SafePoint - Design Document

Phase 3

Team 3

Danish Jahangir
Matthew Garrett
Priscilla Ishida-Foale
Rory Holmes

Contents

Change Log

Section Number	**Section Title**	**Change Made**	**Made by**
Section 2	Stakeholders and Requirements	Changed use cases to match those that were implemented in the final product Edited use case diagram	Priscilla Ishida-Foale
Section 3	Acceptance Testing	Re-added the removed section Edited some acceptance testing to match current use cases	Danish Jahangir Priscilla Ishida-Foale
Section 5	UML Diagrams	Updated UML class diagram	Matthew Garrett
Section 6	Testing Procedure	Added new Manual test cases Updated test coverage and results, added detailed explanation to classes/line/package coverage	Matthew Garrett Priscilla Ishida-Foale
Section 7	Current product version	Changed the product version details	Danish Jahangir
Section 7	Noteworthy Features	Added some of the most noteworthy features about SafePoint	Danish Jahangir, Matt Garrett
Section 9	Project Plan	Redid the Gantt chart and changed some descriptions	Rory Holmes, Danish Jahangir
Section 10	Lessons learned	Added the section and wrote about the lessons I learned	Danish Jahangir, Matthew Garrett Priscilla Ishida-Foale, Rory Holmes

Section 1 - System and Business Context

1.1 System Context

SafePoint is designed as an application targeting tourists travelling to new locations. Before travelling to new places, the tourists would be able to use SafePoint to check for information about the criminal activity of the area in a listed, map or graphical view, which would potentially allow them to make better informed decisions about whether they want to visit that place or not. Tourists can also inform themselves of what types of crimes are more prevalent in which areas.

Figure 1 illustrates the main stakeholder (tourists), interacting with the system. The application displays crime records retrieved from official public records, as well as manually uploaded records from other tourists.

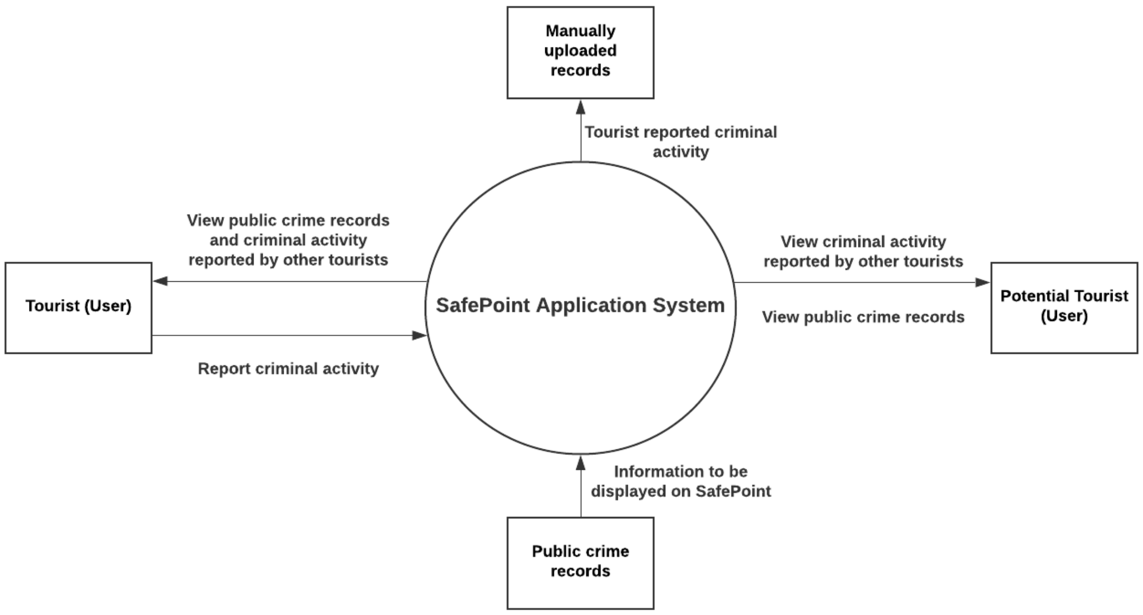

Manually uploaded records
Tourist reported criminal activity
View public crime records and criminal activity reported by other tourists
View criminal activity reported by other tourists
Tourist (User)
SafePoint Application System
View public crime records
Potential Tourist (User)
Report criminal activity
Information to be displayed on SafePoint
Public crime records

Figure 1: System Context Diagram

1.2 Business Context and Services Provided

As a local, we are aware of locations that should be avoided due to higher criminal activity. However, as a tourist, the likelihood of falling victim to crimes for venturing into unfamiliar areas is significantly increased. Given the expected tourism increase once borders reopen, it is relevant for these individuals to be able to easily identify potential criminal hot spots.

There are currently several applications available on the market which utilise a similar crime monitoring system. These include Neighbors (Neighbors by Ring, 2021) and Citizen (Citizen, 2021). The uniqueness of SafePoint stems from the fact that there is a focus on tourists' needs - considering their own personal experiences in specific areas of the city they are in.

Tourists are less likely to report crimes for several reasons (such as unfamiliarity with local laws, fear of repercussions, etc.) (Glensor & Peak, 2004). Therefore, implementing a system where users can informally report criminal occurrences (whilst also being able to view existing, official data) could help combat crime and prevent others from being affected by similar problems.

Section 2 - Stakeholders and Requirements

2.1 Stakeholders

Within this project, several stakeholders were considered as a starting point. These were identified, analysed and ordered within Table 1, according to their significance and impact on SafePoint. The “Concerns” column in Table 1 further details each stakeholder’s involvement with SafePoint.

Stakeholders were classified following a high, medium or low scale, with high priority stakeholders playing the largest role on the application given their direct interactions with the product. For this project, it was decided that these stakeholders with the highest level of interaction with the system would have higher priority. This included tourists, the development team and local police.

Medium stakeholders were defined as mainly those who would be subject to an indirect consequence from the product, such as the effect that identifying high criminal activity areas would have on local hotels.

Finally, low priority stakeholders within this project’s scale are those that will not have much influence on SafePoint, such as competitors. Given the main focus is delivering a university project, it was decided that these should not be emphasised as much, unless the product became marketable in the future.

Table 1: Stakeholder Table

ID	Stakeholder	Description	Concerns	Priority
SH_1	*Tourists*	Foreigners visiting unfamiliar locations - to begin with, anyone visiting Chicago (as this is the city that the project's database will be based off of).	As the main target audience of the application, tourists would have the benefit of identifying dangerous locations in an unknown city. The app also has to be attractive enough for the tourists to use it.	High
SH_2	*Potential tourists/citizens*	Any individual that is looking at visiting a new city, but has not done so yet and people who want to look into moving to a new place.	The application could affect their choice in visiting a location or not, depending on its criminal activity.	High
SH_3	*Teaching Staff*	SENG202 lecturers, senior tutor and tutors responsible for supervising and providing feedback for the application.	The teaching staff determines the quality and main requirements of the products. Therefore, the product must consider the SENG202 project requirements as its constraints.	High
SH_4	*Development Team*	Software Engineering students working on creating the application.	A successful product created by the development team will produce good grades. The product has to fit within the timeline set by the development team, and be within the capabilities of the team's skill set.	High
SH_5	*Local Police*	The police within the locations that the app will contain information on - to begin with, the Chicago Police Department.	Local police would be interested in identifying potential criminal hot spots which go undetected, given the lessened likelihood of tourists reporting crimes (Glensor & Peak, 2004).	High
SH_6	*Choose Chicago - Tourism Board*	The official destination marketing organisation for the city of Chicago.	The board could benefit from the application by getting more tourist traffic given the additional safety measure	Medium

			the app provides. Furthermore, it could affect what regions within the application require data (suburbs with most tourist attractions, for example).	
SH_7	*Local Hotels*	Local hotels in Chicago, displayed on the application map for the region chosen.	Local hotels could either benefit or suffer from the application's use depending on whether they are in higher or lower criminal activity areas. This is particularly relevant as they are displayed on the map through the web search feature.	Medium
SH_8	*Local Retail*	Local retailers in Chicago.	Local shops could either benefit or suffer from the application's use depending on whether they are in higher or lower criminal activity areas, although these will not be displayed on the map, unlike the hotels.	Medium
SH_9	*Google*	Company that specialises in Internet-related services and products - in this case, the developer of the Google Maps Platform API.	Google could affect our application given any change implemented to the Google Maps Platform API.	Medium
SH_10	*Competitors*	Other similar applications such as Neighbors (Neighbors by Ring, 2021) and Citizen (Citizen, 2021), which involve a community-watch based crime monitoring system.	Our application must somewhat be up to date with what is already in the market, whilst ensuring that it still has a unique selling point. Other applications would be concerned about the similar target markets that SafePoint is aiming for.	Low
SH_12	*University of Canterbury*	The University in which the development team studies Software Engineering.	UC establishes constraints on our product as it is considered a university project as students are required to adhere to the UC Code of Conduct.	Low

2.2 Use Cases

2.2.1 Use Case Diagram

The stakeholders identified in Table 1, along with the system context diagram (Figure 1) were utilised to identify actors for the UML use case diagram. Use cases were determined according to the concerns of each stakeholder - although some detail was omitted on this diagram and explored in further depth in Table 2.

Some use cases were altered/removed during the planning process of the application. An example of this is how originally there was intention to search for background information of a specific region on the map. This was substituted by being able to view hotels in the area, as it was deemed more relevant to high priority stakeholders (such as tourists and potential tourists/citizens). However, due to time constraints for the final iteration, this final feature was also removed. The diagram below illustrates the use cases included in our final deliverable.

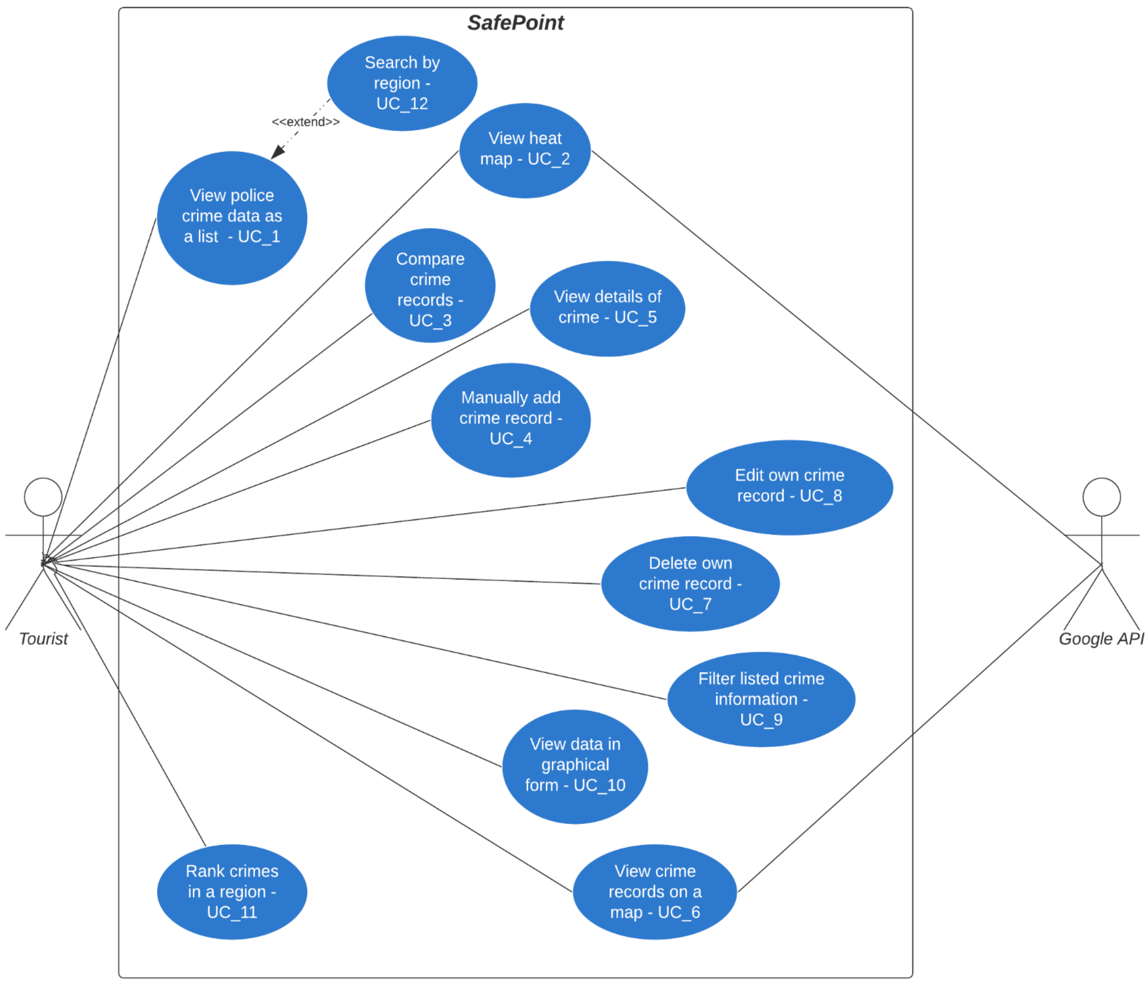

SafePoint
Search by region - UC_12
<<extend>>
View heat map - UC_2
View police crime data as a list - UC_1
Compare crime records - UC_3
View details of crime - UC_5
Manually add crime record - UC_4
Edit own crime record - UC_8
Delete own crime record - UC_7
Filter listed crime information - UC_9
View data in graphical form - UC_10
Rank crimes in a region - UC_11
View crime records on a map - UC_6
Tourist
Google API

Figure 2: Use Case Diagram

Table 2: Textual Description of Use Cases

UC_1	**View crime data**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. Crimes are automatically loaded onto the main display screen.	1. Click on the reload data button if crimes are not visible.	N/A	N/A
Postcondition			
The main viewing screen will display a list of crime records on the sidebar, along with a map of Chicago on the right side, with markers indicating crime records.	The main viewing screen will display a list of crime records on the sidebar, along with a map of Chicago on the right side, with markers indicating crime records.	N/A	N/A

UC_2	**View heat map**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. User ticks a checkbox for the heat map, located on the left sidebar.	N/A	N/A	N/A
Postcondition			
A map of Chicago on the right	N/A	N/A	N/A

side will be displayed in the form of a heat map, with clusters instead of individual markings.			

UC_3	**Compare crime data**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. User accesses the desktop application. Crimes have been loaded onto the display. 2. User ticks a checkbox for comparing crimes, located on the left sidebar. 3. User selects two crime records from the list of crimes.	N/A	(Continues from step 2 of basic flow) 1. User selects the same crime twice from the list of crimes.	N/A
Postcondition			
A pop up window will be displayed with the information regarding these two crimes, and additional information derived from them (distance between crimes, time between crimes, etc.)	N/A	No pop up window will be displayed as a crime cannot be compared to itself.	N/A

UC_4	**Manually add crime data**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. Users select to report crime. 2. Users are prompted to enter crime information through a pop-up screen (crime date, location, and type of crime). 3. Users are given a unique ID to associate them with an uploaded crime.	N/A	1. Users select to report crime. 2. Users are prompted to enter crime information through a pop-up screen (crime date, location, and type of crime). 3. User inputs invalid data. 4. Text is displayed requiring users to input data again.	N/A
Postcondition			
A new marking of a tourist-reported crime is displayed on the map, along with a new entry on the list of crimes on the left sidebar.	N/A	The information displayed remains the same, and a new marking of a tourist-reported crime or a new entry on the list is not created.	N/A

UC_5	**View details of a crime record**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow		**Alternative Flow**	
1. User selects to view details of a crime record on the list of crimes on the left sidebar.		1. User hovers over a marker on the map to view additional details.	

A pop up window will be displayed with the details about this specific crime record (such as date and time, location, crime type, ward, longitude and latitude)		An info window or title is shown above the marker, with some summarised information about the crime (crime ID and crime type).	

UC_6	View crime records on a map		Actor: Tourist
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. User accesses the desktop application.	N/A	N/A	N/A
Postcondition			
A map is displayed on the right hand side, with markings representing crime records.	N/A	N/A	N/A

UC_7	Delete user reported crime record		Actor: Tourist
Precondition: User reported crime exists and the user is on the main viewing screen, with the list of crimes and map being displayed.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. User finds user reported crime	N/A	(Continues from step 4 on basic	N/A

in crime data list. 2. User selects to view details of user reported crime (see **UC_5**). 3. User selects the delete button. 4. Users are prompted to enter their unique code. 5. User enters valid code.		flow) 1. User enters an invalid code. 2. User is informed an invalid code has been entered. 3. Users are prompted to enter code again or exit.	
Postcondition			
Selected crime records are no longer displayed on a map or list.	N/A	Selected crime records will still be displayed on map and list.	N/A

UC_8	**Edit user reported crime record**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	**Exceptional Flow**
1. User finds user reported crime in crime data display. 2. User selects the edit button. 3. Users are prompted to enter their unique code. 4. User enters valid code. 5. User inputs data from the Crime Report window (Figure 4). 6. Select the report crime button to update the crime record.	N/A	(Continues from step 4 in basic flow) 1. User inputs invalid data (such as a wrong address, or an empty input) 2. Information box is shown requiring the user to input data again.	(Continues from step 3 in basic flow) 1. User inputs invalid code. 2. Text is displayed requiring the user to input the code again or exit.
Postcondition			

Crime record changed to new data.		Crime record remains the same. User is now viewing the additional information box on top of the main view.	Crime record remains the same. User is now viewing the additional information box on top of the main view.

UC_9	**Filter crime information**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. User accesses the desktop application. 2. User selects the filter(s) from the left sidebar.	N/A	N/A	N/A
Postcondition			
Only the filtered crime data is displayed on the crime data list and map.	N/A	N/A	N/A

UC_10	**View data in graphical form**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. User selects Graph View from Figure 3. 2. User chooses between filtering	1. User chooses between filtering only certain crimes, or locations, or both using the settings panel	N/A	N/A

only certain crimes, locations, or both using the left sidebar.	2. User selects the Graph View option on the sidebar.		
Postcondition			
The map screen is replaced with a graph of the selected information from the user over time.	The map screen is replaced with a graph of the selected information from the user over time.	N/A	N/A

UC_11	**Rank crimes in a region**		**Actor: Tourist**
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. User selects “Show region data” on the left sidebar. 2. User selects an option from the dropdown menu (highest frequency, lowest frequency, high risk areas, low risk areas).	N/A	N/A	N/A
Postcondition			
The list of crimes will now display the crimes according to each key. For example, with high frequency, the crimes which occur more frequently will be displayed first.	N/A	N/A	N/A

UC_12	Search by region		Actor: Tourist
Precondition: The user accesses the desktop application, and is on the main viewing screen. Crimes are loaded on the display.			
Basic Flow	**Alternative Flow**	**Exceptional Flow**	
1. Users input a region from the “Search by region” search bar.	N/A	1. Users input an invalid region from the “Search by region” search bar. 2. Application informs input is invalid.	N/A
Postcondition			
The crimes within that area will be displayed on the list of crimes and the map.	N/A	Crimes will not be displayed/no change occurs.	N/A

2.3 Requirements

2.3.1 Quality Requirements

Table 3 includes the quality requirements of the system. These were identified based on the ideal experience that an user would have with the application, and what expectations they may already have. In order to verify their implementation in our application, we applied the use of concrete values to measure these against the final product. This can be seen, for example, when considering an optimal number of consecutive clicks for a user to achieve their goal in the usability section (QR_1). For this project, we divided our quality requirements by level of priority - utilising those highest ranked to develop the key drivers table (Table 4).

Table 3: Quality Requirements Table

ID	Description	Stakeholder	Priority
QR_1	**Usability.** The application should be designed to cater to users of a range of ages and backgrounds. The tourists should be able to easily view, identify and report crimes. The application should not be unnecessarily complex to use and should be designed in a way such that users do not feel overwhelmed when using it for the first time.	Tourist	High
QR_2	**Reliability.** The user should be able to feel that there are no major issues in the application and that if any bugs or crashes occur, then they are handled well in such a way that the user does not feel troubled. For example, if the application suddenly crashes for some reason, then the developers need to make sure to solve the issue as soon as possible while also notifying the user about the issue and assuring them that steps are being taken to mitigate it.	Tourists, Developing team	High
QR_3	**Efficiency.** The application should be able to perform the functions intended without any issues and not take too much time when performing them. For example, when a user tries to perform any action, the result produced by the action should	Tourists, Developing team	High

	not take more than a couple of seconds to be displayed which can be achieved by properly managing the time complexity of the code.		
QR_4	**Legality.** The application should not copy any source code from copyrighted sources and should follow all the guidelines of the UC Code of Conduct.	University of Canterbury, Developing team	High
QR_5	**Security.** Normal users would not be able to add/update the existing database; this would be reserved for the developer of the application and what they wish to display, they can only report a crime and edit the crime reported by them.	Developing Team	Medium
QR_6	**Maintainability.** Updating the application should not be costly in terms of time and resources; and any updates built on top of an application already in use should not change data uploaded by the tourists.	Developing Team, Tourists	Medium
QR_7	**Compatibility.** The application should be compatible with Windows, Linux and Mac operating systems- whichever one the tourist would prefer to use.	Tourists	Medium
QR_8	**Availability**. The application should have more than 99% uptime and should any bugs or any other issues occur they should be able to be solved as soon as possible.	Developing team	Medium
QR_9	**Consistency.** The application should be consistent in its design and structure. For example, the exit button’s position in one screen should not be different from its position in another.	Developing Team	Low
QR_10	**Scalability.** The application should be able to handle an influx of a great amount of data or users without crashing or causing any other issues. For example, if a large amount of data is imported from the database, it should not cause the application to crash nor should it lag.	Developing team	Low

2.3.2 Key Drivers

Key drivers are the most important quality attributes of a project. If there is ever a case that the project has to be delivered post haste, due to issues like time restraints for example, these are the quality attributes that are prioritized to be in the project over all things because they are the ones that the stakeholders who hold the most influence over the project will look for first. Within this project, it was found that usability and reliability are the most relevant quality requirements - being particularly significant to the tourists and potential citizens/tourists. The weighting system was chosen according to the priority of the stakeholder and the degree to which the application influences them. Tourists, the development team, the teaching staff and potential citizens/tourists had higher weightings due to this criteria. Having our main end-users (tourists and potential citizens) along with the acknowledgement of this being a

As our application would not realistically affect local police in the context of a university project, they were placed at a lower weighting on the key drivers table.

Table 4: Key Drivers Table

Stakeholders	**Weight**	**Usability (QR_1)**	**Reliability (QR_2)**	**Efficiency (QR_3)**	**Legality (QR_4)**	**Security (QR_5)**
Tourists	0.25	30	30	20	5	15
Development Team	0.20	20	25	20	15	20
Teaching Staff	0.20	20	20	20	20	20
Local Police	0.15	20	30	20	5	25
Potential citizens/tourists	0.20	30	30	20	5	15
Total	**1**	**25**	**27.75**	**20**	**9**	**18.25**

Section 3 - Acceptance Tests

Acceptance tests are formal descriptions of the behaviour of a program, giving strict parameters to the performance of the application. Criticality is the indication of how important it is that the mentioned test passes for the functionality of the application. The criticality of each test is given a number from 1-10, with 10 being essential for the performance of the application and 0 being completely unnecessary. In the following table (Table 5), we have assessed a number of different situations and how the application should perform in those particular situations. Based on the result of whether the acceptance criteria is met, it is decided whether the test passes or fails.

Table 5: Acceptance Test Table

ID	Description	Acceptance Criteria	Criticality (0-10)	Use Case
AT_1	Given a user has opened the application and any optional filtering has been selected (crime type, time, etc.). When the ‘Reload data’ button is selected then the crime data displayed on the sidebar and map is the correct data (with filters applied).	Crime data is filtered to show only crimes reported within the regions specified.	Being able to display the crime data in list form and having a working filter is critical to the usability of the program, as without this feature the application would be too slow to use comfortably. (10)	UC_1: ‘View crime data’
AT_2	Given a user accesses the desktop application and the map is displayed on the screen, when the heatmap checkbox is ticked then the map is displayed in a hotspot format with clusters instead of individual points.	When the heatmap checkbox is ticked the map displays clusters of crimes instead of individual points, with hotter colors such as red being high crime rate areas and colder colors such as green and blue being areas of low crime.	This functionality allows users to get a clearer image of where crimes are taking place, allowing the user to avoid high crime rate areas. (7)	UC_2: ‘View heat map’
AT_3	Given a user ticks a checkbox for comparing crimes on the left sidebar, when the user selects two crimes then an information box is shown	An information box is shown with the correct crimes to be compared as well as correctly calculated comparisons such as the time and	Comparing two crimes is important functionality to have as it allows users to better grasp the information shown.	UC_3: ‘Compare crime data’

	comparing the data of the two crimes, including the distance and time between crimes.	distance is shown.	(7)	
AT_4	Given a user is on the home page (with the map showing) when the user clicks the 'Report a crime' button then the application window opens to allow data input before returning to the home page after adding the data to the database.	'Report a crime' application window opens allowing correct input of crime data, and displaying an error message when given incorrect information. After given correct information gives the user a unique code representing the crime.	Reporting a crime is a stable requirement for the application, therefore it is critical this works. (9)	UC_4: 'Manually add crime data'
AT_5	Given the user is on the desktop application, when the user selects the view button under a crime record then an information box is shown with the details of the selected crime record.	An information box is shown revealing the specific crime data related to the crime selected, including information such as the location, date and time of the crime as well as whether an arrest was made and if the crime was domestic.	Viewing a specific crime is simple functionality that must be able to work to present accurate data to the user, without this functionality the user is unable to properly analyse crimes, which drastically decreases the value of the application. (9)	UC_5: 'View details of a crime record'
AT_6	Given the user accesses the desktop application, when the graph view is not selected then a map showing crimes in a visual format should be displayed with each crime being in the correct location.	A map is shown with each crime representing a point on the map, each crime should be correctly represented, in the right location, and the right colour depending on if the crime is user reported or a public record.	Giving a visual representation of crime locations allows the user a much easier experience when looking at crimes within a specific location, drastically increasing the usability of the program. (9)	UC_6: 'View crime records on a map'
AT_7	Given a user reported crime exists and is open in the 'view crime data' window, when the user selects the delete button and enters their unique	When the delete button is selected in the 'view crime data' window and the correct unique code is entered, otherwise displaying an	Being able to delete a crime the user entered allows for human error without irreversible consequences and	UC_7: 'Delete user reported crime record'

	code, the crime record is no longer displayed on the map and list.	error message, the crime record is deleted and can no longer be viewed in any form.	avoids clusters of duplicate crimes. (8)	
AT_8	Given a user reported crime exists and ‘view crime data’ window is open, when the user selects edit crime and enters the correct unique code for the crime, otherwise displaying an error message, allows the user to edit the crime information as long as the new data is in correct format.	When the edit crime button is selected in the ‘view crime data’ window and the correct unique code is entered, otherwise displaying an error message, allows the user to edit the crime record as long as the newly entered data is in the correct format, otherwise displays an error message.	While this functionality is not critical to the program as the same result can be achieved through deleting the crime and then re-entering it, it allows for a smoother and easier process so the user does not become frustrated. (7)	UC_8: ‘Edit user reported crime record’
AT_9	Given a user accesses the desktop application, when the user selects one or multiple filters on the left settings panel then the listed crime data is changed to display only crimes that match the filter(s) selected.	When any filters are selected the listed crime data is filtered to show crimes only matching the filters selected.	Filtering data is an essential way to navigate the large amounts of data to find specific crimes and is critical to the functionality of the application. (8)	UC_9: ‘Filter Crime information’
AT_10	Given user reported crimes exist and the list of crimes are displayed on the screen when the user selects the graph view button and optionally selects filters then the map screen is replaced with a graph of the information.	When the user selects the graph view the map is replaced with a graph view of the data, optionally if a filter is selected in either crime type or location then only the data within those bounds are shown on the graph.	This functionality gives another method for the user to view information, especially the use of tracking crimes over a period of time, allowing users to get a clear image of when types of crimes occured. (7)	UC_10: ‘View data in Graphical form’
AT_11	Given the user accesses the SafePoint application and the map is being displayed, when the user selects “Show region data” on the	The list of crimes to the left of the map on the sidebar is filtered and sorted according to the user's selection.	This functionality is important to the usability of the program, and would make finding specific information much	UC_12: ‘Rank crimes in a region

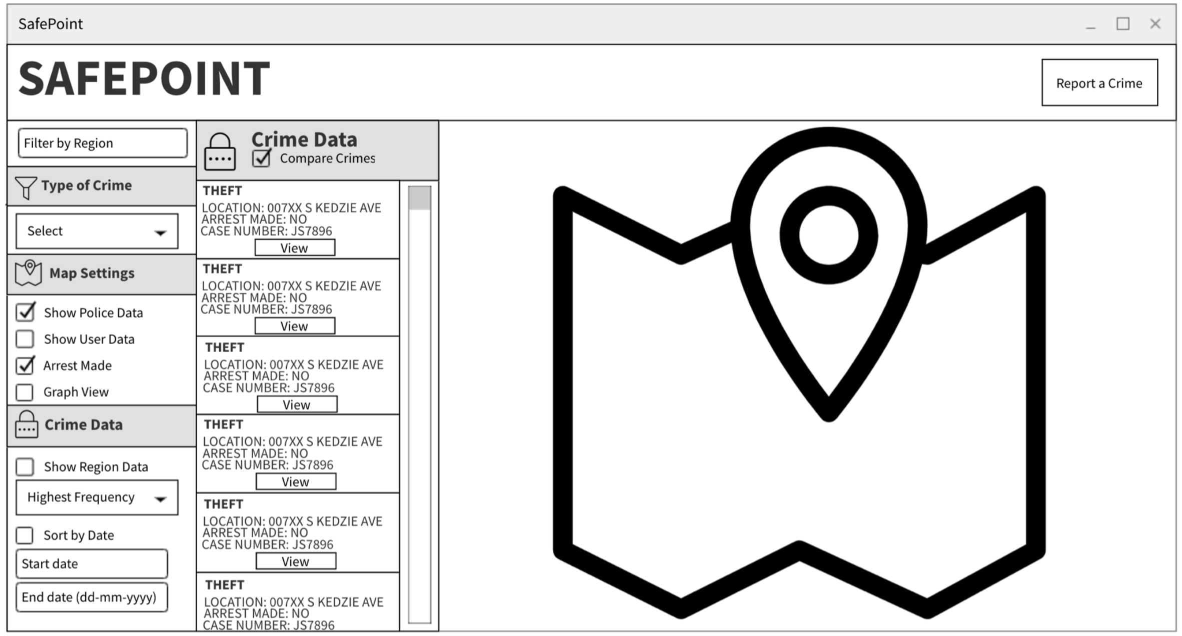
	left sidebar and the user selects an option from the dropdown menu (highest frequency, lowest frequency, most dangerous, etc.). Then the list of crimes will now display the suburbs in the city according to the criteria of the user’s choice.		easier for the user, encouraging them to use the application more often (7)	
AT_12	Given the user has accessed the SafePoint application and the map is being displayed when the user enters the region from the “Filter by region” search bar. Then the region of choice is now the main focus of the map, displaying the crimes within that area on the list of crimes and the map.	When the user has the map displayed and a region filter has been inputted then the map is altered to show the selected region and the displayed crimes in list form are shown in the sidebar	This is important functionality as it allows the user to easily navigate between specific regions and makes the user much more able to sort through large amounts of data. (8)	UC_13: “Search by region”

Section 4 - GUI Prototypes

Creating GUI prototypes helps to bring the requirements of the system into focus, ensures that we all are aware of how the app will function, and shows the physical actions of the described use cases. The current GUI prototype was constructed along with the UML class diagram (Figure 8), as we need to ensure all features of the GUI are able to be implemented in our program, and that we don't have redundant classes in the UML class diagram. Note that our current GUI prototype consists of 4 separate windows, and more than one may be open at any time. (All data shown in GUI prototypes is dummy data)

4.1 Main Application Window (GUI1):

Figure 3: Main Application Window for GUI



☑ Compare Crimes

☑ Show Police Data
☐ Show User Data
☑ Arrest Made
☐ Graph View

☐ Show Region Data
☐ Sort by Date

THEFT
LOCATION: 007XX S KEDZIE AVE
ARREST MADE: NO
CASE NUMBER: JS7896

THEFT
LOCATION: 007XX S KEDZIE AVE
ARREST MADE: NO
CASE NUMBER: JS7896

THEFT
LOCATION: 007XX S KEDZIE AVE
ARREST MADE: NO
CASE NUMBER: JS7896

THEFT
LOCATION: 007XX S KEDZIE AVE
ARREST MADE: NO
CASE NUMBER: JS7896

THEFT
LOCATION: 007XX S KEDZIE AVE
ARREST MADE: NO
CASE NUMBER: JS7896

THEFT
LOCATION: 007XX S KEDZIE AVE
ARREST MADE: NO
CASE NUMBER: JS7896

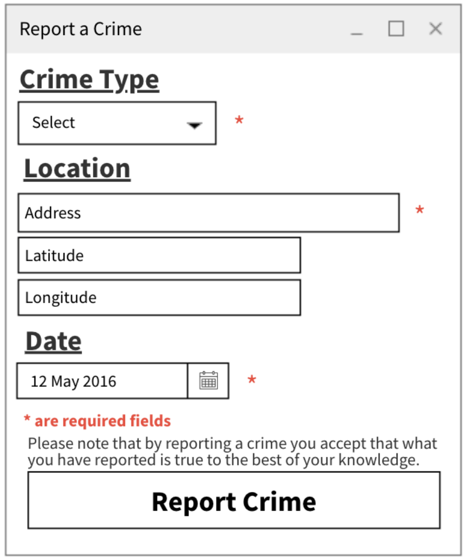
Shown above is the GUI prototype for the main window of SafePoint. The button on the top right of the screen opens up the Report Crime window (Figure 4), shown below, related to UC_4. On the right side of the window are the filter controls (UC_9), settings for the map, and settings for how to view the data, displayed both in a listed and map form (UC_1 and UC_6). This includes filtering by region, which is a dropdown menu; filtering by one or more types of crime, also implemented as a dropdown menu; and some map settings. These affect the crimes shown in the crime data panel, as well as what is being shown on the map. The map API being used is the Google Maps API.

Down the bottom of the settings panel, there are configurations to adjust the crime list on the crime data panel, which will allow for users to view the information for most/least dangerous areas, crime frequency, and date filtering (UC_12). These settings do not change the information displayed in the map screen, but will sort the list of crime records on the left.

Finally, the crime data panel shows the information for all the crimes that fit what the user has chosen in the settings/filters panel. Each one (implemented as a CrimeData object) has a view button component, which opens up a separate window to view the information more clearly (see View Crime Data window below (Figure 5)), or if the compare crimes button at the top is selected, waits for two crimes to be selected before showing a more comprehensive Compare Crime Data window (Figure 6 also shown below).

Not shown on the GUI is the ability to remove or edit user reported crimes. This is achieved by users by pressing an "X" button on the crime data they want to remove, or edit button. Pressing the edit button will open up the window on Figure 4 to re-input data for that crime data. In order to remove or modify these crimes the user will need to input the unique ID number they received when reporting the crime. This means only users that reported the crime or had permission from the user who created the crime can remove this data.

4.2 Report Crime Window (GUI2):


Report a Crime

Crime Type

Select *

Location

Address *

Latitude

Longitude

Date

12 May 2016 *

*** are required fields**

Please note that by reporting a crime you accept that what you have reported is true to the best of your knowledge.

Report Crime

This is the Report Crime window, which will open up as a separate window when the user clicks the report crime button shown in the Main Application window (Figure 3). The user will be able to select the type of crime that they experienced from a dropdown menu. Users will only be able to select one type of crime to keep this consistent with the police data. They will also need to input a valid address which will be checked for validity when the report crime button is clicked.

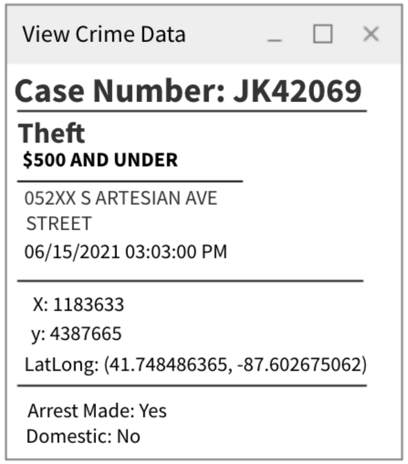
The latitude and longitude inputs are not required as they can be retrieved just from the address, but serve as a "safety net" in case the user puts the wrong address in. The date is also a required field.

Figure 4: Report Crime Window for GUI

This window has very few input fields to be more friendly for new users of the system that want to input data. The assumption is users that are reporting a crime didn't file a police report, and thus no arrest or case number for the incident.

When valid input is given and the report crime button is pressed the window will close. The user will then be given a unique ID number for this data which will be used for modifying or deleting this unique CrimeData object. This stops anyone from deleting any information that they did not create.

4.3 View Crime Data Window (GUI3):



View Crime Data

Case Number: JK42069

Theft

$500 AND UNDER

052XX S ARTESIAN AVE
STREET
06/15/2021 03:03:00 PM

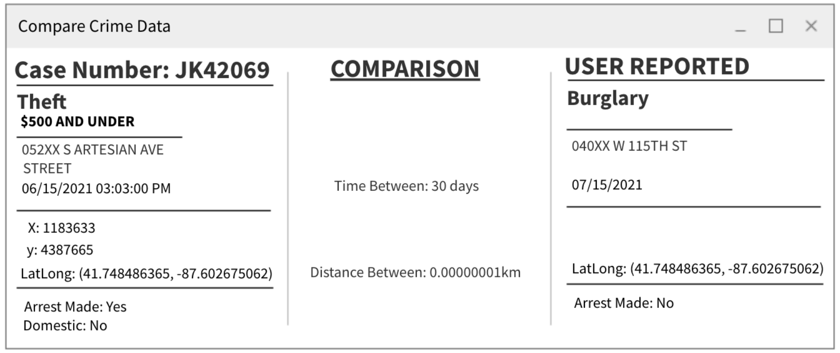
X: 1183633
y: 4387665
LatLong: (41.748486365, -87.602675062)

Arrest Made: Yes
Domestic: No

The View Crime Data window is what opens up when a crime is selected to be viewed more closely. The view on the right shows the view of police data, the Compare Crime Data window (Figure 6) has an example that shows the user reported data. If there is information missing then the lines are left blank.

Figure 5: View Crime Data Window for GUI

4.4 Compare Crime Data window (GUI4):



Compare Crime Data

Case Number: JK42069

Theft

$500 AND UNDER

052XX S ARTESIAN AVE
STREET
06/15/2021 03:03:00 PM

X: 1183633
y: 4387665
LatLong: (41.748486365, -87.602675062)

Arrest Made: Yes
Domestic: No

COMPARISON

Time Between: 30 days

Distance Between: 0.00000001km

USER REPORTED

Burglary

040XX W 115TH ST

07/15/2021

LatLong: (41.748486365, -87.602675062)

Arrest Made: No

Figure 6: Compare Crime Data Window for GUI

The Compare Crime Data window is what opens when the compare button in the Main Application window (Figure 3) is selected and two crimes are viewed. On the left is police data, and on the right is an example of user inputted crime data. All data that would be visible in the View Crime window (Figure 5) is shown, as well as some extra information to help compare the two crimes. The difference in time of the crimes (in days) is shown, as well as the distance between them (in kilometers).

Section 5 - UML Diagrams

5.1 UML Deployment Diagram

The UML deployment diagram shows how SafePoint will work with both hardware and software, and how the deployment of the program will work, including its interaction with databases.

The main program (Safepoint.jar) is contained within the JRE (Java Runtime Environment), which is in the users operating system, which could be Windows, Linux, or OSX, contained within the users device. To access the crime data, a database containing all the crime information is accessed. As we are using Google Maps API for displaying the map portion of the application, we also have to access the google maps web server when running the program.

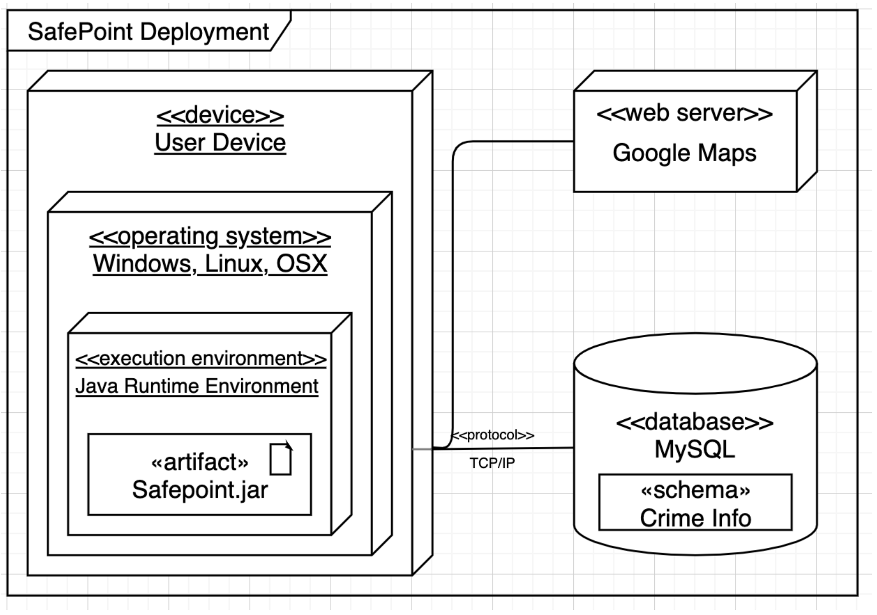
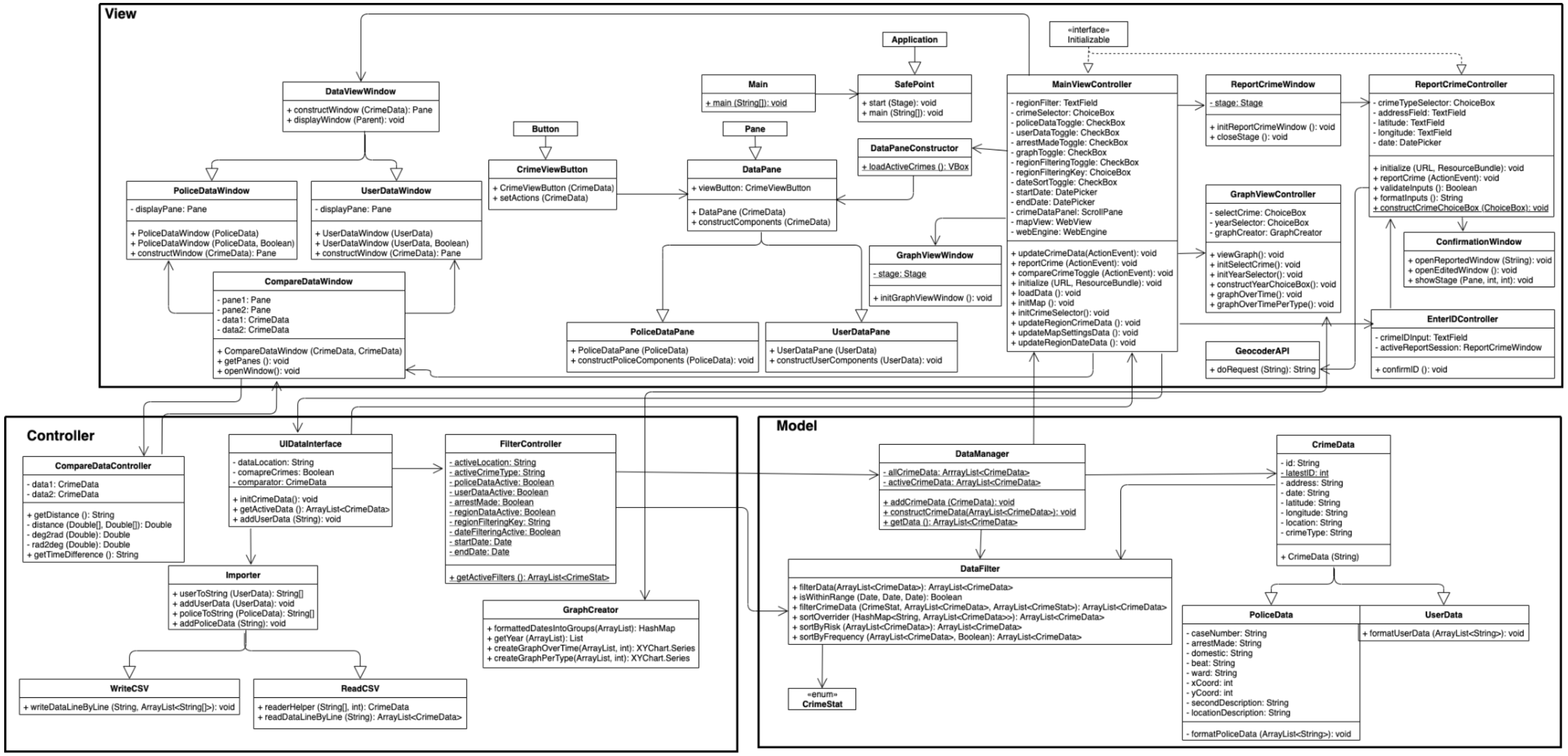
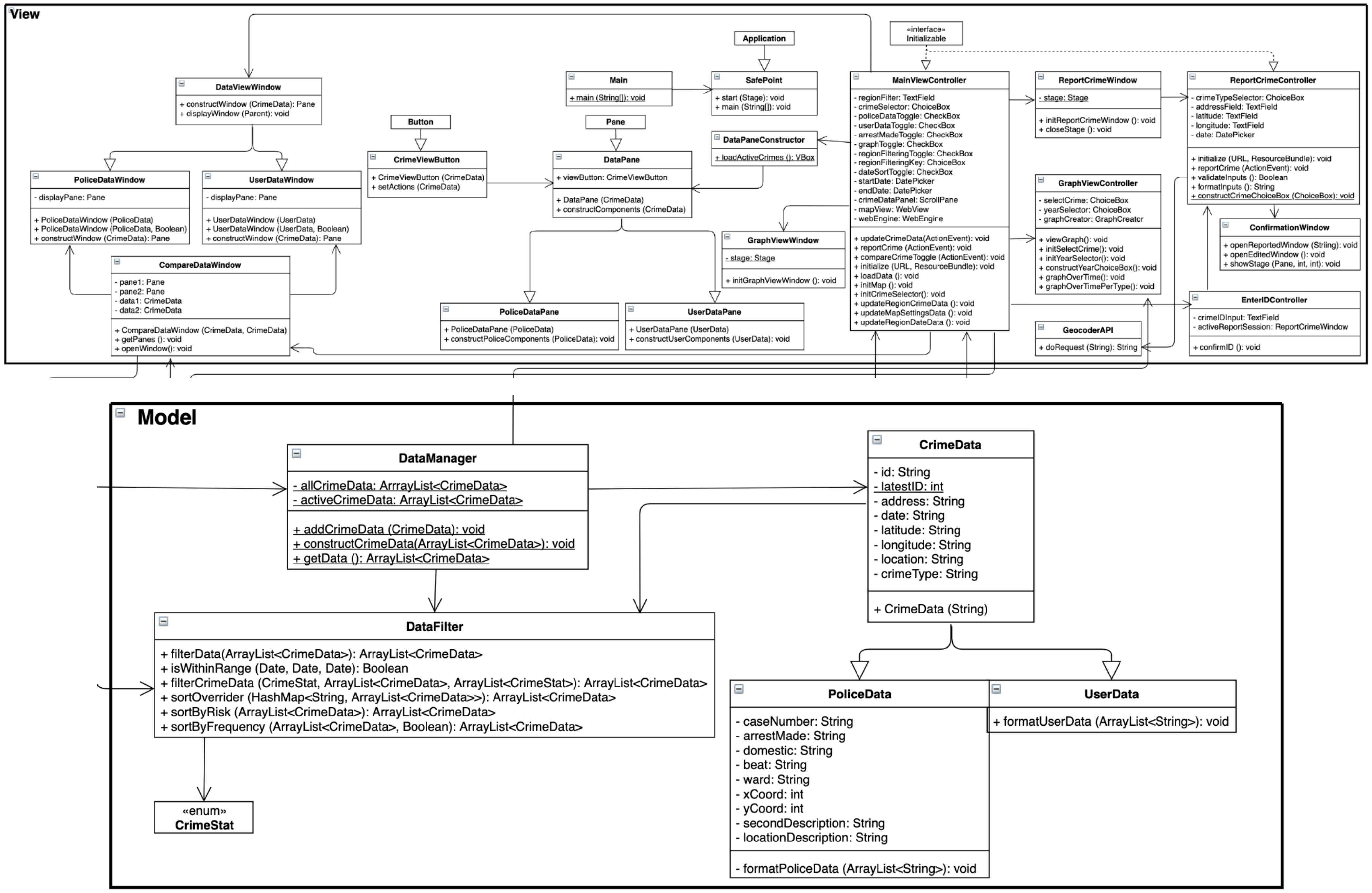
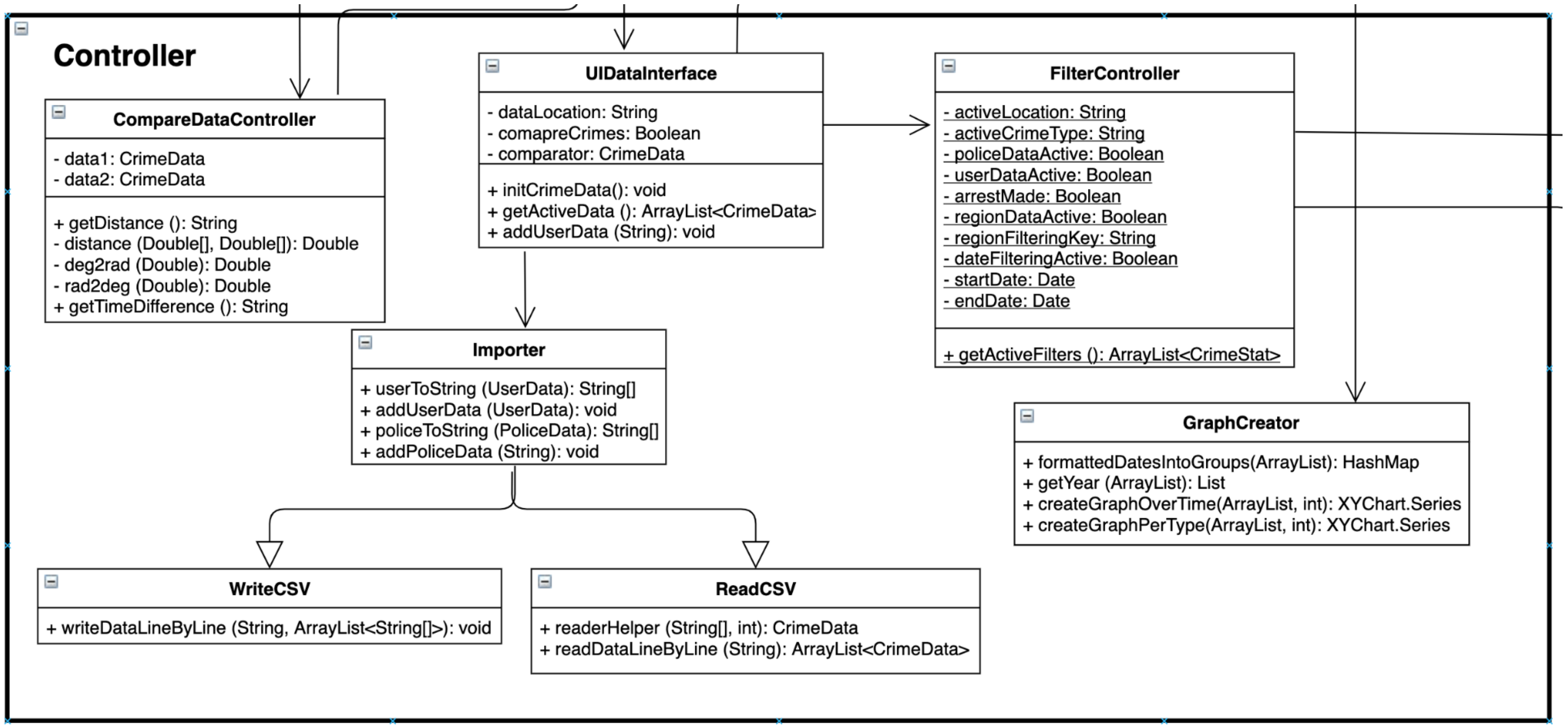


Figure 7: UML Deployment Diagram for SafePoint

5.2 UML Class Diagram


View
DataViewWindow
+ constructWindow (CrimeData): Pane
+ displayWindow (Parent): void
PoliceDataWindow
- displayPane: Pane
+ PoliceDataWindow (PoliceData)
+ PoliceDataWindow (PoliceData, Boolean)
+ constructWindow (CrimeData): Pane
UserDataWindow
- displayPane: Pane
+ UserDataWindow (UserData)
+ UserDataWindow (UserData, Boolean)
+ constructWindow (CrimeData): Pane
CompareDataWindow
- pane1: Pane
- pane2: Pane
- data1: CrimeData
- data2: CrimeData
+ CompareDataWindow (CrimeData, CrimeData)
+ getPanes (): void
+ openWindow(): void
Button
CrimeViewButton
+ CrimeViewButton (CrimeData)
+ setActions (CrimeData)
Main
+ main (String[]): void
Application
SafePoint
+ start (Stage): void
+ main (String[]): void
Pane
DataPane
+ viewButton: CrimeViewButton
+ DataPane (CrimeData)
+ constructComponents (CrimeData)
DataPaneConstructor
+ loadActiveCrimes (): VBox
GraphViewWindow
- stage: Stage
+ initGraphViewWindow (): void
PoliceDataPane
+ PoliceDataPane (PoliceData)
+ constructPoliceComponents (PoliceData): void
UserDataPane
+ UserDataPane (UserData)
+ constructUserComponents (UserData): void
«interface»
Initializable
MainViewController
- regionFilter: TextField
- crimeSelector: ChoiceBox
- policeDataToggle: CheckBox
- userDataToggle: CheckBox
- arrestMadeToggle: CheckBox
- graphToggle: CheckBox
- regionFilteringToggle: CheckBox
- regionFilteringKey: ChoiceBox
- dateSortToggle: CheckBox
- startDate: DatePicker
- endDate: DatePicker
- crimeDataPanel: ScrollPane
- mapView: WebView
- webEngine: WebEngine
+ updateCrimeData(ActionEvent): void
+ reportCrime (ActionEvent): void
+ compareCrimeToggle (ActionEvent): void
+ initialize (URL, ResourceBundle): void
+ loadData (): void
+ initMap (): void
+ initCrimeSelector(): void
+ updateRegionCrimeData (): void
+ updateMapSettingsData (): void
+ updateRegionDateData (): void
ReportCrimeWindow
- stage: Stage
+ initReportCrimeWindow (): void
+ closeStage (): void
ReportCrimeController
- crimeTypeSelector: ChoiceBox
- addressField: TextField
- latitude: TextField
- longitude: TextField
- date: DatePicker
+ initialize (URL, ResourceBundle): void
+ reportCrime (ActionEvent): void
+ validateInputs (): Boolean
+ formatInputs (): String
+ constructCrimeChoiceBox (ChoiceBox): void
GraphViewController
- selectCrime: ChoiceBox
- yearSelector: ChoiceBox
- graphCreator: GraphCreator
+ viewGraph(): void
+ initSelectCrime(): void
+ initYearSelector(): void
+ constructYearChoiceBox(): void
+ graphOverTime(): void
+ graphOverTimePerType(): void
ConfirmationWindow
+ openReportedWindow (Striing): void
+ openEditedWindow (): void
+ showStage (Pane, int, int): void
EnterIDController
- crimeIDInput: TextField
- activeReportSession: ReportCrimeWindow
+ confirmID (): void
GeocoderAPI
+ doRequest (String): String
Controller
CompareDataController
- data1: CrimeData
- data2: CrimeData
+ getDistance (): String
- distance (Double[], Double[]): Double
- deg2rad (Double): Double
- rad2deg (Double): Double
+ getTimeDifference (): String
UIDataInterface
- dataLocation: String
- comapreCrimes: Boolean
- comparator: CrimeData
+ initCrimeData(): void
+ getActiveData (): ArrayList<CrimeData>
+ addUserData (String): void
Importer
+ userToString (UserData): String[]
+ addUserData (UserData): void
+ policeToString (PoliceData): String[]
+ addPoliceData (String): void
WriteCSV
+ writeDataLineByLine (String, ArrayList<String[]>): void
ReadCSV
+ readerHelper (String[], int): CrimeData
+ readDataLineByLine (String): ArrayList<CrimeData>
FilterController
- activeLocation: String
- activeCrimeType: String
- policeDataActive: Boolean
- userDataActive: Boolean
- arrestMade: Boolean
- regionDataActive: Boolean
- regionFilteringKey: String
- dateFilteringActive: Boolean
- startDate: Date
- endDate: Date
+ getActiveFilters (): ArrayList<CrimeStat>
GraphCreator
+ formattedDatesIntoGroups(ArrayList): HashMap
+ getYear (ArrayList): List
+ createGraphOverTime(ArrayList, int): XYChart.Series
+ createGraphPerType(ArrayList, int): XYChart.Series
Model
DataManager
- allCrimeData: ArrrayList<CrimeData>
- activeCrimeData: ArrayList<CrimeData>
+ addCrimeData (CrimeData): void
+ constructCrimeData(ArrayList<CrimeData>): void
+ getData (): ArrayList<CrimeData>
DataFilter
+ filterData(ArrayList<CrimeData>): ArrayList<CrimeData>
+ isWithinRange (Date, Date, Date): Boolean
+ filterCrimeData (CrimeStat, ArrayList<CrimeData>, ArrayList<CrimeStat>): ArrayList<CrimeData>
+ sortOverrider (HashMap<String, ArrayList<CrimeData>>): ArrayList<CrimeData>
+ sortByRisk (ArrayList<CrimeData>): ArrayList<CrimeData>
+ sortByFrequency (ArrayList<CrimeData>, Boolean): ArrayList<CrimeData>
«enum»
CrimeStat
CrimeData
- id: String
- latestID: int
- address: String
- date: String
- latitude: String
- longitude: String
- location: String
- crimeType: String
+ CrimeData (String)
PoliceData
- caseNumber: String
- arrestMade: String
- domestic: String
- beat: String
- ward: String
- xCoord: int
- yCoord: int
- secondDescription: String
- locationDescription: String
- formatPoliceData (ArrayList<String>): void
UserData
+ formatUserData (ArrayList<String>): void
Figure 8: UML Class Diagram for SafePoint


View
DataViewWindow
+ constructWindow (CrimeData): Pane
+ displayWindow (Parent): void
PoliceDataWindow
- displayPane: Pane
+ PoliceDataWindow (PoliceData)
+ PoliceDataWindow (PoliceData, Boolean)
+ constructWindow (CrimeData): Pane
UserDataWindow
- displayPane: Pane
+ UserDataWindow (UserData)
+ UserDataWindow (UserData, Boolean)
+ constructWindow (CrimeData): Pane
CompareDataWindow
- pane1: Pane
- pane2: Pane
- data1: CrimeData
- data2: CrimeData
+ CompareDataWindow (CrimeData, CrimeData)
+ getPanes (): void
+ openWindow(): void
Button
CrimeViewButton
+ CrimeViewButton (CrimeData)
+ setActions (CrimeData)
Pane
DataPane
+ viewButton: CrimeViewButton
+ DataPane (CrimeData)
+ constructComponents (CrimeData)
PoliceDataPane
+ PoliceDataPane (PoliceData)
+ constructPoliceComponents (PoliceData): void
UserDataPane
+ UserDataPane (UserData)
+ constructUserComponents (UserData): void
Main
+ main (String[]): void
Application
SafePoint
+ start (Stage): void
+ main (String[]): void
DataPaneConstructor
+ loadActiveCrimes (): VBox
GraphViewWindow
- stage: Stage
+ initGraphViewWindow (): void
«interface»
Initializable
MainViewController
- regionFilter: TextField
- crimeSelector: ChoiceBox
- policeDataToggle: CheckBox
- userDataToggle: CheckBox
- arrestMadeToggle: CheckBox
- graphToggle: CheckBox
- regionFilteringToggle: CheckBox
- regionFilteringKey: ChoiceBox
- dateSortToggle: CheckBox
- startDate: DatePicker
- endDate: DatePicker
- crimeDataPanel: ScrollPane
- mapView: WebView
- webEngine: WebEngine
+ updateCrimeData(ActionEvent): void
+ reportCrime (ActionEvent): void
+ compareCrimeToggle (ActionEvent): void
+ initialize (URL, ResourceBundle): void
+ loadData (): void
+ initMap (): void
+ initCrimeSelector(): void
+ updateRegionCrimeData (): void
+ updateMapSettingsData (): void
+ updateRegionDateData (): void
ReportCrimeWindow
- stage: Stage
+ initReportCrimeWindow (): void
+ closeStage (): void
ReportCrimeController
- crimeTypeSelector: ChoiceBox
- addressField: TextField
- latitude: TextField
- longitude: TextField
- date: DatePicker
+ initialize (URL, ResourceBundle): void
+ reportCrime (ActionEvent): void
+ validateInputs (): Boolean
+ formatInputs (): String
+ constructCrimeChoiceBox (ChoiceBox): void
GraphViewController
- selectCrime: ChoiceBox
- yearSelector: ChoiceBox
- graphCreator: GraphCreator
+ viewGraph(): void
+ initSelectCrime(): void
+ initYearSelector(): void
+ constructYearChoiceBox(): void
+ graphOverTime(): void
+ graphOverTimePerType(): void
ConfirmationWindow
+ openReportedWindow (Striing): void
+ openEditedWindow (): void
+ showStage (Pane, int, int): void
GeocoderAPI
+ doRequest (String): String
EnterIDController
- crimeIDInput: TextField
- activeReportSession: ReportCrimeWindow
+ confirmID (): void
Model
DataManager
- allCrimeData: ArrrayList<CrimeData>
- activeCrimeData: ArrayList<CrimeData>
+ addCrimeData (CrimeData): void
+ constructCrimeData(ArrayList<CrimeData>): void
+ getData (): ArrayList<CrimeData>
CrimeData
- id: String
- latestID: int
- address: String
- date: String
- latitude: String
- longitude: String
- location: String
- crimeType: String
+ CrimeData (String)
DataFilter
+ filterData(ArrayList<CrimeData>): ArrayList<CrimeData>
+ isWithinRange (Date, Date, Date): Boolean
+ filterCrimeData (CrimeStat, ArrayList<CrimeData>, ArrayList<CrimeStat>): ArrayList<CrimeData>
+ sortOverrider (HashMap<String, ArrayList<CrimeData>>): ArrayList<CrimeData>
+ sortByRisk (ArrayList<CrimeData>): ArrayList<CrimeData>
+ sortByFrequency (ArrayList<CrimeData>, Boolean): ArrayList<CrimeData>
«enum»
CrimeStat
PoliceData
- caseNumber: String
- arrestMade: String
- domestic: String
- beat: String
- ward: String
- xCoord: int
- yCoord: int
- secondDescription: String
- locationDescription: String
- formatPoliceData (ArrayList<String>): void
UserData
+ formatUserData (ArrayList<String>): void



Controller
CompareDataController
- data1: CrimeData
- data2: CrimeData
+ getDistance (): String
- distance (Double[], Double[]): Double
- deg2rad (Double): Double
- rad2deg (Double): Double
+ getTimeDifference (): String
UIDataInterface
- dataLocation: String
- comapreCrimes: Boolean
- comparator: CrimeData
+ initCrimeData(): void
+ getActiveData (): ArrayList<CrimeData>
+ addUserData (String): void
FilterController
- activeLocation: String
- activeCrimeType: String
- policeDataActive: Boolean
- userDataActive: Boolean
- arrestMade: Boolean
- regionDataActive: Boolean
- regionFilteringKey: String
- dateFilteringActive: Boolean
- startDate: Date
- endDate: Date
+ getActiveFilters (): ArrayList<CrimeStat>
Importer
+ userToString (UserData): String[]
+ addUserData (UserData): void
+ policeToString (PoliceData): String[]
+ addPoliceData (String): void
GraphCreator
+ formattedDatesIntoGroups(ArrayList): HashMap
+ getYear (ArrayList): List
+ createGraphOverTime(ArrayList, int): XYChart.Series
+ createGraphPerType(ArrayList, int): XYChart.Series
WriteCSV
+ writeDataLineByLine (String, ArrayList<String[]>): void
ReadCSV
+ readerHelper (String[], int): CrimeData
+ readDataLineByLine (String): ArrayList<CrimeData>

The UML Class Diagram (Figure 8) uses a MVC software design pattern. This means there are 3 distinct modules, Model, View (implemented here as GUI) and Controller. The user interacts with the controller (through user input via the GUI), and sees the GUI. The controller manipulates the model, which in turn updates the GUI.

The benefit of having these three distinct packages with few interactions between them, means that it allows us as developers to be working on different parts of the program at the same time, without it majorly affecting others' work. It also keeps it simple when changing parts of the code, as there are not many classes interacting with each other from different parts of the program.

Inside each module, we used a Master-Slave design pattern, where one class sends out commands to the rest of the classes. In the Model module, this is the DataManager class, and in Controller this is the UIDataInterface class. The CrimeStat enum contains the different ways the CrimeData will be filtered/sorted. For more information, the rest of the values can be found in the appendix. The Importer classes interact directly with our database, in the form of CSV files.

Section 6 - Testing Procedures

6.1 Manual Testing:

ID	Test (ID)	Description	Tester	Result	Pass / Fail	Criticality	Use Case
MT_1	AT_1	Given the SafePoint application has been opened and some data has been selected by the user (such as ‘Show police data’) when the reload data button has been selected data is displayed in list form.	Matthew	Expected data is shown in the list	Pass	This is critically important to the usability of the application and without it the application cannot function. (10)	UC_1: ‘Show Crime data’
MT_2	AT_2	Given a user ticks a checkbox for comparing crimes on the left sidebar, when the user selects two crimes then an information box is shown comparing the data of the two crimes, including the distance and time between crimes.	Matthew	Expected crimes are compared, with correct comparisons	Pass	Comparing two crimes is important functionality to have as it allows users to better grasp the information shown. (7)	UC_3: ‘Compare crime data’
MT_3	AT_3	Given a user ticks a checkbox for comparing crimes, when the user selects two identical crimes nothing is compared.	Matthew	No comparing happens and window doesn’t open	Pass	Not allowing two identical crimes to be compared is important to ensure less confusion about the data. (4)	UC_3: ‘Compare crime data’
MT_4	AT_4	Given a user is on the home page (with the map showing) when the user clicks the 'Report a crime’ button then the application window	Matthew	Crime added to list and to database file	Pass	Reporting a crime is a stable requirement for the application, therefore it is critical this works. (9)	UC_4: ‘Manually add data’

		opens to allow user data input before returning to the home page after adding the data to the database when the ‘Report Crime’ is selected.					
MT_5	AT_5	Given the user is on the desktop application, when the user selects the view button under a crime record then an information box is shown with the details of the selected crime record.	Matthew	Correct data displayed and window opened	Pass	Viewing a specific crime is simple functionality that must be able to work to present accurate data to the user, without this functionality the user is unable to properly analyse crimes, which drastically decreases the value of the application. (9)	UC_5: ‘View details of a crime record’
MT_6	AT_6	Given a user has accessed the SafePoint application and data is being displayed in list form, the data being displayed should also be displayed on the map in the form of markers.	Matthew	All data points and graphical components exist	Pass	Giving a visual representation of crime locations allows the user a much easier experience when looking at crimes within a specific location, drastically increasing the usability of the program. (9)	UC_6: ‘View crime records on the map’
MT_7	AT_7	Given the user is editing a crime and invalid data is given, the crime is deleted	Matthew	Data removed from database file and list	Pass	Ensuring users have the ability to remove crimes that have been previously reported by them, for privacy reasons. (8)	UC_7: ‘Delete user reported crime record’

MT_8	AT_8	Given the user has entered the correct unique code for that crime, and pressed edit crime, a window will open to allow the user to edit it. If valid data is entered the crime record will be replaced by the new data	Matthew	Data point in database replaced with new data	Pass	Allowing users to edit their reported crimes, because they entered incorrect information, or for another reason is important to ensure users confidence when they report a crime. (9)	UC_8: ‘Edit user reported crime record’
MT_9	---	Given the user has pressed the edit crime button on a user reported crime, and the incorrect ID is entered, the program doesn’t allow the user to edit this crime.	Matthew	Window closes and no further access is given until the correct code is entered	Pass	Ensuring only the correct people can edit or remove a crime is vital to ensure integrity of the data in the program. (9)	UC_8: ‘Edit user reported crime record’
MT_9	AT_9	Given the SafePoint application has been opened, when show user crimes and/or show police crimes checkboxes have been selected and the ‘Reload data’ button is selected then the correct crimes are shown in list form on the sidebar and the map displays correct the correct	Matthew	All data from database file is shown	Pass	Filtering data is an essential way to navigate the large amounts of data to find specific crimes and is critical to the functionality of the application. (8)	UC_9: ‘Filter Crime information’
MT_10	---	Given a file path for a csv file containing crimes the program copies all the crimes into a database in the correct format.	Rory	All crimes are filtered for information required and written into the database file	Pass	Importing large amount of data into the program is a vital part of the program (9)	Not Applicable
MT_11	AT_ 10	Given the user selects the ‘graph view’ button when the user selects	Rory	A graph showing the	Pass	This functionality gives another method for the	UC_10: ‘View data in

		the ‘view graph’ button then a graph displaying the correct data is displayed		correct data is being displayed		user to view information, especially the use of tracking crimes over a period of time, allowing users to get a clear image of when types of crimes occured. (7)	Graphical form’
MT_12	AT_11	Given the user accesses the SafePoint application and the map is being displayed, when the user selects “Show region data” on the left sidebar and the user selects an option from the dropdown menu (highest frequency, lowest frequency, most dangerous, etc.). Then the list of crimes will now display the suburbs in the city according to the criteria of the user’s choice.	Rory	The list of crimes to the left of the map on the sidebar is filtered and sorted according to the user's selection.	Pass	This functionality is important to the usability of the program, and would make finding specific information much easier for the user, encouraging them to use the application more often (7)	UC_12: ‘Rank crimes in a region
MT_13	AT_12	Given the user has accessed the SafePoint application and the map is being displayed when the user enters the region from the “Filter by region” search bar. Then the region of choice is now the main focus of the map, displaying the crimes within that area on the list of crimes and the map.	Rory	The correct crimes are displayed after a correct region is entered	Pass	This is important functionality as it allows the user to easily navigate between specific regions and makes the user much more able to sort through large amounts of data. (8)	UC_13: “Search by region”

6.2 Overall testing results:

The manual tests performed covered the main functionality of the application and covered all of the critical acceptance tests that had been implemented. At present, none of the manual tests which ensure basic functionality of the project and implemented use cases have failed.

6.3 Test coverage:

Acceptance tests were covered through a combination of manual testing and automated Cucumber testing. Features that relied heavily on the GUI, which constituted the majority of the tests within our acceptance tests table, were tested manually. This was executed following white-box and black-box testing, with the developers themselves testing the product and other individuals who were not a part of the project also testing the application for possible bugs. Manual tests were mapped back to the original requirements, and ensured basic functionality of the application.

Additionally, automated testing utilising cucumber was carried out related to Adding a User Reported Crime (AT_4/UC_4), Filtering Crimes (AT_9/UC_9) and Ranking Crimes (AT_11/UC_11). These followed a similar process to unit testing and aimed to extensively test the functionalities which were most prone to small errors. The automated testing focused on these particular use cases/acceptance tests so as to test their underlying functionality separated from the GUI, which could not be done for our other features.

In terms of JUnit testing, these achieved an overall line coverage of 33% for the entire project, with 38% of classes being covered. This seemingly low percentage can be attributed to the inability of writing tests for GUI classes contained in the ‘view’ package, paired with the existence of several getter and setter methods which would be redundant to test.

Unit tests were mainly split into the two other packages related to the MVC architecture - ‘controller’ and ‘model’.

For the ‘controller’ package, 71% of classes were covered (5/7 classes - UIDataInterface and GraphCreator were not tested due to their interdependence with the GUI), with the most relevant classes such as FilterController, CompareDataController, and Importer achieving line coverage of 98%, 92% and 63%, respectively. The overall line coverage for the package was 51%.

For the ‘model’ package, 87% of classes were covered, achieving overall line coverage of 82%. The most relevant classes for functionality, such as CrimeData, DataFilter, PoliceData and UserData achieved line coverage of 100%, 79%, 100%, 85%, respectively. As DataManager involved the formatting of strings and storing of the activeCrimeData, it was not tested and contributed for an overall lower line coverage within the package.

Section 7 - Current Product Version

7.1 Use Cases and Features Implemented in this Release

In this release version of the application SafePoint, we implemented the following features:

1. View Heat Map (UC_2)
2. Deleting the user reported crime records (UC_7)
3. Editing the user reported crime records (UC_8)
4. View the data in Graph representation (UC_10)
5. Allow the users to search for crime by region (UC_14)

We were not able to implement some of the features that were slated to be implemented such as view hotels and comment on other tourists' crime records due to insufficient time.

7.2 Noteworthy Features

The most noteworthy accomplishment is the design of the application. Compared to other such apps, SafePoint has a unique design and aesthetic. One of the main key drivers, usability, is well highlighted with the accessible GUI. The majority of features the user requires (such as viewing crime data, adding crime data and filtering) can be performed through the main screen. This facilitates navigation for new users to utilise the application's main features, and ties in with our main stakeholders (tourists) being individuals from all ages and backgrounds.

We also added a manual culling function to display the data more efficiently. In our last deliverable, the performance of the program was very poor, due to us trying to load thousands of data points into separate panes on a parent pane all at once. Now we generate solely the panes that will be visible for the scroll pane. This means that even if we have thousands of data points that should be displayed, we only display 8 at once, without the user noticing any difference, except a huge performance increase. This works by taking the position of the scroll bar, and generating a large empty pane that takes up the non-visible space below the data, and another one above it. This functionality can be found in the DataPaneConstructor class.

7.3 Requirements and Features in next Release

As this is the final release of this product, we have no new features slated to be implemented anymore.

Section 8 - Risk Assessment

Risk assessment is the identification and analysis of potential events that could implicate the confidentiality, integrity and availability of information resources on the project and the tolerance of such an event. The impact and likelihood of the risks are measured by the terms High, Medium and Low. High indicates that there is a greater chance of the occurrence of the risk in terms of likelihood and that if the risk occurs, then the damage incurred by it would be great in terms of impact. Medium and Low indicate the same in decreasing ranks of priority. The consequences are the damages or problems caused by the occurrence of the risk and countermeasures are the steps taken to mitigate the consequence as much as possible.

The table is structured in such a way that the level of impacts and likelihood of the risks are mentioned in descending order (high to low), where impact is given more precedence than likelihood.

The risks identified, the consequences of the risk, the impact they have, and actions taken to ensure the possibility that the risk occurs is as low as possible are given in Table 6.

Table 6: Risk Assessment table

ID	Risk	Consequences	Impact	Likelihood	Responsibility	Counter-Measures
R1	Occurrence of Scope creeps in the middle of the project.	Can cause delay in submission, impact the cost, may introduce unforeseen risks and can also impact the overall quality of the project.	High	High	Development team, stakeholders.	Include the possibility of occurrence of scope creep in the middle of the project in the planning stage and take precautionary measures by allotting extra budget and time to it.
R2	Inaccurate Resource Estimates like insufficient time.	Can cause us to put a lot of constraints on the development of other resources. May also lead to the final product being released having bugs or other problems.	High	Medium	Development Team	Prioritize the work properly and consider the uncertainty risk factor in the planning stage. Also, ensure you are careful while allocating your resources and make sure to have daily analysis about the current state of your deadline.
R3	Technical Difficulties causing loss of data and/or integrity like laptop crashing before pushing to Git.	Can cause loss of a great amount of data and can also compromise data integrity leading to delay in project completion or project failure.	High	Medium	Development Team	Ensure proper risk mitigation strategies are applied to prevent such difficulties like frequent commits and pushes to Git and make sure not to post the code on open source forums which could allow others to copy it.
R4	Covid-19 causing lockdown	May cause the entire team to work from home causing limited team meetings due to which project completion efficiency may suffer.	High	Medium	Development Team	Ensure that everyone has zoom set up so that the team can have frequent meetings and make sure that everyone is updated on what has been done, what needs to be done and who is doing what and also help each other if they have

						any problems doing their tasks. Also perform pair programming among team members to increase efficiency.
R5	Poor Productivity due to developers being burnt out.	Can cause delays in reaching deadlines and increase the cost of maintainability, due to which the project sponsors or stakeholders may withdraw their support and cause the loss of credibility for the developing team.	High	Medium	Development Team	Set team-imposed deadlines to be able keep track of progress efficiently and also be sure to make the team member take regular breaks which will lead to stress-free working making the person more productive.
R6	Inadequate Risk Management	May lead to many serious issues like insufficient time, bugs in the code and compromise code integrity ultimately leading to the failure of the project.	High	Low	Development Team	Acknowledge risk possibility, as in always make sure to follow the risk assessment and not become overconfident and consider prior mitigation strategies while also including risk in project estimations.
R7	Poor Quality Code like bugs or code repetition	Can cause the app to break or cause bugs and other major or minor issues or can also cause technical debt accumulation leading to higher costs for project management and completion.	High	Low	Development Team	Ensure the quality of the code is consistent and code is error free by cross- checking with more than one person. Also make sure that there is no duplication and that all basic programming rules of thumb such as ‘Rule of Three’ and ‘KISS’ are followed.

R8	Team member drops out / leaves	Can cause the entire team structure to destabilise and put additional pressure on the remaining team members.	High	Low	Development Team	Split the tasks of the team member who left equally among the other team members so that one person does not get too much pressure on themselves and also ask for some extension to complete the project due to the lack of team members.
R9	Team members cannot perform efficiently due to unexpected personal issues like the death of a family member.	This will cause the project quality to suffer and the other team members to take up the slack and also may cause the team member to drop out if it's serious enough.	High	Low	Development Team	Give the team member time off to cope with their issues and in the meantime split their tasks among the team equally or to a member who has less to do, also if possible ask for a time extension for the completion of the project.
R10	Health Issues / Serious Injuries faced by the developing team	Can cause delays in finishing the tasks and deliverables and put more pressure on the rest of the team members.	Medium	Medium	Development Team	Ensure that the member who's sick or injured has Zoom set up so that they can keep up with regular communication and can complete their tasks on time.
R11	Stakeholders may have expectations developers may not be able to fulfil	Project may not be successful due to the expectations of the stakeholders not being fulfilled causing them to withdraw their support.	Medium	Low	Development Team	Obtain frequent approval and acknowledgement of the project from a stakeholder.
R12	Lack of proper code review.	Can cause the final product to have unexpected errors , bugs or may cause it to have redundant or suboptimal code.	Medium	Low	Owner, Development Team	Ensure proper code review is conducted by the team members who have not worked on the code being reviewed. Also perform proper integration testing to make sure that it works efficiently with the code written by other developers.

Section 9 - Project Plan

A project plan is a plan used to document milestones for the project. This also breaks the project down into achievable goals within realistic time restraints so that a team following the plan will be able to successfully achieve the set goals.

9.1 Major Milestones

Table 7.1: Major Milestones

Milestones	Date	Description	Related Tasks
M1	09/08/21	First deliverable	● Checking all requirements are fulfilled. ● Making sure all the required documents are stored neatly to make it easier to understand. ● Submitting the deliverable as required.
M2	27/09/21	Second deliverable	● Running unit tests and integration tests to ensure the code does not have any bugs and is running as intended. ● Making sure all the classes and tests are packed into a single zip folder for submission. ● Ensuring the design document topics required to be submitted are included. ● Submitting the deliverable as required.
M3	11/10/21	Third deliverable	● Running last minute tests to ensure there are no unexpected issues. ● Ensuring all documents required are ready to be submitted. ● Submitting the deliverable as required.
M4	Week 11	Project demo	● Going to the mentioned venue for the demo and showing off the application.

9.2 Minor Milestones

Table 7.2: Minor Milestones

Milestones	Date	Description	Related Tasks
M1	22/07/21	Initial team meeting, design document and task delegation	● Introductions with all team members. ● Setting up communication. ● Delegating the first set of tasks and outlining expectations for the first deliverable.
M2	30/07/21	First set of tasks due date and second set delegation	● Ensuring the first set of tasks delegated to the team are done and if not seeking a reasonable explanation. ● Delegating the second set of tasks for the first deliverable. ● Discussing various topics related to the project tasks.
M3	05/08/21	Second set of tasks due date	● Discussing progress and ensuring the completion of all tasks for the first deliverable. ● If a task is still due, further breaking up the task and delegating it to ensure timely completion. ● Cross-checking every team member's task to ensure it is up to the Min. ● Discussing further plans.
M4	12/08/21	Planning the tasks of each member in implementation phase	● Meeting and discussing the implementation phase and delegating tasks to each team member. ● Ensuring that each team member is aware of the coding style decided to be used, to make the overall look consistent.
M5	16/8/21	Project setup complete and first test build done	● Ensure all team members can pull/push from github, and are aware of how to create/manage branches etc. ● Create example build where each team member contributes to get practice in the proper procedure
M6	19/8/21	Implement Model module section 1.	● DataManager Class, CrimeData class, PoliceData class, UserData class must be implemented. ● Testing must be done in each individual class, and integration testing between newly implemented classes. ● Ensure task load on each team member is acceptable.

M7	29/8/21	Implement GUI module section 1	● Implement the classes to create the main window, including header, settings panel, and crime panel. ● Add required components (e.g buttons, checkboxes, dropdown menus), and ability to easily adjust the action events for these components. ● Ensure extensive testing throughout the GUI panel. ● Integration tests on Module Section 1, e.g display of CrimeData objects properly displaying.
M8	5/9/21	Implement Controller module section 1	● UIDataInterface Class, and FilterController class to be implemented. ● Testing in each class, in between each other, and integration tests with the Model module component must all be completed.
M9	12/9/21	Implement GUI module section 2	● All buttons, filters, and settings in the UI to be created and assigned a function. ● User input should now affect the implemented classes in the Model and Controller modules. ● Extensive testing to be done on all operating systems to ensure consistent expected response on all possible devices.
M10	19/9/21	Implement Model section 2	● Implement DataFilter Class and CrimeStat enum. ● Individual testing of DataFilter Class. ● Integration testing with Controller and GUI modules, ensuring all functionality works with UI and edge cases are accounted for.
M11	26/9/21	Conducting Full Code Review 1	● All team members look through code, emphasis on code developed by other team members. ● Perform JUnit testing to ensure it has a respectful percentage of coverage. ● Perform integration testing to ensure that there are no unexpected problems in integrating the various classes. ● Ensure Documentation including Java Doc is present and it matches the style throughout the code. ● Ensure all code standards like style and limited use of static and global variables are implemented as agreed .
M12	30/9/21	Construct Database	● Construct a basic database. ● Add data.

			● Ensure database requirements are met for the task.
M13	3/10/21	Implement Controller section 2	● Implement Exporter Class and Importer class and their subclasses. ● Ensure proper interaction with the database. ● Integration tests with DataManager and CrimeData classes.
M14	7/10/21	Implement GUI section 3	● Implement Map API. ● Implement Graph display. ● Integration tests.
M15	10/10/21	Conducting Full Code Review 2	● All team members look through code, emphasis on code developed by other team members. ● Ensure JUnit testing is up to standard. ● Ensure Documentation including Java Doc is present. ● Ensure all code standards are implemented how we agreed.

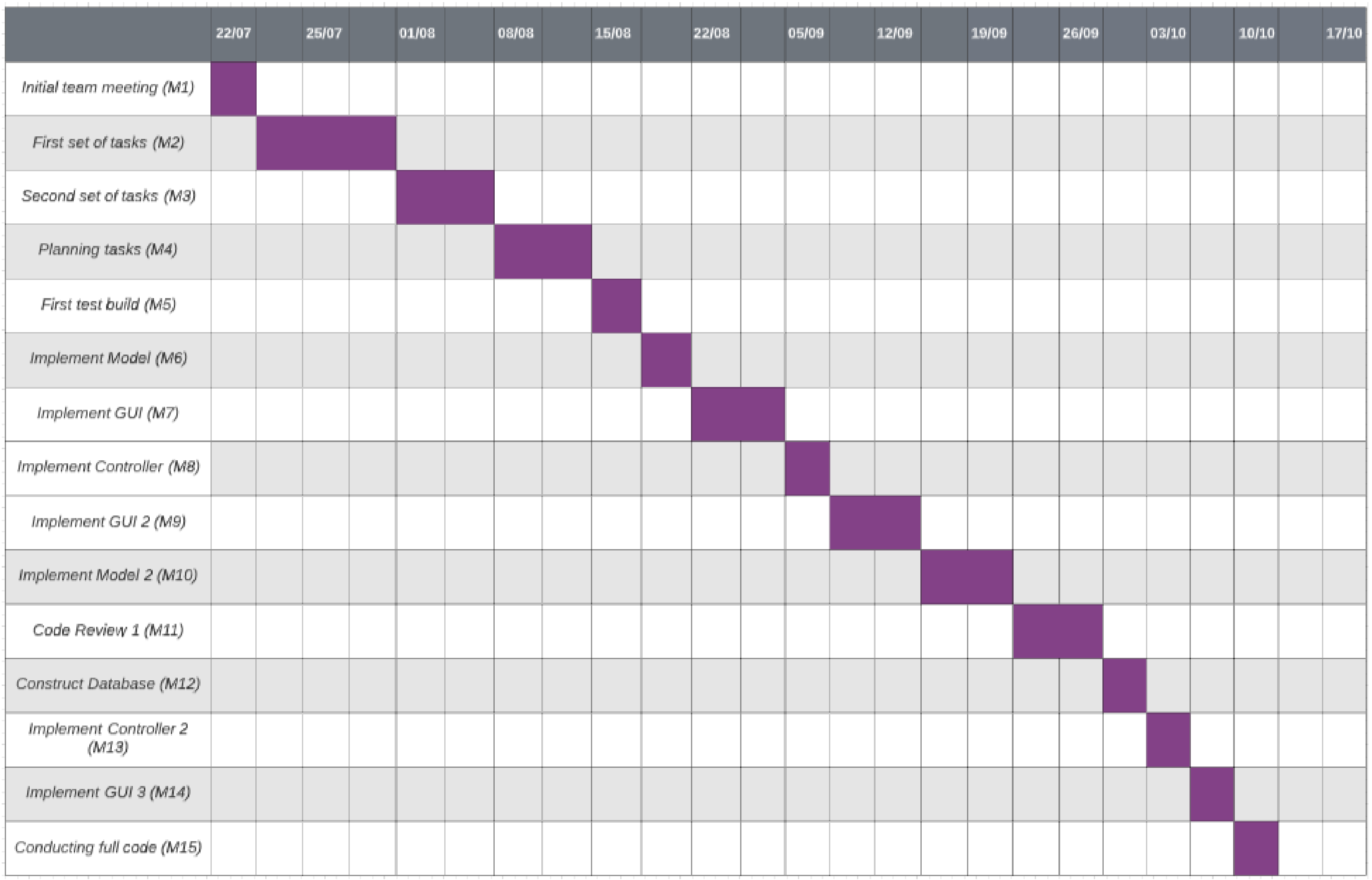


Figure 9: Gantt Chart for Table 7. Shows a visual representation of the minor milestones, including their start and end dates.

A Gantt chart was created to visualise our minor milestones and the timeframe required to achieve them.

Though the Gantt chart doesn’t show the visualisation of them, buffers were added between some of the milestones to make sure that the development team have the time to relax and ensure that they do not suffer from burn-outs. The inclusion of buffers also ensures that any unforeseen issues causing delays, such as pandemic lockdowns, do not put too much stress on us to meet the deadlines.

For example, the week ending on 17/10 leads up to exams for all team members, therefore the bar for the milestone on that week is shorter, so that less time needs to be spent on this project. During the 2 week term break the expectation for team members is that work still needs to be done on the project, albeit slightly less than during term time. This means that there are gaps in the plan at this time, and the milestone lengths are longer than they normally are, to visualize that work is being done, although with breaks in between.

Section 10 - Lessons Learned

10.1 Danish Jahangir

Time Management

One of the first key lessons I learned while working on the project was that as important as I thought time management was, I was still underestimating it, which unfortunately led to a fatal error while submitting our ‘Phase 2 deliverable’ causing my team to lose quite a few marks. Working on the project day in and day out made me realise how important it is especially in real-life scenarios to manage work-life and other responsibilities simultaneously while making sure that neither suffer because of too much focus on one side.

Communication

Having great communication between the members working on the same project makes it much more efficient was one of the lessons me and my team learned early in the project. Due to this, even though we had a few hiccups during lockdown, we were mostly able to avoid too many clashes and were able to work together pretty well..

Planning

The thing I learned about planning is that one should always make sure to keep some room for flexibility. Following a rigid structure of a plan is next to impossible due to the fact that external factors can and will interfere in it. Having buffer times allow us to cover those missed times and also sometimes allow for some downtime for some R&R.

10.2 Matthew Garrett

New Skills

Throughout this course and project I learnt many new skills that will help me in my future courses and career. Unfortunately, many of these skills were learnt too late to be fully implemented into SafePoint due to time constraints. Having a self motivated learning strategy ended up working really well for me, as I found resources and made notes based on how I learn, rather than following a strict lesson plan that is designed to be used for many people.
Maven, GitLab, CI, and MVC were all technologies that I learnt during this project, and this project has made me understand them much better than if I just watched a lecture on them. I also learnt how to properly work with teams in a software engineering setting.

Time Management

Communication

This project has taught me that communication between teams is vital for project success. Prior to this project I assumed that one meeting a week would suffice for communication in teams, but this has taught me that constant communication is much more important. As well as this, pair programming and co-located programming makes it much easier to get work done, and improves the quality of the work. In my experience with this project, most of the issues that cost us a lot of time could have been solved earlier on with a small question to the team.

Planning and Testing

Planning and testing were both aspects of project management that I wasn’t too familiar with prior to this project, and as such were both implemented relatively poorly. Because the technologies that we as a team used for SafePoint are new to us, we didn’t account for time in our plan, or tests in our test cases unique to these technologies. Often test cases were written after the class was written, due to us not fully understanding the limitations of what we were using, for example anything with JavaFX functionality. This was also reflected in our plans, where we only allocated a small amount of time for Map API and Graphs, as our research prior to making this plan led to us assuming it would be a relatively easy task to do. In future projects more in depth research will be undertaken, and a full understanding of the underlying technology will be required before writing either tests or project plans.

10.3 Priscilla Ishida-Foale

Unforeseen Issues, Planning

One of the main lessons I learnt from this project ties to the importance of a good project plan. The inclusion of buffers when unexpected events occur (in our case, the most significant issue being lockdown), is invaluable to ensure that a project can run smoothly. Being able to come up with strategies to rethink our current situation when things deviate from what was originally intended was also a skill developed throughout this semester, as we reallocated specific tasks to suit varying schedules.

It was also interesting to compare the amount of time we originally estimated a particular feature/milestone would take compared to the time that was actually taken for implementation. From a personal point of view, I did not expect the data filtering to take as long as it did - the UML class diagram had to be changed as some of the original features did not work well with our data.

New Skills

The self-teaching aspect was quite interesting to me - as I was responsible for implementing most of the features related to the Google Maps JavaScript API, I had to learn new skills (in particular, JavaScript and how to execute these functions calling them from Java). It was also my first time working with an API, so getting to understand how this works was a novel experience. I struggled quite a bit with understanding how Maven works and how to handle dependency issues, however, so ensuring everything is set up properly (and in all our team members' machines) from the beginning is definitely an aspect I will take into consideration for future projects.

Time Management

Unlike other courses, SENG202 was different as it relied heavily on working independently and managing your own time well. This was quite different to any other course I had previously taken, therefore the shift took some adjusting - it was harder to keep a structured and consistent schedule as different sections of the project required varying levels of work/hours to be put in. Using tools such as Trello to our advantage certainly helped with the organisation of the project, and on a personal level, applications such as Todoist helped me improve my own time management skills by prioritising things that needed to be done (whether for SENG202 or other courses). Despite these new tools, there was still added pressure especially on the final days to the deadline, which is something I would like to improve on for future projects.

Communication Issues

I believe our team struggled mainly with accountability and the lack of open communication when it came to problems such as technical issues that we were going through. Additionally, many times misunderstandings happened as we were not sure who was responsible for/working on what feature. This would consequently lead to complicated merges as our branches were not consistently being merged into master. We improved on this issue particularly in the final weeks of the project as we adhered to a consistent branching and merging strategy.

10.4 Rory Holmes

Time Management

Going into the project I knew that it would be the biggest coding task I had undertaken so far and knew immediately that having good time management would save us down the line. I unfortunately still managed to underestimate the workload that the project brought and thus struggled with managing it for each stage of the project, which in turn led to cramming tasks in the weeks leading to a deadline rather than spreading out the work evenly. I also underestimated how much other courses would affect my schedule which contributed to a less even spread of the workload throughout my time.

Communication

Although I knew that a project such as this would require lots of communication, I definitely underestimated how much was needed and the problems we would face due to poor communication throughout the project. I think as a team we all learnt to better communicate what was happening to each other but in the early stages of the project there was much less communication which I think tripped us up further on.

New Skills

The project taught me a lot of new things other than the importance of time management and communication in a project. More specifically about the methods to manage time and communication such as effective use of the trello board as well as vastly improving my skills in git, maven and java which gives me some confidence to understand how a real world project is implemented.

Appendix

Section 5.2 - UML Diagrams

5.2 UML Class Diagram:

CrimeStat (enum): [LOCATION, CRIME_TYPE, ARREST_MATE, DATE, DATE_RANGE, POLICE_DATA, USER_DATA, HIGH_FREQUENCY, LOW_FREQUENCY, HIGH_RISK_AREA, LOW_RISK_AREA]

When filtering information, an ArrayList containing CrimeStat values is passed in, any or all of the CrimeStat values NOT in this ArrayList will be filtered out. For example, if all except for POLICE_DATA is passed in, all CrimeData will be shown except police crime data. Some values, such as HIGH_FREQUENCY will change the way the data is manipulated, for example, HIGH_FREQUENCY will sort the data based on the highest frequency areas and lowest frequency areas and show them in the Crime Info list.

Higher quality PDF files for each figure can be found in the submitted folder.

References

1. Glensor, R. W., & Peak, K. J. (2004). *Crimes Against Tourists*. Retrieved from ASU Center for Problem-Oriented Pricing: https://popcenter.asu.edu/content/crimes-against-tourists-0
2. Neighbors by Ring. (2021). Retrieved from Ring: https://ring.com/au/en/neighbors
3. Citizen. (2021). *Citizen*. Retrieved from Citizen: https://citizen.com