LookOut Design Document

Final Product / Deliverable 3

Team 7 SENG202

Team members:

Jack McCorkindale	(71415038)
John Elliott	(98040483)
Sam McMillan	(87685388)
Sami Elmadani	(68705375)
Shaylin Simadari	(66259837)

Submission Date:

11/10/2021

Table of contents

1: Business and system context

1.1 Relevant business information

Our application is one that can be used by an everyday consumer. It is designed to provide much of the functionality that basic police-grade data applications would, in a user-friendly and focused manner. The basic functionality provided allows the user to import crime data and analyse and edit it before exporting sections out, based on filtered results.

The key feature and unique selling point of the application is its analysis of the imported data. This is done through a variety of graphs and mapping tools such as local pin maps that allow visual comparisons between two distinct crimes and their relevant differences. Continued development of the application could lead to using heat maps to aid in the visualisation of regions with higher crime rates than others.

The information provided through the app could cover any information that an average user would need about a crime in the area. This would be beneficial to a range of people including home buyers, the elderly, or people looking to rent or flat.

As there is a range of needs for each potential user of the application a login system can be implemented to cater for these needs. For example, in community groups, only certain people should be able to add, remove or edit crimes that are in the system. The user system can also be used to customise some values such as date being in dd/MM/YYYY or MM-dd-YYYY. Therefore the login system would only allow people with permission to do these tasks.

Limitations of this product would be the lack of features it would have in comparison to other products, such as the police data system in place currently. To avoid our application being redundant we will focus on specializing in a user-friendly design that would make our application more practical for non-technical users over the complex police system.

Existing Software Products:

- Auror:
    - (*Auror - The Retail Crime & Loss Prevention Platform*, n.d.)
    - Made for enterprise retailers
    - Very focused on shoplifting and robbery of goods
    - Somewhat related, however, does not record much data required by the average citizen
    - Only available internationally
    - Paid
- policedata.nz:
    - (*policedata.nz*, n.d.)

    - Huge competitor, and highly relevant.
    - Provides all forms of tagging features, filtering, heatmaps, and more
    - Data is verified
    - However: Does not allow anyone to upload data, much criminal activity goes unnoticed. Does not consider levels of security in each area
    - Very difficult to navigate for the average user

Business opportunities & Unique selling points:

- Import police reports that can be integrated into the user database for analysis.
- Add personal reports that are localized to the application and therefore private.
- Filter crimes into categories that allow users to visualize and see relevant information such as robberies in their area.
- User-friendly layout and operation as opposed to existing applications which provide tools requiring knowledge on how to use them.
- A Login system could be implemented in future releases and would provide security to data created by users and application customizability for the user.
- Heatmaps could be implemented in future releases and would provide visualisation of high crime areas with a steady gradient representation.

Anticipated advantages/disadvantages over current products:

- Criminals could capitalise on such software. Knowing areas of unrecorded crimes might alert them about areas with little security. This could then increase the crime rate in the area.
- Lack of mapping features may be an issue for some users who would rather have more complex and detailed visualisations of crimes within a region.
- Being able to only export on a filter result might not be convenient to all users that want a specialized selection of data exported.
- A user-friendly design would make our application more accessible to users that have less technical knowledge.
- Allows a user their own local persistent system for storing data and analysing it rather than an online system like the NZ police application.

Target markets, potential customers:

This application will provide much of the functionality that existing applications, such as the NZ police database, do, but in a more straightforward and intuitive way. It will provide easy methods of retrieving only the relevant data that the target market would want. This target market currently includes people looking to flat or rent, home buyers, and the elderly. Many of the benefits of this product could be shared between each user but some features will be targeted towards some more than the others. This product will use its selling points to focus each target market's needs specifically, and be a much more lightweight method to receive criminal information within local areas.

1.2: System context:

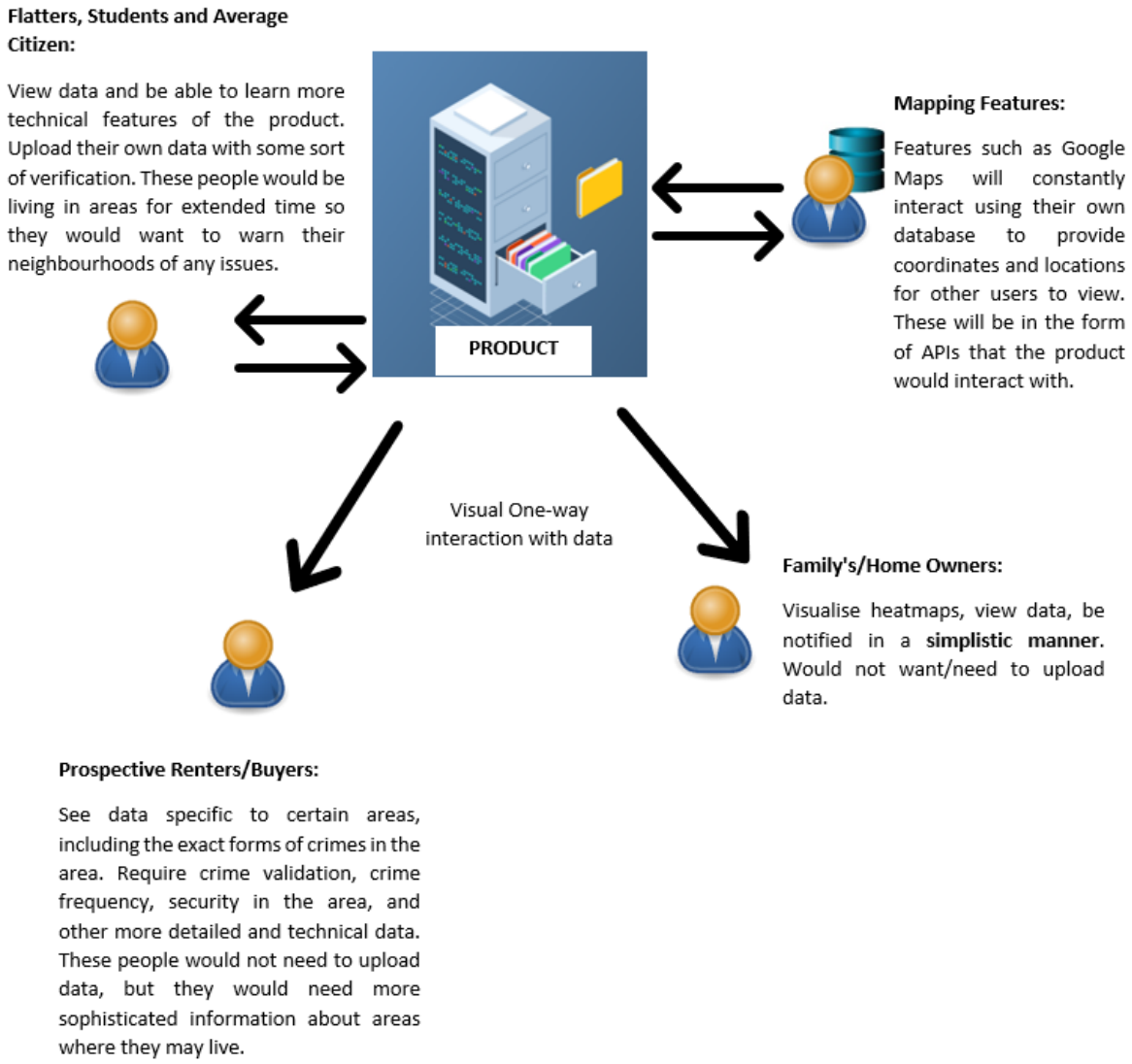

Figure 1: System context diagram of product.

The system context will involve many different users. You will have different stages of access to the data, with Mapping features in Figure 1 at the top. They will use APIs that assist with the data present in the application, such as Google Maps, and will use queries of data to present information visually. The next is the more technical users such as flatters, students and the average citizen.

These will still want to be able to manipulate the existing data on a simple level to be able to inform those around them of dangers in the area. The final group expected is the homeowners and families. They have the same requirements in viewing data, and both do not need to manipulate the data. Yet, the home buyers may want more specific information about each area, while elderly users would want the most simple and comprehensible version of the data to view, hence their access levels may vary.

2: Stakeholders and Requirements

2.1: Stakeholders

During our analysis of our business and system context, we identified three key types of stakeholders that would use the application. These were “Prospective Renters/Buyers”, “Student Flatters” and “Family's/Home Owners” as shown in Table 1. As one of the application's key purposes is in the analysis of crime data in a non-professional context, such as statistics of a specific crime in a simple visualization, these stakeholders are all considered high ranking. This is because most would not have the means to use and understand professional systems for this analysis and those systems provide a far more complex set of tools than would be necessary for a simple user.

As we are stakeholders of our project we also spent time considering what our concerns were and that of the teaching staff. This helped us to identify the quality of the project application we want to achieve and the expectations of the team.

Finally, we identified many low priority stakeholders which may have an interest in our application but are not likely to interact with it as a user such as API providers like Google or the police which already have advanced systems that can accomplish the same analysis

Table 1: Stakeholders with relevant concerns and priorities about the envisioned LookOut application.

ID:	Stakeholder:	Concerns:	Priority:
SH_1	Development Team	To produce a successful product that will help ensure the final application works in the outlined business system. Create a system that will better improve the safety of users through the collection and analysis of crime/incident data. That the application phases are completed in the time frame outlined in the milestone calendar in the project plan section. That the final application results in a good grade for all the team members.	High

		That if the application will be used by members after the initial project time frame other members have agreed and have had their necessary information removed from the project.	
SH_2	Prospective Renters/Buyers	Allows a user to effectively review and analyse a given set of data for a region to be able to make a more informed decision about the safety near a property. Provides information relevant to a prospective Buyer/Renter such as the given number of a specific crime that has occurred in a region. That there is a visual way to interpret the data so that it is user friendly to those less technical.	High
SH_3	Student Flatters	Allows a user to make incident reports that can be shared with others to raise awareness or be kept private for personal reasons. That filling out an incident report is simple and user friendly with concise and relevant errors if the necessary information is missing. They can resume what they were doing previously without having to take the time to repeat everything. That the system is responsive and can quickly return results of a search on specific crimes.	High
SH_4	Family's/Home Owners	Allows a user to effectively review and analyse a given set of data for a region to be able to make a more informed decision about the safety near a school. Can provide information relevant to their needs such as the type of crimes that have occurred in a specific area. For example, if they have kids they might be interested in areas surrounding a school. Allows users to visualize information about key areas in their region like stores and homes to better understand risks in their area.	Medium

		That the system is responsive and can quickly return results of a search on specific crimes.	
SH_5	Teaching Staff	To help in the development of an application that will allow us to complete the project successfully. That the team members are respective and reasonable to other members and teams during the project. That the resulting application is of a good grade that positively reflects the course.	Medium
SH_6	Google	That the application is responsibly using their services. That if their API systems are changed or offline that there is a system in place so the application is still functional and/or can be updated. That users are aware of the API systems teams of services and what it means for their potential data.	Medium
SH_7	University of Canterbury	That the application adheres to the universities code of conduct. That the team members also adhere to the same code of conduct.	Low
SH_8	Criminals	No private information that is not publicly or should not be publicly available is released through the application. That the use of the application will help to decrease the likelihood of potential criminal acts in flatting areas.	Low
SH_9	Police	That the system allows for a quick way to review simple sets of crime data. That information gathered and/or used has been consented to and/or is publicly available.	Low

2.2: Requirements

Use cases

Diagram

The requirements of the application have been identified and presented in Figure 2. These have been assigned a number from 1 to 15 relating to the expected implementation order of the use cases in relation to the minimum viable product of the application.

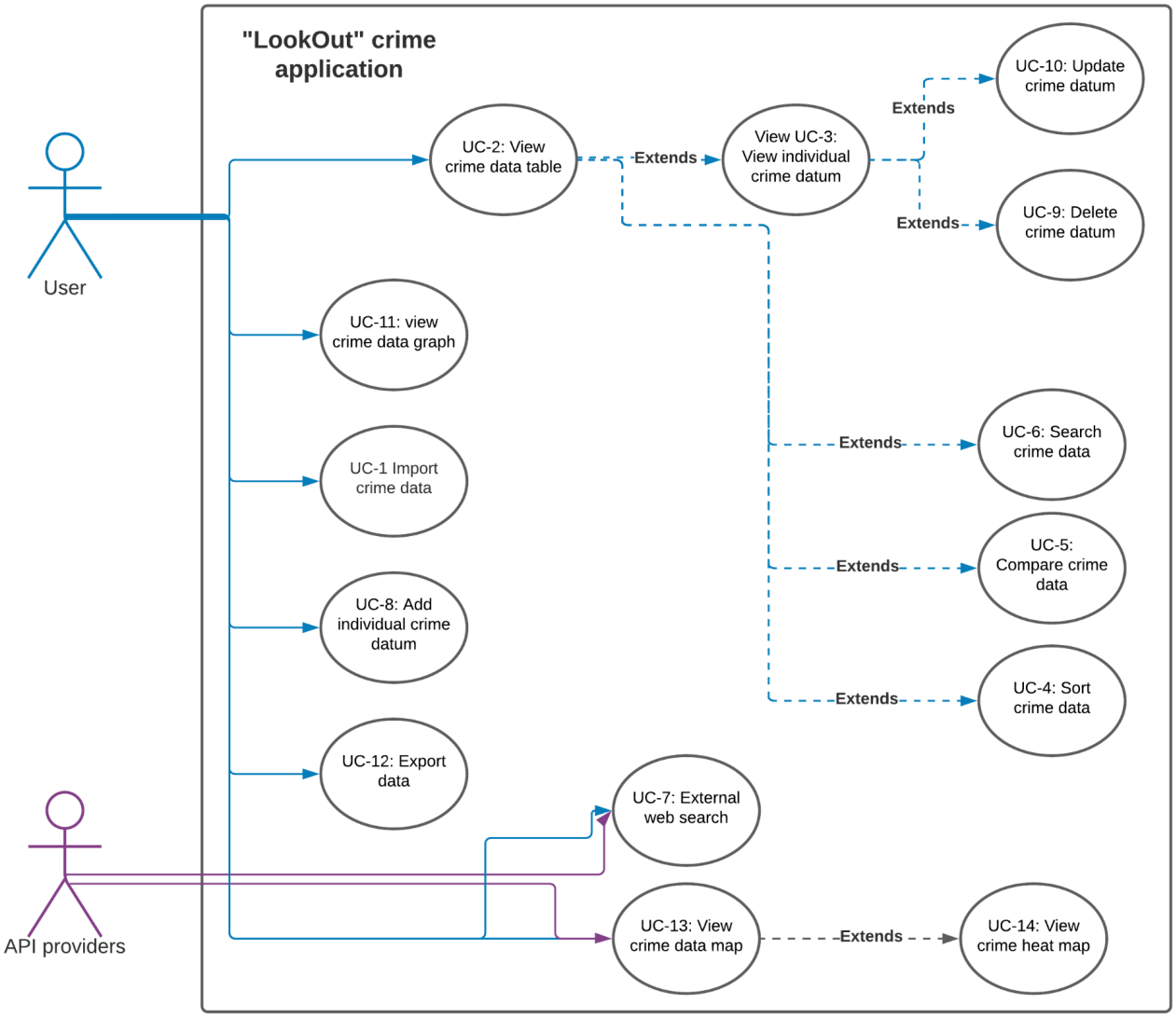

Figure 2: The use case's of the LookOut application represented by a use case diagram.

Actors

There are two actors present in Figure 2, the User and the API provider. The user has the most use cases due to them being the primary user of the system. This is in contrast to the API providers which have limited access due to their supporting role.

User – ‘User’ includes the target audience for the crime data application. user. ‘User’ includes prospective renters/buyers, student flatters, families/homeowners. ‘User’ has access to view and analyse the data.

API providers - Includes the Google Maps API for providing the map API for heat maps and crime visualization. Also, a Google search API for as an external search feature.

Textual Description

Table 2: The textual description of each of the use cases outlined in Figure 2 ordered in implementation order.

ID:	UC-1
Name:	Import crime data.
Actor(s)	User
Priority	High
Status	Implemented
Description:	Allows the user to import crime data to the application from a CSV file.
Precondition:	The user is in the add crime tab in the application. The user has a CSV file saved on their computer.
Basic flow:	Basic Flow: 1) User selects “import button” 1) An import file window pops up. User selects the file from a directory popup window. 2) User clicks the “import”: button.

	3) The application will perform a check on the data to ensure it follows the correct format and is not duplicate data.
Alternative flow:	Add data to an existing table: 4) User selects a pre-existing table to add the data to by clicking a button on the screen, appending the data to the already existing table in the database.
Postcondition:	Data from the CSV file is now saved in the application database in the pre-existing table. The user is returned to the table view. The data is now accessible from within the application.
Alternative flow:	Create a new table and add the data to it. 4) The user selects a to create a new table. 5) The user types a string into a text input box to create the name of the table and then clicks create. The application checks that no other table exists with that name. The data is added to the new table. The data is now accessible from within the application.
Postcondition:	Data from the CSV file is now saved in the application in a new table. The user is returned to the add crime data window. The new data is now accessible from within the table view
Exceptional flow:	CSV is not accepted by the application: 4) Data does not pass the application's checks and is not added. The new data is not accessible from within the table view.
Postcondition:	The user is returned to the start of the add crime data window and a text prompt tells the user that the data was incorrect, faulty, or duplicate. The new data is not accessible from within the table view.

ID:	UC-2
Name:	View Crime data table.
Actor(s)	User
Priority	High
Status	Implemented
Description:	Allows the user to view raw crime data in a table. This use case acts as the central point in the application, many use cases branch from the view crime data table.
Precondition:	The user is in the view data table window. The application already has some crime data saved in the database.

Basic flow:	1) Users can see a large table of crime data.
Postcondition:	The user is in the view crime data table window.

ID:	UC-3
Name:	View individual crime datum
Actor(s):	User
Priority:	High
Status:	Implemented
Description:	Allows the user to see all details of a crime datum
Precondition:	The user is in the view data table window. The application already has some crime data saved in the database
Basic flow:	1) User double clicks on an individual crime which brings up an individual crime data view screen in the central pane. The window has all the details about the crime.
Postcondition:	The user is in the individual crime view screen.

ID	UC-4
Name:	Sort crime data
Actor(s)	User
Priority:	Medium
Status:	Partially implemented (only works in the current page)
Description:	Allows the user to sort/filter the data in the crime data window.
Precondition:	The user is in the view data table window. The application already has some crime data saved in the database.
Basic Flow:	1) User clicks on the “sort” button in the menu on the left-hand side of the application, bringing up a screen of drop-down and input boxes. 2) The user can now select how they would like to sort the data by selecting from time, location, crime type, etc. The user clicks a sort button. 3) The crime data table is now sorted by the specified parameters the user selected.
Postcondition:	User in the view data table window with the data sorted accordingly.

ID:	UC-5

Name:	Compare Crime data
Actor(s)	User
Priority:	Medium-High
Status:	Implemented
Description:	Allows the user to compare two or more crimes.
Precondition:	The user is in the view data table window. The application already has some crime data saved in the database.
Basic flow:	1) Users select on the left-hand menu to compare crime. 2) The general menu switches to a compare menu with two input boxes 3) The user types in the primary key of the crimes they would like to compare, then the user clicks compare. 4) Text appears displaying the time or distance between crimes.
Postcondition:	The user is in the view data table with the compare menu displayed.

ID:	UC-6
Name:	Search crime data
Actor(s)	User
Priority:	Medium-High
Status:	Implemented
Description:	Allows the user to search for specific crime(s).
Precondition:	The user is in the view data table window. The application already has some crime data saved in the database.
Basic flow:	1) User clicks on the search button in the left-hand menu. 2) user selects which attribute they would like to search then inputs a string into a text entry box. 3) User clicks “search” 4) The crime view table displays all crimes that contain the string in the search box. If no crime matches the box will be empty.
Postcondition	The user is in the view crime table window.

ID:	UC-7
Name:	External web search
Actor(s)	User, Search API provider

Priority:	Medium-Low
Status:	Implemented
Description:	Allows the user to search for additional crime-related information.
Precondition:	N/A
Basic flow:	1) User selects the external search window. 2) User selects the type of crime information they would like to externally search by selecting from a drop-down menu. 3) User clicks “search” 4) Information about the crime type they selected appears on the screen.
Post Condition:	User in the external search window.

ID:	UC-8
Name:	Add individual crime datum
Actor(s)	User
Priority:	Medium-High
Status:	Implemented
Description:	Allows the user to manually add an individual crime to the application.
Precondition:	User in the add crime window.
Basic flow:	1) The user selects the “new entry” button in the left-hand menu. 2) The add individual crime screen loads in the central pane. The user inputs all details into text boxes and drop-down menus. 3) User selects the “Save” button. 4) The input is checked by the application to ensure the correct format and it is not duplicate data. 5) The input passes the test and the crime is added to the table the user is currently in.
Postcondition:	The crime is now added to the existing data and can be accessed in the application. The user is in the add crime window.
Exceptional flow:	4) The input does not pass the tests and a text prompt appears informing the user what was wrong with the input. Duplicate data or incorrect format.
Postcondition:	No new crime is added to the data, the user is in the add individual crime window.

ID:	UC-9

Name:	Delete crime datum
Actor(s):	User
Priority:	Medium-High
Status	Implemented
Description:	Allows the user to delete individual crime data.
Precondition:	The user is in the individual crime view window. The application has crime data saved.
Basic flow:	1) User clicks on the “Delete crime” Button. 2) The crime is removed from the table. 3) User is returned to the view crime data table.
Postcondition:	User in the view crime data table. The selected crime has been removed from the table.

ID:	UC-10
Name:	Update crime datum
Actor(s)	User
Priority:	Medium
Status:	implemented
Description:	Allows the user to update individual crimes.
Precondition	The user is in the individual crime window. The application has crime data saved in the database. The user has an account and is currently logged in.
Basic flow:	1) The User clicks on the update crime button. All the boxes in the individual crime view window are now able to be edited. 2) The User inputs the changes they would like to make to the crime datum by inputting into text boxes or changing drop-down menus. 3) User click saves 4) The application checks to ensure the changes are correct format 5) The crime is now changed.
Post condition:	The user is in the individual crime view window. The changes are now present.
Exceptional flow:	5) The changes made to the crime do not pass the tests performed by the application. 6) The boxes with faulty input are highlighted red and the user can not save the changes.
Postcondition	The user is in the change crime window for the selected crime but no changes are made.

ID:	UC-11
Name:	View crime data graph
Actor(s):	User
Priority:	Medium-High
Status	Implemented
Description:	Allows the user to view graphs based on overall crime rates of a specific type, area, or time.
Precondition:	The user is in the graph window. The application has some crime data saved in the database.
Basic flow:	1) User selects what data they would like to view by using a series of drop-down menus and text input boxes 2) user selects view graph 3) A graph is generated based on the inputted values.
Postcondition:	User in the graph window a graph is displayed
Exceptional flow:	3) If the user's selections were invalids or no data selected exists a text prompt pops up asking the user to try a different input.
Postcondition:	The user is in the graph window no graph is displayed

ID:	UC-12
Name:	Export data
Actor(s)	User
Priority	Medium
Status	Implemented
Description	Allows the user to export data from within the application to csv or database with or without filter conditions.
Precondition:	NA
Basic Flow:	1) User selects to export data 2) User selects ether csv or database filetype 3) User selects save
Postcondition	The data which was in the table display ether filtered or not, is now exported to ether csv or database file to the selected directory
Alternative Flow:	1) User highlights over the drop down menu and selects ether export with or without filter 2) User selects ether csv or database filetype

	3) User selects save
Postcondition	The data which was in the table is now exported with or without a filter if no filter was applied then all data in the current list will be exported.

ID:	UC-13
Name:	View crime data map
Actor(s):	User and google maps API
Priority:	Medium-Low
Status:	Implemented
Description:	Allows the user to view a crime data map.
Precondition:	The user is in the map window. The application has some crime data saved in the database.
Basic flow:	1) User selects map “Map” button 2) A map is generated based on the current data list
Postcondition:	The user is in the map window. A map is displayed.

ID:	UC-14
Name:	View crime data heat map
Actor(s):	User and google maps API
Priority:	Medium
Status	Future release
Description:	Allows the user to generate a heat map of crime hotspot areas based on the selected attribute.
Precondition:	The user is in the map window. The application has some crime data saved in the database.
Basic Flow:	1) User selects what data and attributes they would like the map to generate from, by using a series of drop-down menus and text input boxes 2) User selects the “view heatmap” button. 3) A heatmap is generated based on the chosen attributes and data.
Postcondition:	The user is in the map window. A heat map is displayed.
Exceptional flow:	3) If the user's selections were invalids or no data selected exists a text prompt pops up asking the user to try a different input.

Postcondition.	The user is in the heat map window and no graph is displayed.

2.3: Quality Requirements

Table 3 shows a list of unique quality requirements that will be considered and used during the creation of the project. These quality requirements express the type of system our final product should aim to achieve to satisfy the business context outlined at the start.

All requirements are based on the principle of SMART (Specific, Measurable, Agreed, Realistic, Time-Bound). This is done to allow us to identify when requirements have been met during the project and ensure that the requirements themselves are realistic enough to be efficiently implemented into the application.

Table 3: Quality requirements and their relevant stakeholders and priorities:

ID:	Description:	Stakeholders:	Priority:
QR_1	The application must be compilable and runnable on lab computers or systems with the same instance of java installed.	Development Team, Teaching Staff	High
QR_2	The application does not violate any laws such as user information privacy or name licensing issues.	Development Team, Teaching Staff, Google, Police, University of Canterbury, Criminals	High
QR_3	The application must be maintainable, more specifically, code must be readable, well documented and have high coverage using unit tests.	Development Team	High
QR_4	The application must be usable with no major bugs or issues that could affect the users ability to use the application.	Development Team, Teaching Staff, Prospective Renters/Buyers, Student Flatters, Family's/Home Owners	High
QR_5	The application response times are reasonable, a couple seconds, and if longer load times are expected a loading bar will be used.	Prospective Renters/Buyers, Student Flatters, Family's/Home Owners	Medium

QR_6	The application should have a simple aesthetic that conveys information to users efficiently.	Prospective Renters/Buyers, Student Flatters, Family's/Home Owners	Medium
QR_7	The application should not be complex and providing high-level analysis of data should be usable for users unfamiliar with complex computer systems.	Prospective Renters/Buyers, Student Flatters, Family's/Home Owners	Medium
QR_8	Interactions with the application systems must be consistent and repeatable so as to not confuse any user.	Prospective Renters/Buyers, Student Flatters, Family's/Home Owners	Medium
QR_9	The application must follow the coding conventions set out by the development team.	Development Team	Medium
QR_10	The analysis of data must be accurate in the results it shows.	Prospective Renters/Buyers, Student Flatters, Family's/Home Owners	Medium
QR_11	The application should maintain additional information created by a user in a separate table to imported crime data.	Student Flatters, Family's/Home Owners	Low

While all quality requirements are useful to improve the application some requirements are of a higher priority as it ensures a successful and useful project for the stakeholders that will be using it. These requirements are identified below in Table 4 using a weighting system the stakeholder's weights are mapped as High = 1, Medium = 0.5 and Low = 0.15, low stakeholders are considered much lower as they interact with a system very little.

Each stakeholder was then analysed against the quality requirements in Table 3 to determine the value that each quality requirement has for them. For each stakeholder, their weights were then multiplied against the quality requirement value and added together to produce a total value shown in Table 4. This provided the 5 key drivers with the largest effect on the project; QR_1, QR_2, QR_4, QR_5 and QR_7. Requirement 2 while not as in line with the business context as the other listed requirement is important as it represents our potential low-end stakeholders concerns for the project.

Table 4: Key drivers as determined by the project's stakeholders and quality requirements.

Stakeholder:	**Weight:**	QR_1	QR_2	QR_3	QR_4	QR_5	QR_6	QR_7	QR_8	QR_9	QR_10	QR_11
Development Team	1.0	25	15	20	25	0	0	0	0	15	0	0
Prospective Renters/Buyers	1.0	0	0	0	25	20	10	20	10	0	15	0
Student Flatters	1.0	0	0	0	25	15	10	15	10	0	10	15
Teaching Staff	0.5	50	25	0	25	0	0	0	0	0	0	0
Family's/Home Owners	0.5	0	0	0	25	15	10	15	15	0	15	5
Google	0.5	0	100	0	0	0	0	0	0	0	0	0
University of Canterbury	0.15	0	100	0	0	0	0	0	0	0	0	0
Criminals	0.15	0	100	0	0	0	0	0	0	0	0	0
Police	0.15	0	100	0	0	0	0	0	0	0	0	0
Total:		50	123	20	100	43	25	43	28	15	33	18

3: Acceptance Testing

The criteria for any acceptance tests that are required to be performed on the system generated are contained in Table 5. When performing the tests the ID can be matched with the outcome of the test and input into the relevant table. The criticality was separated into three main components, High, Medium and Low. The criticality took a base in the priority of the use case that each test links to, with modifiers based on if the test was an edge case or expected input. Other factors that raise the criticality of a test are those that, if they fail, the application cannot run correctly or tests that have multiple dependencies. Factors that reduced the criticality of the test from the use case priority were if they involved circumstances that are unlikely to occur in regular use of the project or could not affect the operation of the application if they did break. For example, in the case of UC-19, there are 3 different methods of continuance, if one of these occurred by default without giving the user a choice it could be acceptable, therefore the criticality is lowered from High to Medium-High. The subsections of UC-19 are reduced further for the same reason.

Table 5: Acceptance tests covering the use-cases outlined in section 2.2 are ordered by their criticality then by use case.

ID:	Description:	Acceptance criteria:	Criticality:	Use case:
AT-1	**Given** a file of any size containing valid data **When** the user loads the file **Then** all data can be found in the data view	The user is given an indication that the data is uploading before being able to find the crimes that they uploaded and view them according to the view requirements	High	UC-1 Import crime data
AT-2	**Given** data is loaded **When** the user selects an entry **Then** the view changes and gives details about the crime	Detailed information about the crime is visible to the user as well as options to modify the data	High	UC-3 View individual crime datum
AT-3	**Given** data is loaded **When** the user adds a constraint **Then** only data relevant	Only data relevant to the constraint is visible to the user	High	UC-6 Search crime data

	to the constraint is displayed			
AT-4	**Given** a file of any size with duplicate data after the user loads the file **When** the user selects overwrite **Then** all data is uploaded and visible in the data view a warning pops up and asks if the user want to overwrite, skip or cancel	A warning box appears correctly	Medium-High	UC-1 Upload crime data
AT-5	**Given** data is loaded **When** the user selects multiple crimes **Then** they can compare the selected data	User is able to see information like distance and time from crime A to crime B	Medium-High	UC-5 Compare crime data
AT-6	**Given** data is loaded **When** the user provides a date range **Then** data outside of that range will not be visible	The data that meets the data range criteria provided will be visible in the data view and all other data will be hidden	Medium-High	UC-6 Search crime data
AT-7	**Given** data is loaded **When** the user adds multiple constraints **Then** only data relevant to all constraints are shown	Only data relevant to all constraints is visible (not just one constraint)	Medium-High	UC-6 Search crime data

AT-8	**Given** data is loaded and at least one constraint is active **When** the user deletes a constraint **Then** data not relevant to the constraint are shown	Only data relevant to current constraints is visible	Medium-High	UC-6 Search crime data
AT-9	**When** the user inputs valid data **Then** data will appear in the data table/database	User can see their own input and it is correctly analysed	Medium-High	UC-8 Add individual crime datum
AT-10	**When** the user inputs invalid data **Then** a warning will inform the user what data is problematic	The warning shows incorrect data and no new data is visible in the data table	Medium-High	UC-8 Add individual crime datum
AT-11	**Given** correct data is stored **When** the user deletes entry **Then** entry cannot be found in the data view	Data entry is removed from the database and the user can no longer find it	Medium-High	UC-9 Delete crime data
AT-12	**Given** data is loaded **When** the user adds axis constraints **Then** a graph is generated visual comparing crime data along the two axis	Selected crime data is present in a graph view which plots entries in relation to constraints selected	Medium-High	UC-11 View crime graph
AT-13	**Given** AT-4	All data from the file is uploaded	Medium	UC-1 Upload crime data

	When the user selects overwrite **Then** data continues uploading	and visible, duplicate entries have had any different values replaced with new values		
AT-14	**Given** AT-4 **When** the user selects skip **Then** non-duplicate data continues uploading	All non-duplicate data from the file is uploaded and duplicate entries have not been modified	Medium	UC-1 Upload crime data
AT-15	**Given** AT-4 **When** the user selects cancel **Then** data stop uploading	No data from the file is uploaded	Medium	UC-1 Upload crime data
AT-16	**Given** a file of any size containing a number of missing attributes **When** the user loads the file **Then** a warning box appears and no data is uploaded	No data is added to the database and warning gives an indication on what went wrong	Medium	UC-1 Add crime to application
AT-17	**Given** data is loaded **When** the user adds sort constraint **Then** data listed in the data table is in sorted order	All crimes are ordered by the constraint selected by the user. Entries that have the same value in the constraint are then sorted by ID number	Medium	UC-4 Sort crime data
AT-18	**Given** correct data is stored **When** the user modifies	Old data is modified to be consistent with input data and is represented correctly	Medium	UC-10 Update crime data

	the entry with correct input **Then** updated information is visible in the view table			
AT-19	**Given** correct data is stored **When** the user modifies the entry with an invalid input **Then** a warning will inform the user what data is problematic	Old data is unchanged and the user can see what data is invalid	Medium	UC-10 Update crime data
AT-20	**Given** data is loaded **When** viewing data through the map view **Then** the main view changes to display a map with indicators of crimes	A map is visible on the page which has some form of indicator for where crimes have taken place	Medium	UC-12 View crime data map
AT-21	**Given** data is loaded **When** the user exports without filter **Then** the data can be exported without the filter	Regardless of a filter being applied, exporting exports the data to a csv without any data missing	Medium	UC-12 Export data
AT-22	**Given** data is loaded and filter is given **When** the user exports with filter **Then** the data can be exported with the filter	Exporting exports only the filtered data to a csv without any data missing	Medium	UC-12 Export data
AT-23	**Given** map view is open	The heatmap regenerates to match	Low-Medium	UC-13 View crime data

	and data is loaded **When** the user enables the heat-map **Then** a heatmap is overlaid on the map	the new data range provided and visual update		heat map
AT-24	**Given** heatmap overlay is on with a constraint active **When** the user modifies a constraint **Then** the heat map updates to meet new constraints	The heat map is updated for the new constraint set	Low-Medium	UC-13 View crime data heat map UC-6 Search crime data

4: GUI Prototypes

The proposed GUI is composed of a menu bar and a tab group. Figure 3 shows the menu bar containing all of the tools you can use to manipulate the data. The menu bar can also be collapsed to display more data as shown in Figure 4.

The view mode can be selected from the tab group from any other view, which allows the user to quickly switch between view modes on the same data. With this design, changing views should be independent of using tools on the data, making it possible for the user to have persistent constraints like sorting or filtering while changing views. These constraints are created by accessing one of the submenus shown in Figure 7.

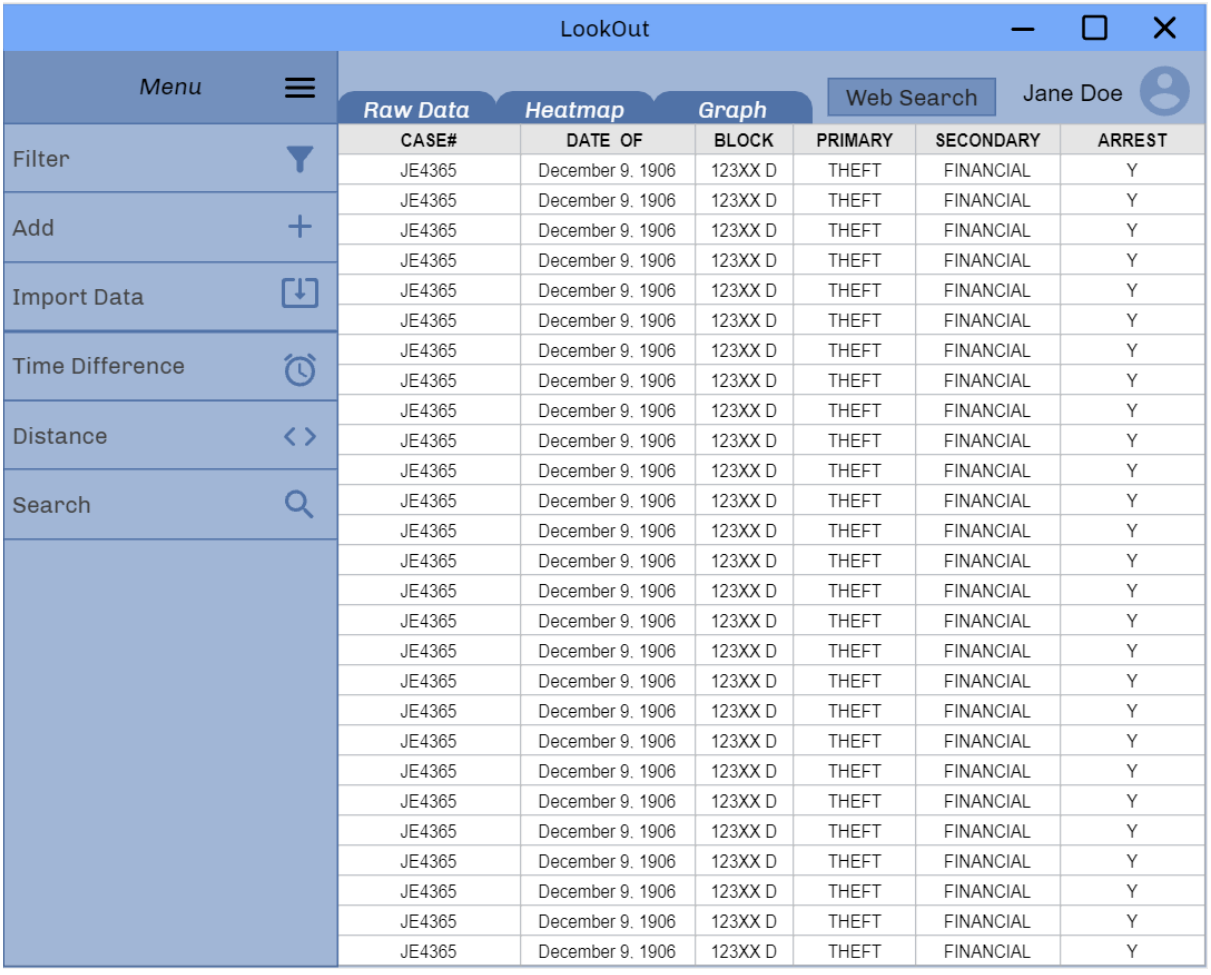

Figure 3: The raw data view with the menu in the open state.

This view is associated primarily with UC-2 View crime data table. UC-4 Sort Crime data can be done by clicking on a column heading to sort by it. UC-7 External web search can be accessed by the web search button on any screen. UC-1 Import crime data is fulfilled by clicking “Import data " on the menu which will bring up a file chooser. Users will be able to click the profile icon which will bring up a login page for useless cases 14 and 15.

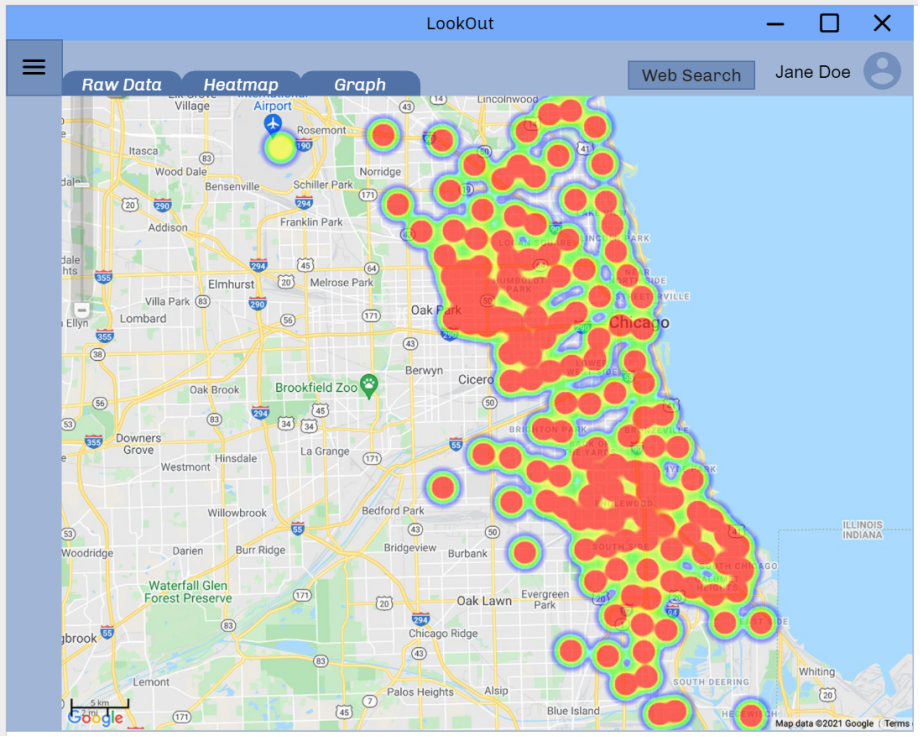

Figure 4: The map view overlaid with a heatmap with the menu in the collapsed state. This heatmap view is associated with UC-12 and UC 13 View Crime data heat map.
Map image: City Of Chicago. (2019)

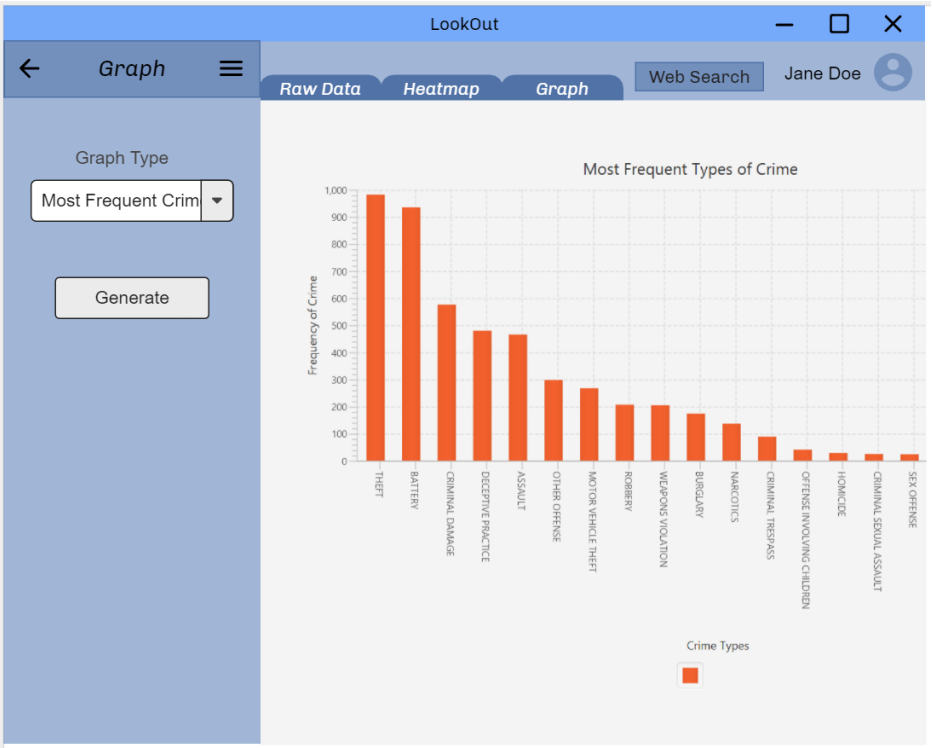

Figure 5: The graph view with the graphing specific submenu displayed. This view is associated with UC-11 View crime data graph.

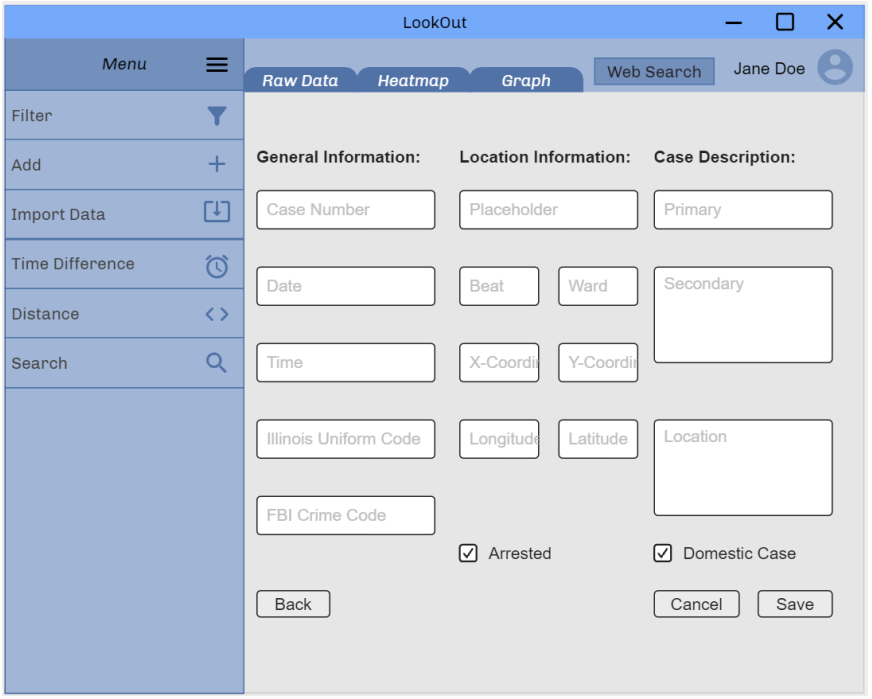

Figure 6: The entry view for adding a crime into the database. This view is associated with use cases UC-8 Add individual crime datum.

In addition to being used to add data, a modified version of this gui is used for use case 3, 9 and 10. This allows for viewing, updating and deleting individual crimes.

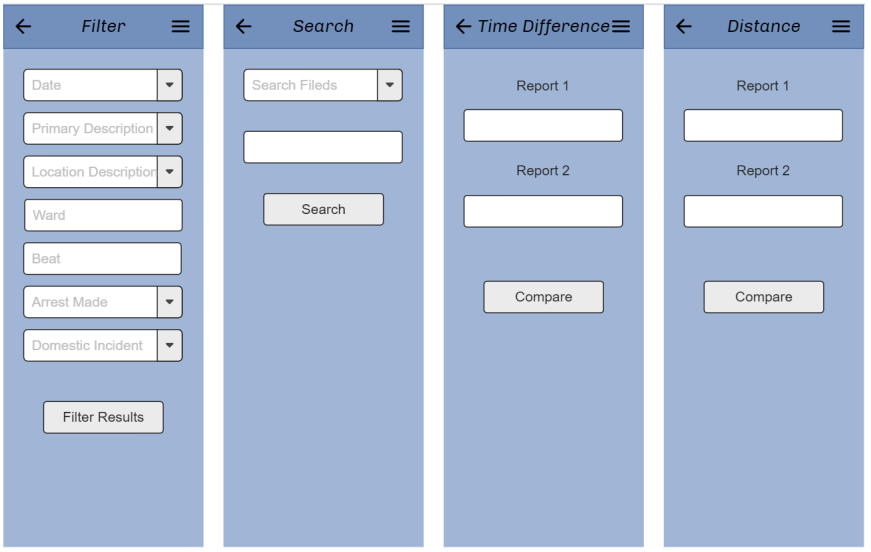

Figure 7: The menus for performing actions on the data.These are associated with UC-6 Search crime data and UC-5 Compare crime data.

This is the currently proposed design but is subject to change as the development process goes on. The colour scheme was the default one of the tool used and may not reflect the final product. As the project progressed, the structure of the interface hasn't changed much. The components in the application currently have the default java fx look and feel, but in the future .css files will be used to give them the desired appearance.

5: Deployment Model

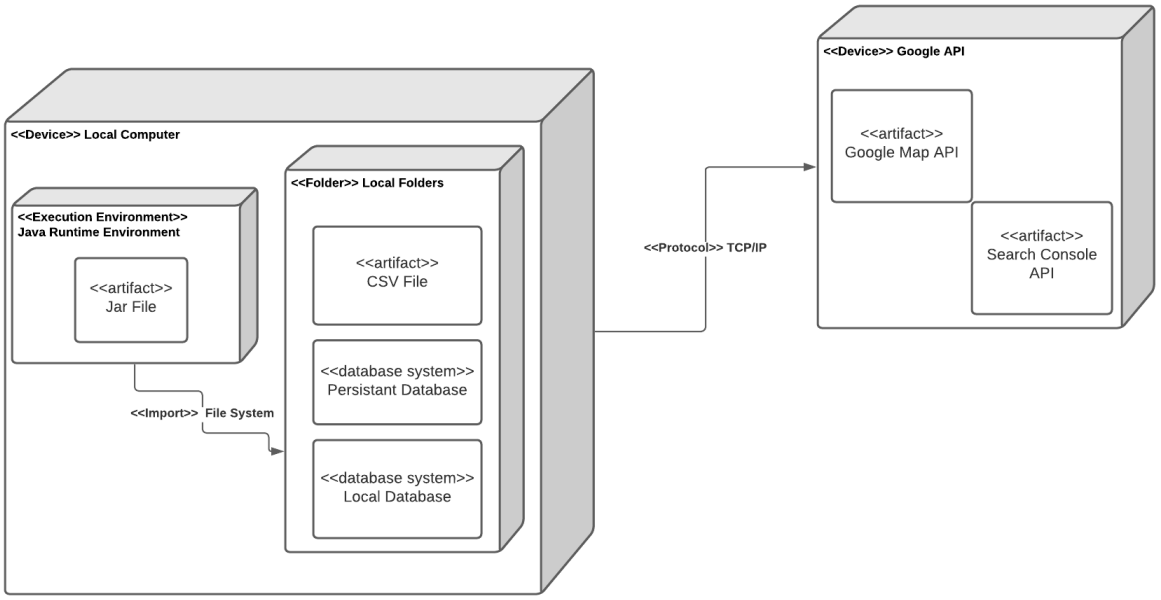

Figure 8: The deployment model of the LookOut application

LookOut is a single user application therefore both its executable JAR file and data will be stored on a local computer. The JAR file will be run through the JRE (Java Runtime Environment) and will access either a local database or a CSV file to import its crime data and store it as shown in a persistent database as shown in Figure 8.

The only connection outside the local computer is to the Google API which will be used to provide a mapping system and will be accessed through the TCP/IP protocol.

6: Detailed UML Class Diagram

The 3-Tier architecture was chosen as this is a tried and tested architecture that fits our use cases and will allow us to implement all our features in a structured manner. The architecture consists of a View layer which represents the graphical interface the user interacts with, the Control layer which communicates with the GUI and the Model layer which houses the representation of the data. The full diagram is available here.

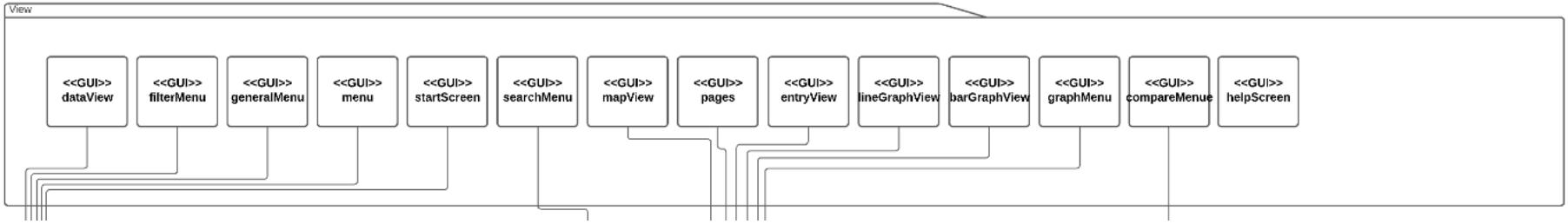

Figure 9: The view package containing the GUI elements.

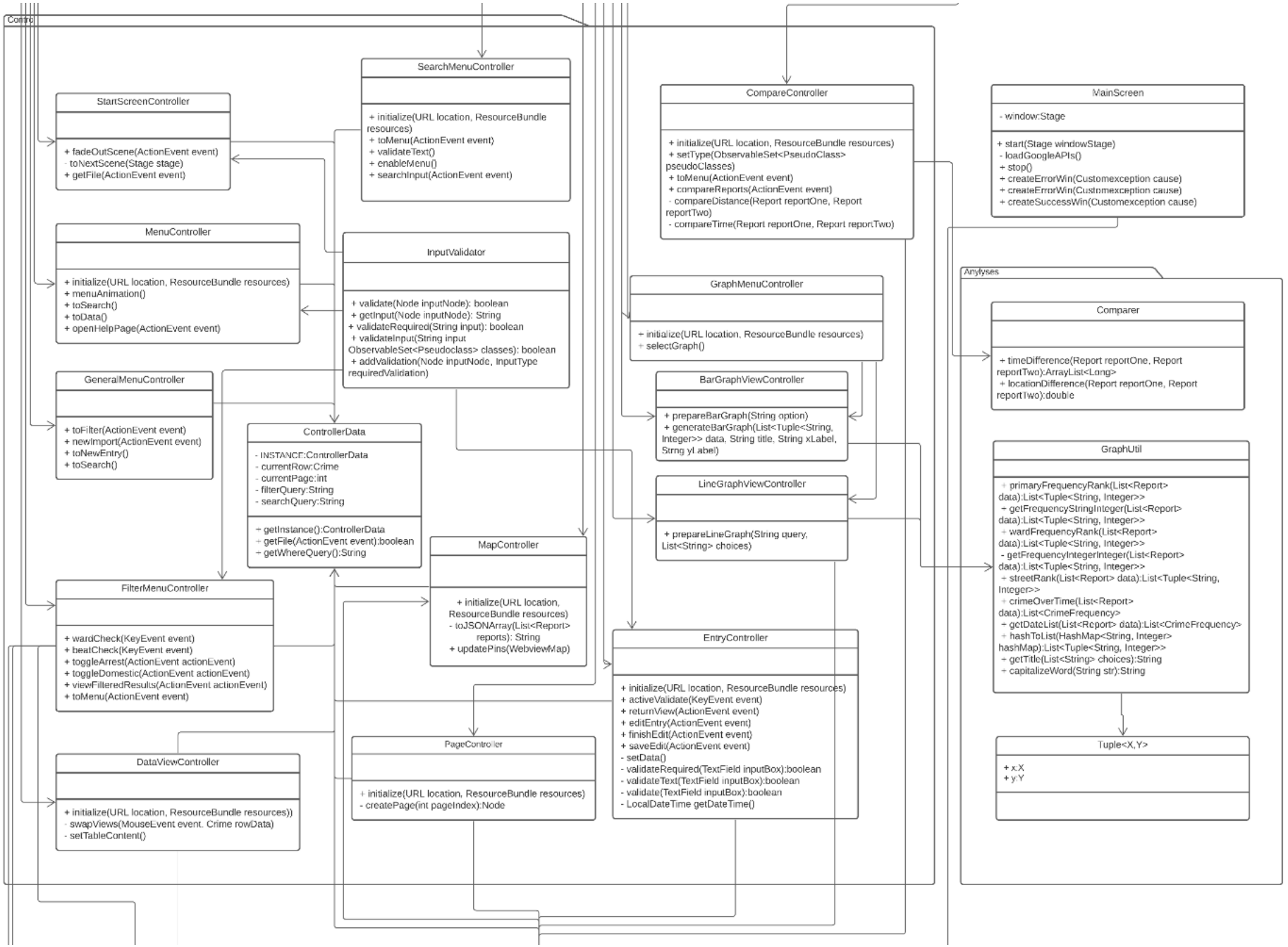
Contrl
SearchMenuController
+ initialize(URL location, ResourceBundle resources)
+ toMenu(ActionEvent event)
+ validateText()
+ enableMenu()
+ searchInput(ActionEvent event)
StartScreenController
+ fadeOutScene(ActionEvent event)
- toNextScene(Stage stage)
+ getFile(ActionEvent event)
CompareController
+ initialize(URL location, ResourceBundle resources)
+ setType(ObservableSet<PseudoClass> pseudoClasses)
+ toMenu(ActionEvent event)
+ compareReports(ActionEvent event)
- compareDistance(Report reportOne, Report reportTwo)
- compareTime(Report reportOne, Report reportTwo)
MainScreen
- window:Stage
+ start(Stage windowStage)
- loadGoogleAPIs()
+ stop()
+ createErrorWin(Customexception cause)
+ createErrorWin(Customexception cause)
+ createSuccessWin(Customexception cause)
MenuController
+ initialize(URL location, ResourceBundle resources)
+ menuAnimation()
+ toSearch()
+ toData()
+ openHelpPage(ActionEvent event)
InputValidator
+ validate(Node inputNode): boolean
+ getInput(Node inputNode): String
+ validateRequired(String input): boolean
+ validateInput(String input ObservableSet<Pseudoclass> classes): boolean
+ addValidation(Node inputNode, InputType requiredValidation)
GraphMenuController
+ initialize(URL location, ResourceBundle resources)
+ selectGraph()
Anylyses
Comparer
+ timeDifference(Report reportOne, Report reportTwo):ArrayList<Long>
+ locationDifference(Report reportOne, Report reportTwo):double
GeneralMenuController
+ toFilter(ActionEvent event)
+ newImport(ActionEvent event)
+ toNewEntry()
+ toSearch()
ControllerData
- INSTANCE:ControllerData
- currentRow:Crime
- currentPage:int
- filterQuery:String
- searchQuery:String
+ getInstance():ControllerData
+ getFile(ActionEvent event):boolean
+ getWhereQuery():String
BarGraphViewController
+ prepareBarGraph(String option)
+ generateBarGraph(List<Tuple<String, Integer>> data, String title, String xLabel, Strng yLabel)
GraphUtil
+ primaryFrequencyRank(List<Report> data):List<Tuple<String, Integer>>
+ getFrequencyStringInteger(List<Report> data):List<Tuple<String, Integer>>
+ wardFrequencyRank(List<Report> data):List<Tuple<String, Integer>>
+ getFrequencyIntegerInteger(List<Report> data):List<Tuple<String, Integer>>
+ streetRank(List<Report> data):List<Tuple<String, Integer>>
+ crimeOverTime(List<Report> data):List<CrimeFrequency>
+ getDateList(List<Report> data):List<CrimeFrequency>
+ hashToList(HashMap<String, Integer> hashMap):List<Tuple<String, Integer>>
+ getTitle(List<String> choices):String
+ capitalizeWord(String str):String
LineGraphViewController
+ prepareLineGraph(String query, List<String> choices)
MapController
+ initialize(URL location, ResourceBundle resources)
- toJSONArray(List<Report> reports): String
+ updatePins(WebviewMap)
FilterMenuController
+ wardCheck(KeyEvent event)
+ beatCheck(KeyEvent event)
+ toggleArrest(ActionEvent actionEvent)
+ toggleDomestic(ActionEvent actionEvent)
+ viewFilteredResults(ActionEvent actionEvent)
+ toMenu(ActionEvent event)
EntryController
+ initialize(URL location, ResourceBundle resources)
+ activeValidate(KeyEvent event)
+ returnView(ActionEvent event)
+ editEntry(ActionEvent event)
+ finishEdit(ActionEvent event)
+ saveEdit(ActionEvent event)
- setData()
- validateRequired(TextField inputBox):boolean
- validateText(TextField inputBox):boolean
- validate(TextField inputBox):boolean
- LocalDateTime getDateTime()
Tuple<X,Y>
+ x:X
+ y:Y
PageController
+ initialize(URL location, ResourceBundle resources)
- createPage(int pageIndex):Node
DataViewController
+ initialize(URL location, ResourceBundle resources))
- swapViews(MouseEvent event, Crime rowData)
- setTableContent()

Figure 10: The control and analyses packages.

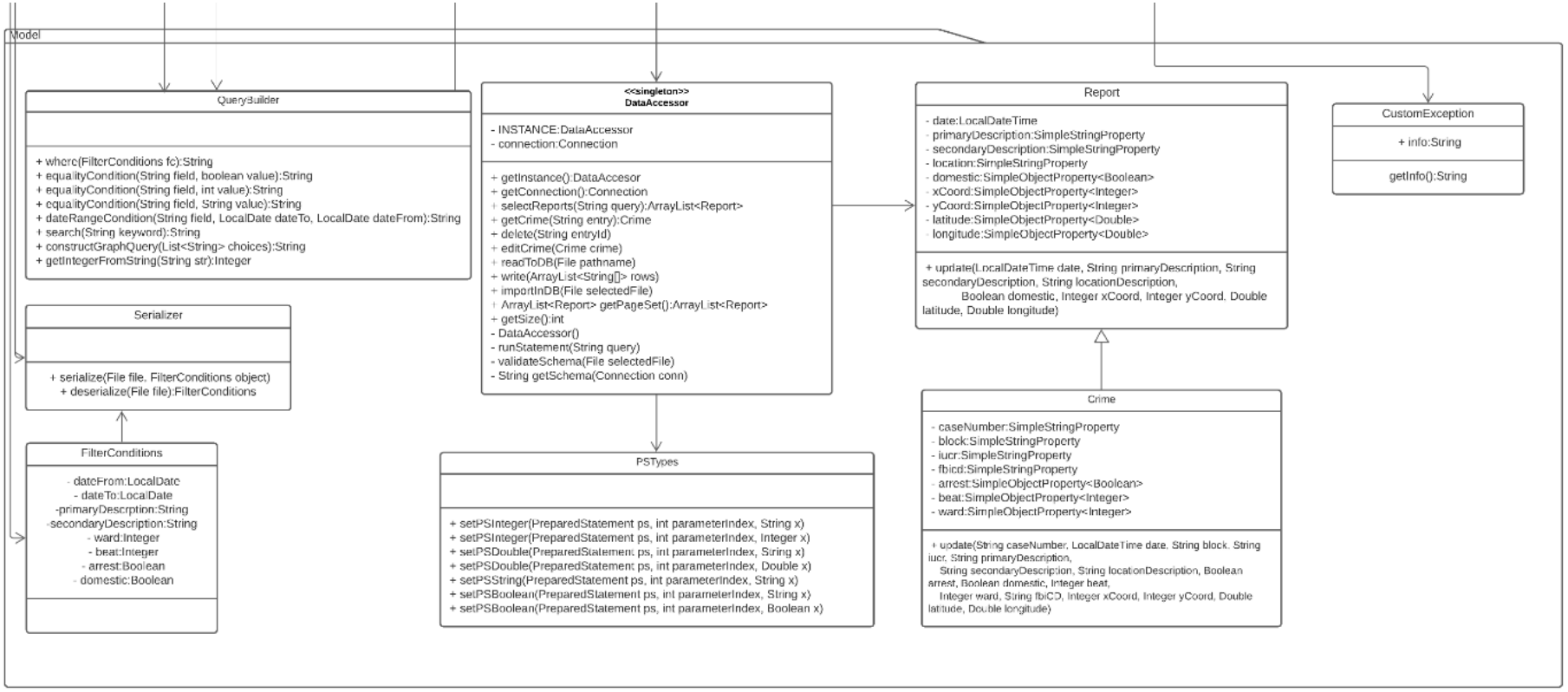

Figure 11: The model class package contains the required classes to interact with the database and represent the application’s data.

View Layer (Figure 9):

- The view layer contains all the .fxml files for the application's GUIs. Each file is linked to a controller in the control package. The GUIs are represented as GUI stereotypes as modeling them accurately would be time consuming and of little consequence to the quality of the diagram.

Control Layer (Figure 10):

- The controller classes are responsible for taking actions from the GUI and communicating with the data access layer to manipulate the database appropriately.
- The ControllerData class stores the current state of the table. This class acts as a connection between the controllers passing data through itself, so that the controller classes can access the data the user is currently viewing rather than having to retrieve it again. It accomplishes this by being a singleton class so only one instance of it can exist.
- The LineGraphViewController and BarGraphViewController control the graph view. The GraphMenuController allows for switching between them. The MapViewController controls the map view through a webview node that interacts with the google maps api. The other controllers each control a menu element, such as searching, filtering or viewing an individual entry.
- Analyses Package:
    - This package contains classes that analyse the data. The compare class is used by the CompareController and the Rank and Tuple classes by the GraphViewController. These classes mainly serve to move some of the large data set processing code out of the controllers.

Model Layer (Figure 11):

- The Report class is an object representation of a crime in the database.
- The Crime class inherits Report and represents a crime reported by the authorities. It has a unique caseNumber and several other attributes which would not be included in civilian reports.
- The DataAccessor class serves as the DAL for the application. The classes in the controller package will communicate with this to retrieve, edit and delete the data. This class is implemented as a singleton to ensure we are only accessing data from one database at a time.
- The PSTypes class contains static methods to add fields to prepared statements for database queries. These functions allow us to handle null values and empty strings in the desired manner.

- The Query builder class contains methods to build a query for filtering, searching and gaphing. Based on some parameters, it will return a String representation of a sqlite query.

7: Testing Procedures

(Test protocol to increase confidence in the quality of product)

Manual testing was largely based around the project's acceptance tests, having a manual test corresponding to each acceptance test. Most of the acceptance tests were covered with only a couple having failed. As shown in Table 6, these were minor failures as they involved incorrect interpretation and implementation of the requirements, such as filtering by only one date instead of two. Due to time constraints, some acceptance tests were not able to be given manual testing as there was not enough time to implement those features, so they have been left to be added to the next release, along with fixing the failing tests.

While acceptance testing focuses on high priority items, the table considers criticality and priority as separate categories, and so, includes manual tests of low criticality acceptance tests. Also, the description for each acceptance test covered diverse cases for the app's features, making the manual tests cover diverse cases too.

Manual Testing:

Table 6: Manual tests performed on the application and the outcome.

Manual Test ID	Acceptance Test	Description	Tester	Result Outcome of test	Pass/fail	Criticality	Use case
MT-1	AT-1	Uploading a file with valid data	Sami	The user can find the crimes that they uploaded and view them according to the view requirements	Pass	High	UC-1 Import crime data
MT-2	AT-2	Double click entry to view more information	Sami	Detailed information about the crime is visible to the user as well as options to modify the data	Pass	High	UC-3 View individual crime datum

MT-3	AT-3	Filter data against one constraint	Sami	Only data relevant to the constraint is visible to the user	Pass	High	UC-6 Search crime data
MT-4	AT-4	Uploading a file with duplicate data	Jack	A warning box appears asking if the user wants to overwrite, skip or cancel	Pass	Medium-High	UC-1 Upload crime data
MT-5	AT-5	Comparing two crimes by distance	Sami	The application shows the distance between the two crimes	Pass	Medium-High	UC-5 Compare crime data
MT-6	AT-6	Filter data between two dates	Sami	Only data that is between the two dates provided is displayed	Pass	Medium-High	UC-6 Search crime data
MT-7	AT-7	Multiple constraints are added to the filter conditions	Sami	Data is correctly filtered to display only data relevant to all constraints	Pass	Medium-High	UC-6 Search crime data
MT-8	AT-8	Deleting constraint from filter conditions	Jack	Data that was filtered due to deleted filter is visible again	Pass	Medium-High	UC-6 Search crime data
MT-9	AT-9	User tries to create new entry with valid data	Sami	User can view their own input in the database	Pass	Medium-High	UC-8 Add individual crime datum
MT-10	AT-10	User tries to create new entry with invalid data	Sami	Values that are invalid are outlined in red and entry is not created	Pass	Medium-High	UC-8 Add individual crime datum
MT-11	AT-11	User tries to remove data entry	Sami	Data entry is removed from the database and the user can no longer find it	Pass	Medium-High	UC-10 Delete crime data

MT-12	AT-12	User switches to graph view	Sami	Selected crime data is present in a graph view which plots entries in relation to constraints selected	Pass	Medium-High	UC-11 View crime graph
MT-13	AT-13	User tries to overwrite duplicate data	Jack	Duplicate data is overwritten	Pass	Medium	UC-1 Upload crime data
MT-14	AT-14	User tries to skip duplicate data	Jack	Duplicate data is unchanged in the database	Pass	Medium	UC-1 Upload crime data
MT-15	AT-15	User tries to cancel upload of duplicate data	Jack	User selects cancel and a warning box appears when duplicate data is detected to inform the user of the cancellation	Pass	Medium	UC-1 Upload crime data
MT-16	AT-16	User tries to skip uploading duplicate data	Jack	User selects skip and existing records are skipped	Pass	Medium	UC-1 Upload crime data
MT-17	AT-17	User tries to sort data by an attribute of the entry	Jack	Implementation of function required has not been completed	Fail	Medium	UC-4 Sort crime data
MT-18	AT-18	User tries to edit an entry with valid data	Sami	Old data replaced with the new data and can be viewed on the data table	Pass	Medium	UC-9 Update crime data
MT-19	AT-19	User tries to edit an entry with invalid data	Jack	Input boxes containing invalid data become outlined in red and the data is not updated	Pass	Medium	UC-9 Update crime data
MT-20	AT-20	User switches view to the map view	Jack	Map appears with markers visible where crime data is recorded	Pass	Medium	UC-13 View crime data map

MT-21	AT-21	User exports without filter	Sami	Data exports to csv without filter	Pass	Medium	UC-12 Export data
MT-21	AT-22	User exports with filter	Sami	Data exports to csv with filter	Pass	Medium	UC-12 Export data
MT-23	AT-23	User enables the heat map in the map view	Jack	No heatmap implemented	Fail	Low-Medium	UC-14 View crime data heat map
MT-24	AT-24	User changes the constraints on the data when viewing the heatmap	Jack	No heatmap implemented	Fail	Low-Medium	UC-14 View crime data heat map UC-6 Search crime data

JUnit Testing:

Package	Class, %	Method, %	Line, %
seng202.group7	0% (0/ 1)	0% (0/ 2)	0% (0/ 3)
seng202.group7.analyses	100% (4/ 4)	84.2% (16/ 19)	95.2% (139/ 146)
seng202.group7.controllers.data	33.3% (1/ 3)	23.3% (7/ 30)	13.4% (15/ 112)
seng202.group7.controllers.menus	0% (0/ 6)	0% (0/ 60)	0% (0/ 565)
seng202.group7.controllers.views	0% (0/ 7)	0% (0/ 56)	0% (0/ 444)
seng202.group7.data	87.5% (7/ 8)	65.7% (69/ 105)	58.2% (373/ 641)
seng202.group7.view	0% (0/ 1)	0% (0/ 8)	0% (0/ 49)

Figure 12 Coverage of JUnit tests.

We created a set of JUnit tests that have a 100% coverage of our classes within our analyses and data packages. Due to the controller and view packages being connected to the GUI we did not focus on creating JUnit tests for these sections as they were tested using manual testing. We have increased analyses testing to about 84% method coverage. Since the controllers were connected to fxml files they could not be improved using unit testing. Finally, our data method coverage was quite low due to lack of time.

Functional Testing:

Based on the functional requirements that were outlined in the use-cases we created a set of JUnit tests to ensure as many of these were tested as possible. Some key testing was;

In relation to the UC-1, importing crime data. Our DataAccessor class had testing for importing data into the application both .CSV and .DB files. This testing also ensured that these were valid files before adding the data to the persistent database.

UC-2, UC-3, UC-4 and UC-6, which involved getting data from our database, were tested. This was done by ensuring when accessing the DataAccessor class, it returned the correct set of data that was being displayed to the user. Further testing was manually done to ensure this was then correctly displayed.

UC-5 had JUnit tests that tested the Compare Classes static methods to ensure that data was being correctly calculated, when comparing two report objects. UC-8, UC-9 and UC-10, which involved creating or removing data from our database, had JUnit tests created. These tested potential scenarios and ensured our system would correctly handle them, without causing SQLite errors or unintended data.

UC-11 was manually tested to ensure the graphs produced the correct outputs. UC-12 also exported all data correctly depending on the filter or without one. UC-13 was the last passing case with the pins on the mapping feature appearing correctly. However, due to not implementing the heatmaps, UC-14 failed.

Quality Testing:

Based on the quality requirements given and their priorities, tests were carried out to ensure the quality of the application was always up to standard. QR_1 and QR_2 were taken very seriously and were consistently made sure to pass. This was done by testing new features in separate Git branches before merging them to master, and immediately addressing conflicts that appeared. As shown by the JUnit testing and manual testing QR_3 was maintained, and in doing so, alongside the merge conflict checks, QR_4 was met by removing major bugs as they appeared.

For medium priority requirements, QR_5 and QR_6 were met. Loading animation was implemented for long response times when navigating the application that exceed 1000ms. This happens during importation of larger data files. The simplicity of the application is met, and providing proper user feedback has been done, by outlining invalid inputs in red, and showing error boxes when the application fails or inputs are incorrect. QR_7, QR_8, QR_9, and QR_10 have all been met. The application is not complex and is very usable, while providing repeatable and consistent interaction. Code follows coding conventions using our own checkstyle system in our IDE's and analysed data accurately presents the output.

QR_11 is the only requirement that has not been met at all. This feature has not yet been implemented, but this is considered reasonable as it is a low priority requirement.

Discussion (overall test results and test coverage):

Overall, we frequently tested with our manual testing procedures and verified that our GUI system was interacting in the intended way. We also created a set of JUnit tests that covered classes that handled the logic and data processing of our application. As was shown in Figure 12, we completed 100% coverage in the analyses classes but our method coverage was lower than was expected. This was because many of our classes were fxml controllers and only worked within the system. To counteract this some logic was taken away from fxml controllers and given other classes to allow for further unit testing. The testing of data class was improved since our last submission, but since it is linked to controllers it was quite hard to implement to over 60%, especially with the time constraint. We made testing for a single issue difficult without rewriting the code. Also, we ran out of time to fully complete all JUnit testing that was intended and some classes lacked diverse testing because of this. Cucumber testing was also not implemented due to lack of prioritisation and time during phase 3. We believed that the functionality provided by Cucumber was not a necessity due to lack of end user testing given the nature of the project. We instead decided to focus on manual and unit testing.

8: Current Product Version

8.1: Use cases and features implemented in this release:

Most of the features described in the use cases have been implemented into deliverable 2 of the application. The user is able to import or create a new list of data satisfying UC-1 and UC-8. From this, the user is sent to the data view screen which provides a table containing all the entries which have been imported to the database fulfilling the requirements for UC-2. This view also contains the tools for UC-1, UC-5, UC-6, and UC-8 with a menu that provides the user with the options to search, filter, compare, add a new entry, or import more data. Double-clicking on a crime entry will send the user to a new page with a more detailed view of the entry, from this page the user is able to delete and update the entry covering UC-3, UC-9, and UC-10 respectively.
Two other views also exist in the application each meeting the requirements for UC-11 and UC-7. These views can be accessed by the menu bar and lead to the graph view and the external search view. The graphing view allows the user to analyse the data on one of the two graph types. Bar graph which displays 7 different bar graphs based on the data and a line graph which displays crime over time statistics. The external search allows the user to search for information from a set of selected sites.
The product has been updated this deliverable to fulfil more of the remaining use cases and acceptance tests that were unfulfilled in deliverable 2. UC-13 is now covered by the new map view which allows the user to view a map with pins where loaded crimes exist. We also covered more of the acceptance test requirements that we were missing. This mostly covers the ability to select how to handle duplicate and invalid data during importing but the search functionality and compare menus useability were both also improved. The search menu had a selector and button removed with the behaviour of searching all columns on a keypress simplifying the design and making it more intuitive to use. The compare menu was improved by adding buttons to select the highlighted crime rather than having to type.

8.2: Features you are most proud of:

Validation

The validation of input values throughout our application is completed in a way that allows for new input boxes to be added with ease. It works by adding to the input's pseudo-classes, the type of validation that is required. If the value does not conform to the required format, the error pseudo-class defined in the CSS will be applied. This gives the input box a red border to provide visual feedback to the user.

Double click view

With one of our goals with the application being easy to access and use, having the ability to double click on an entry to transition into a separate view of that particular entry appears intuitive. The view that it connects to is also where you could find the option to edit or delete the entry. An alternative to this that was discussed was having a view entry button on the side menu, the current implementation was chosen due to it being more straightforward for the user.

Pagination

Another method that we used to ensure our application is usable was to use a paginator in the data view screen. The paginator reduces the number of entries visible at once and splits the data into multiple pages, currently set to 1000 per page, so it is easier for the user to find an entry reliably. Another advantage of the reduced number of entries per page is how it allows the application to reduce the size of queries to the database decreasing the resource requirement of the user's system.

Multi-Threading

When importing and exporting data a different thread is used. This allows the user to continue using other elements of the application that don't alter the database while the import/export is happening. It does this by beginning transactions when altering data and catching an error if a transaction is already occurring. This improves the usability of the app and prevents the app from appearing to have frozen. While one of these actions is occurring, the mouse will change to show a loading animation to provide the user with feedback.

8.3: Requirements and features not implemented in this release:

For the final product, there were only 2 features and some quality of life changes that were unable to be implemented. The first of the two features was the Heat-map. This was planned as a unique selling point from the beginning of the project but when the mapping view wasn't implemented until after deliverable two it became unrealistic for it to be implemented and tested to a standard the team could be happy with. The other feature was the login system. This was going to be an interesting feature to implement due to how it could have to interact with the other controllers while minimising coupling. Due to these reasons and the short time remaining for the application to be ready, the team decided it could be best to continue working on existing features and making them as functional as possible.

The QoL changes that we ran out of time to add was the sort functionality. The user can sort a single page of data by any attribute but when they switch pages it reverts to the default ordering by case number. Another is feedback when validating user input. When invalid data is input there is a red box that appears around the input box, however, it would have been good to have a message appear informing the user of what about the input was invalid.

9: Risk Assessment

The project contains a number of risks that need to be identified. Table 6 outlines the risks that have been identified by the team and provide instructions and ideas on how the severity of these can be reduced.

Impact scale is 1 - 5:

- 5 The project is not completed at all.
- 4 The project is completed poorly and likely not on time.
- 3 The project is completed but not up to the team standard.
- 2 The project is completed with some stress.
- 1 The project is completed with some minor stress.

Likelihood 1 - 5:

- 5 It will happen
- 4 It will likely happen
- 3 It may happen
- 2 It is unlikely it will happen
- 1 It will not happen

Table 7: The risks identified and related information. Risks are ordered high - low risk (likelihood multiplied by the impact).

ID:	Description:	Impact:	Likelihood:	Responsibility:	Consequences:	Prevention:

RA-16	A team member struggles with delegated tasks.	4	4	Development team	Poor quality content is created. Deadlines potentially are not met. Potential loss of marks for the team.	Communicate any issues with current tasks with other team members. The team decides on deadlines before a current task. This will allow some extra time for other team members to help the struggling member with work.
RA-15	A team member does not meet a deadline	4	3	The development team and teaching staff.	The team either loses marks on submission or the team falls behind.	Implement soft deadlines and communicate them to the whole team. Encourage team members to reach out when they are struggling. Ensure the yellow card policy is followed. If it happens multiple times involve teaching staff.
RA-14	Inaccurate time estimations	3	4	Development team	Deadlines are not being met. Uneven workload between team members.	Set up team decided deadlines before the actual deadlines. To ensure there is extra time for any difficult tasks and time for other team members to help. Communicate any difficulties with current tasks. Have frequent meetings.

RA-13	Team members have extracurricular activities	2	5	Development team	Deadlines are not being met. Poor quality output due to time constraints. Potential loss of marks for the team.	Set up communication deadlines before the due date. To ensure that if a team member is falling behind there is time for other team members to pick up their work. Use of organisational software such as Trello and google calendar. Log any extracurricular events in the project plan.
RA-12	Team disagreement	2	5	Development team. Team leader	Productivity is reduced until a resolution is found.	Communication, review of all work. Allow both sides to present ideas then vote. The team leader may make the final call.
RA-11	Team members produce poor code.	3	3	Development team	Code becomes harder and harder to maintain. Hard to implement new features. Hard to bug fix and test. Potential loss of marks.	Create review sessions after new features are implemented. Use of peer coding. Encourage communication of any difficulties or weaknesses. Ensure that team members follow coding conventions.

RA-10	The presentation goes poorly.	3	3	Development team	Potential loss of marks for the team.	Ensure that a plan is in place before the presentation. Ensure each team member knows what they will be talking about. Ensure that only complete or near completion parts of the project are shown in the presentation/ demo.
RA-9	API becomes unavailable after implementing it in the project.	4	2	Development team	The development team will have to implement new features close to deadlines. Could lead to a potential loss of marks for the team	Ensure research is done on API or library before use in a project. The use of soft deadlines. Have a backup plan for certain features.
RA-8	The project does not meet the requirements of feature packages	4	2	Development team	Loss of marks for the team.	Ensure that the project requirements are being revisited by each team member. Ensure that a rigid plan is in place before the team begins coding. Weekly reviews of work.

RA-7	Data gets corrupted or lost.	4	2	Development team	The whole project could be potentially lost. Parts of the project could be potentially lost.	Ensure the use of GIT. Ensure team members are pushing to git after an hour of coding. Ensure all work is backed up on google drive.
RA-6	Team members don't have the same technical knowledge	2	4	Development team	Some team members have a higher output than others leading to frustration. Uneven workload between team members.	Communicate weaknesses and strengths. Create a team environment where it is ok not to know everything. Communicate to other team members when you don't understand something.
RA-5	Team member gets sick	2	4	Development team	Productivity decreases and meetings are less efficient. The team can fall behind	Team hygiene. Look after yourself. Set up online means of communication and submission. For example, zoom, google docs etc. Allow a buffer time so one person's work can be absorbed by the team.
RA-4	New Zealand going into covid alert level 2, 3 or 4	2	3	The development team and teaching staff	Team members having to work remotely from home. As a result, productivity decreases and it becomes harder to communicate with team members.	More frequent shorter team members to ensure team members stay motivated and up to date. Emphasis on team use of Trello, Slack, zoom, and other online services.

RA-3	Project requirements change during the project	3	2	Development team	Having to go through and make changes to parts of the project. A shorter period of time to complete a feature. Potential loss of marks.	Ensure that coding conventions, documentation, and unit tests are put in place to keep the code maintainable. Ensure weekly reviews of the project requirements and feature packages.
RA-2	Acceptance tests not met.	3	2	Development team	Having to make changes close to the deadline.	Ensure review of acceptance tests throughout the development phase. Ensure the use of soft deadlines.
RA-1	Feature creep	3	2	Development team	The team runs out of time to implement important features. Code becomes unmaintainable. Code becomes very hard to bug fix and test.	Plan a rough timeline of feature implementation at the start of the project. Prioritise features and look at feature dependencies. Weekly reviews of what the team would like to implement next. Implement a code freeze at milestone points.

10: Project Plan

Table 8: The dates for all known milestones and the tasks required for completion.

Milestone	Date	Description	Related Tasks
M1 - Design Document Draft #1	29th July	Design Document draft one, first sections completed	• Business and system context • Stakeholders and requirements • Acceptance tests • GUI prototypes
M2 - Design document draft #2	8th August	Design Document draft two	• Deployment model • Detailed UML class diagram • Risk assessment • Project plan • Checklist
M3 - Phase 1 hand in	9th August	Handing in the design document and project checklist for phase 1	• Whole Design Document • Project checklist
M4 - Phase 1 presentation	11th August	Prepare and present presentation for phase 1	• Presentation preparation • Present
M5 - Feature Package 1	26th August	The first feature package of the project	• Data importing • View data table • Individual crime view
M6 - Feature Package 2	7th September	The second feature package of the project	• Data sort • Comparing crimes • Search crime data
M7 - Feature Package 3	16th September	The third feature package of the project	• External web search • Delete crime • Add crime • Update crime
M8 - Feature Package 4	23rd September	The fourth feature package of the project	• Graph #1 • Database implementation • Paginator • Testing
M9 - Code freeze	23rd September	No more features implemented until after phase 2	• All feature packages 1-4 implemented or put on hold till phase 3

M10 - Phase 2 hand in	27th September	Handing in the first iteration of crime application and updated design document	● Crime application version 1 (Feature packages 1 -4) ● Updated Design document
M11 - Phase 2 presentation	28th September	Preparing and presentation the presentation/demo for phase 2	
M12 - Deadline for feature package 5 and code freeze for phase 3	7th October	The final feature package for the project. Code freeze for all feature implementation	● Database overhaul ● Crime over time graphs ● Mapping and heat map features ● Sign in system ● Testing and tidying up code
M13 - Phase 3 hand in	11th October	Handing in the final version of the crime application and final version of the design document	● Final version of application ● Updated design document
M14- Demo of final product	Week 11	Prepare a presentation and present the final version of the application	● Prepare presentation ● Present presentation

All of these milestones utilise Trello to keep track of what is to be done, what has been done, and what is being done. The above major milestones and tasks leading up to them are visualised using a Gantt chart in Figure 12. Each milestone marks a code/document freeze, a hand in point or the completion of a feature package. Also shown in the chart are team member commitments, where certain team members will be unavailable for that duration.

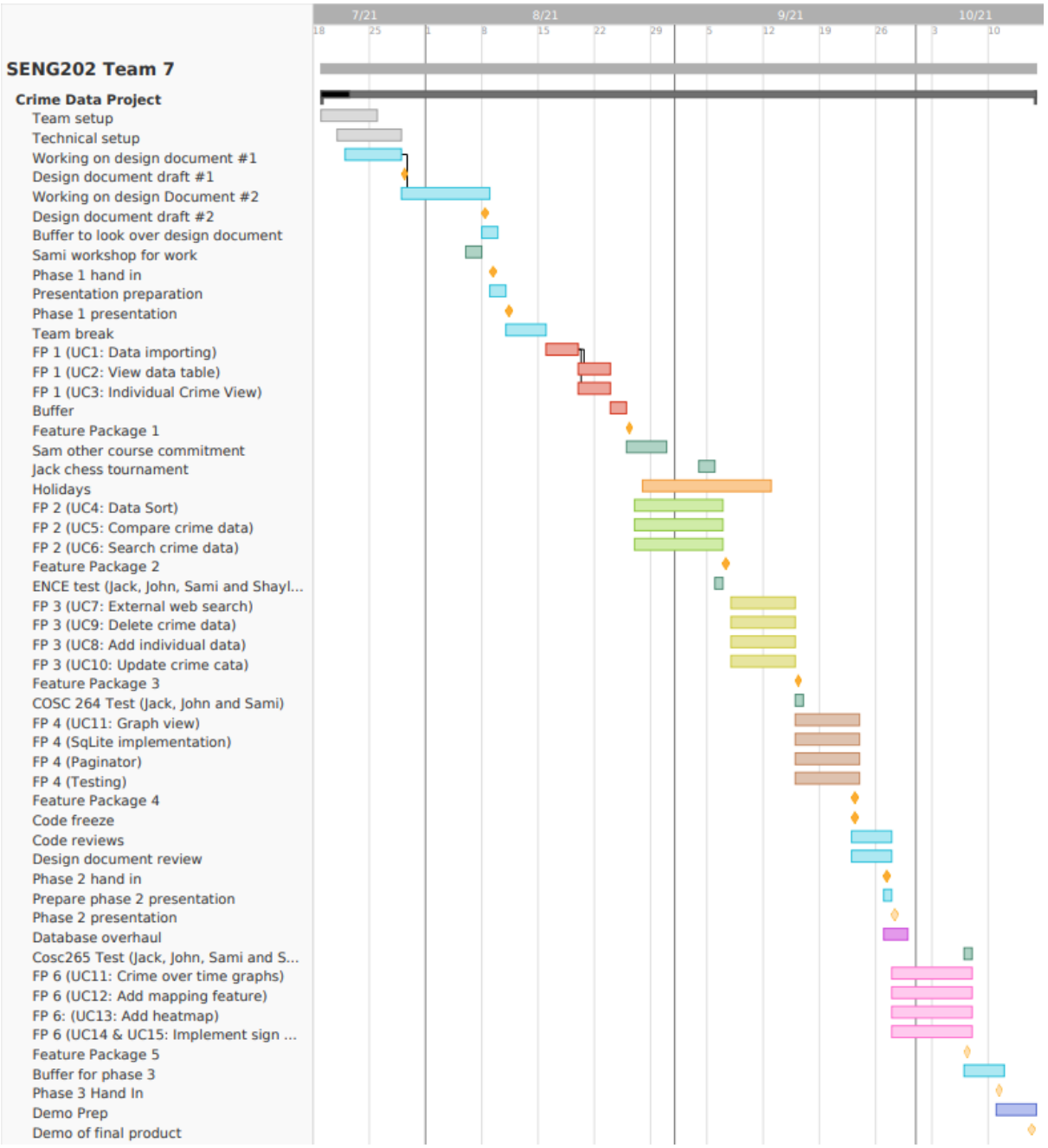

Figure 12: Gantt Chart of current project schedule.

Table 9: Team member weekly commitments

	Seng202	Jack	John	Sam	Sami	Shaylin
Monday	Lecture 16-17	17:30-19	n/a	n/a	n/a	n/a
Tuesday	Stand Up Prep:	n/a	n/a	16 - 18	19-21	n/a

	11:-12: Stand UP: 12 - 14					
Wednesday	Lab: 16-18	n/a	n/a	n/a	n/a	n/a
Thursday	Team Meeting: 12 - 14	17:30-19	n/a	16-23	n/a	n/a
Friday	n/a	n/a	n/a	n/a	17-19	n/a
Saturday	n/a	n/a	10-18	n/a	n/a	n/a
Sunday	n/a	19-21:30pm	10-18	16-23	n/a	n/a

11: Lessons Learned

Jack McCorkindale:

Deadlines and Yellow Card Policy

At the beginning of the project, we developed a set of yellow card policies revolving around deadlines. However, as we went through the project these were not/could not be upheld as much as they should have been. During meetings, we would normally spend time assigning tasks to each person to complete with a typical deadline of “before the next meeting”. If I were to do another project similar to this, I would insist on a stricter deadline system to increase individual responsibility toward the project. With the deadlines being set like this the ability to apply the yellow card policy became difficult and led to team members not completing their tasks in an acceptable time.

Continuous testing

Another decision that we made as a group at the beginning of the project was to follow testing driven development. This means that each component should have had tests written prior to the implementation for the class. However, no team member followed this scheme which resulted in testing being left toward the end of the project. This was further emphasised due to the lack of a strict yellow card policy as mentioned previously. Having completed this testing earlier would have reduced the number of problems arising when adding more components to the application and when

John Elliott:

Team tasks allocation with deadlines

During the start of the project, be it due to being a new team or inexpressive, we often slacked on properly allocating and setting deadlines for tasks. While it wasn't noticeable at the start of the project as we moved into phase-2 and phase-3 we began to feel the pressure whenever a task was not completed. We worked on in the later stages ensuring a fair delegation of tasks and setting deadlines but not having done it from the start differently affected the final project feature set.

Ensuring project time management

Another issue similar to that of task deadlines was overall time management. During this course there were weeks when I did not plan out my weekly tasks. This made it difficult for myself when trying to complete tasks before a given deadline. I worked on improving this by setting up a calendar with my schedule and ensuring I gave myself adequate time to complete the tasks I was assigned.

Doing continuous integration testing

Finally, a key thing I learnt from this course was the concepts and methodologies associated with continuous testing. Ensuring that the project remained functional during each change helped in making the application more reliable and gave us trust in its functionality. I feel that we could still have done more testing during the project as our test coverage was low for methods (about 60%).

Sam McMillan:

Overcoming the issues of working in a team of differing abilities

Although I thoroughly enjoyed working in team seven over the semester, I found the most significant difficulty was working in a group with differing ability levels. One of the issues caused by this for me personally was when conversations in meetings turned technical. Gaps in my knowledge or other team members sometimes created confusion and communication barriers, often at the cost of meeting time. Another issue I found was when going from implementing features by myself to implementing features with the team. I had to spend time going over team members' code to understand the implementation and their use of packages and tools that were not familiar to me. It often led me to guilt as it took me longer than others to implement new features into the application. However, working in a team of differing abilities was a great lesson. Our team overcame these issues with peer programming and frequent meetings. For me reaching out to a team member to explain certain parts of code was the better alternative over using google to search for answers.

Overestimating task completion

A big issue I found was the underestimation of the time needed to implement a feature in code. The team was optimistic about what we could accomplish in a given period. However, due to the nature of software engineering, where bugs and errors occur frequently, and each new feature requires a certain level of research before implementation, deadlines were often pushed back, and features were left unimplemented. I will allow for a more realistic timeline for future software projects, including extended deadlines and buffer periods. This will help to ensure a less is more approach, where code implemented is of a higher quality and allows for more time researching, testing, and writing documents.

The amount of self-learning required to be a 'good' software engineer

“seng202” covered a wide array of software engineering tools and knowledge. However, I found that I needed a lot more self-learning to be successful in this course. The labs were often the tip of the iceberg regarding what specific tools such as JavaFX and Git could do and provide. I often went back over labs and used tools such as Google and stack overflow to give clarity and more knowledge. During the implementation of a new feature, I often spent time going over what solutions had been used for similar problems. Looking ahead to future projects and, hopefully, a career in software engineering, the ability to self-teach will be paramount.

Sami Elmadani:

Power of pair programming

This project consisted highly of programming and implementing certain features for the first time. Having the ability to search for information online on the methods used to implement these features is very useful, however being able to apply this to your own project may have drawbacks. The implementation resides on good coding conventions, checking that there is not a better way to implement, and error checking. Pair programming and the use of ‘Code Together’ therefore became fundamental to my own success in this project, as it allowed a team member, who has full understanding of the project, to walk me through the process of implementation of areas I needed assistance. It also let me watch as other members

implemented their tasks off Trello, allowing me to keep up with changes to the code and how they were made.

Taking on new tasks

This came with a second fundamental lesson learned, which involved the importance of taking on a new type of task. As phase one involved solely on the design document and no coding, once phase 2 began I continued with minimal coding, due to the lack of coding experience. This became a huge drawback for the initial week of coding as I would overcome this hurdle without taking on a task I had never done before. Once I chose to finally begin programming tasks, concurrently with pair programming, I was able to create a wider understanding of the project and the feature/code implementation. This also increased my confidence and understanding in Java coding style, so it became easier to read code and therefore implement features quickly.

Immediately addressing team issues

With all of this coding related learning alongside team members, the drawback of other team members became more clear. This is a normal part of group projects, however it became clear quickly the importance of addressing these issues immediately. The team had created a yellow card policy in the first weeks of the project, and had agreed to follow through with consequences, involving the lack of effort of individuals. This became clear when we did not follow the policy strictly, causing team members, including myself, to repeat mistakes without much concern. From this I learned the ability to confront these issues directly to the team member rather than forgetting about them during meetings. By doing so the team's cooperation and efficiency would have become stronger and more concise as we were not afraid to address issues, as we should be able to.Shaylin Simadari:

Communicating with a group

One way our group tried to tackle this project was by having frequent team meetings. In meetings, I thought at first that I was communicating better by speaking less and listening more. But I've learned that the amount of attention you pay is equally as important as the amount of time you spend communicating. Not to say that I didn't pay any attention, but I realised that when speaking, it is important to understand from the other person if they are following you, and this requires a lot of concentration when a whole group is involved. Similarly, it is important to really listen so that you can understand the speaker well and have to clarify less things later, allowing the meeting to move forward smoother.

Developing with differing visions

Naturally, each person in the group started off with very different visions of how the product would turn out. Through discussion, we started to merge those into a single team vision. Having differing views is expected and it is important to build a healthy discussion around it so that the best solution can be reached. I've learned that it is important to bring up your viewpoint and the reasons for it, even if there is expected push back. It is also important to break down why you think something is a good idea and, while listening to constructive feedback, keep engaging in the discussion.

Change Log

Tabel 10: Changes that occurred within the design document since the submission of deliverable 2.

Section Number	Section Title	What changed	Changed by
11	Lessons Learned	Added key lessons learned for each member	Jack McCorkindale, Sam McMillan, Shaylin Simadari, Sami
7	Testing Procedures	Updated tests to reflect new components that have been added	Jack McCorkindale
2.2	Requirements	Updated the use case diagram and textual use cases to reflect our final product	Sam McMillan
3	Acceptance Testing	Added export and removed login features to reflect our final product	Sami Elmadani
7	Testing Procedures	Added export test cases and removed login test cases. Updated testing procedures	Sami Elmadani
6	UML Class Diagram	Updated to match new code	Shaylin Simadari
7	Testing Procedures	Updating to the tests being run in the latest version of the application	Jack McCorkindale
8	Current Product Version	Updating to represent the latest version of the application, removed newly implemented features in unimplemented features	Jack McCorkindale

References

City Of Chicago. (2019). *Gun Crimes Heat Map* [Map]. Gun Crimes Heat Map. https://cdn.vox-cdn.com/thumbor/xXDE-eOycsiiaPSSUIFwlnEzbCY=/0x0:2040x1325/fit-in/2040x1325/cdn.vox-cdn.com/uploads/chorus_asset/file/22530689/Gun_Crimes_Heat_Map.jpg

Auror - The Retail Crime & Loss Prevention Platform. (n.d.). Auror. Retrieved August 1, 2021, from https://www.auror.co/

policedata.nz. (n.d.). New Zealand Police. Retrieved August 1, 2021, from https://www.police.govt.nz/about-us/publications-statistics/data-and-statistics/policedatanz

Appendix

Each team member contributed to different sections of this document with varying levels of time. Table 7 documents which section of the document each team member contributed towards. The team members that worked on each section are then shown in the order of who did the most work in each section

Table 11: Each section of the document paired with the team members who worked on it.

Section	Team Members
Title	Jack McCorkindale
Business and system context	Sami Elmadani, Jack McCorkindale
Stakeholders and requirements	John Elliott, Sam McMillan
Acceptance Testing	Jack McCorkindale
GUI prototypes	Shaylin Simadari, Jack McCorkindale
Deployment model	John Elliott
UML class diagram	Shaylin Simadari
Testing Procedures	Sami Elmadani, Jack McCorkindale
Current Product Version	Jack McCorkindale
Risk assessment	Sam McMillan
Project Plan	Sami Elmadani, Sam McMillan
Lessons learned	Jack McCorkindale, Sami Elmadani, Sam McMillan, John Elliot, Shaylin Simadari
References	Sami Elmadani, Shaylin Simadari
Appendix	Jack McCorkindale

The UML diagram displayed in Figures 9, 10, and 11 can be viewed in totality in the attached document “seng202_2021_team7_UML_class_diagram.pdf”.