. (2023, July 28). Retrieved from ProductPlan:
<https://www.productplan.com/glossary/stakeholder-analysis/>

Struyk, T. (2023, June 7). *The Factors of a "Good" Location*. Retrieved from Investopedia:
<https://www.investopedia.com/financial-edge/0410/the-5-factors-of-a-good-location.aspx>

Waka, T. M. (2023, July 28). *How - Te karore ā-whānau (Household travel)*. Retrieved from Te Manatū Waka - Ministry of Transport: <https://www.transport.govt.nz/statistics-and-insights/household-travel/how/>

13. Appendix

Appendix A – Sections Worked on by Members

The following is the percentage breakdown of design document contributions by team member.

All team members worked together on all sections in terms of ideation and overall shape of what the section communicates. The following talks about the members who took what was agreed on from team meetings and expanded on those items.

Executive Summary

- Zipporah Price (90%)
- Angelica Silva (10%)
- 1. Business and System Context
 - Zipporah Price (95%)
 - Todd Vermeir (5%)
- 2. Stakeholders and Requirements
 - 2.1 Stakeholders
 - i. Neil Alombro (70%)
 - ii. Todd Vermeir (30%)
 - 2.2 Requirements
 - i. 2.2.1 Functional Requirements
 - 1. Todd Vermeir (50%)
 - 2. Neil Alombro (50%)
 - ii. 2.2.2 Quality Requirements
 - 1. Christopher Wareing (100%)
 - 2.3 Key Driver Analysis
 - i. Neil Alombro (20%)
 - ii. Zipporah Price (16%)
 - iii. Todd Vermeir (16%)
 - iv. Angelica Silva (16%)
 - v. William Thompson (16%)
 - vi. Christopher Wareing (16%)
- 3. Acceptance Tests
 - Neil Alombro (100%)
- 4. GUI Prototypes
 - William Thompson (100%)
- 5. Deployment Model
 - Angelica Silva (40%)
 - Zipporah Price (40%)
 - William Thompson (20%)
- 6. Detailed UML Class Diagram
 - Christopher Wareing (50%)
 - Todd Vermeir (20%)
 - Neil Alombro (30%)

7. Risk Assessment
 - Angelica Silva (100%)
8. Project Plan
 - Christopher Wareing (75%)
 - Todd Vermeir (5%)
 - Neil Alombro (5%)
 - Angelica Silva (5%)
 - Zipporah Price (5%)
 - William Thompson (5%)

Following on with Deliverable 2 and Deliverable 3, we utilised the change log for the ongoing changes on our collaborative contribution. We did not update these percentages but contributions by team members are detailed in that change log.

Appendix B – Excel Sheets

The developers made use of Excel spreadsheets to document functional and quality testing.

- [Team 10 Manual Tests.xlsx](#)
- [Team 10 Quality Tests.xlsx](#)