ASSIGNMENT ONE

mAP EVENT MANAGER FOR USERS WITH COGNITIVE BRAIN PROBLEMS

**ICT365 MAJOR ASSIGNMENT**

**CHECK LIST**

**Surname (Family Name):** Horbury

**Given Names:** Kye

**Student Number:** 31900516

**Tutor’s Name:** Afaq Shah

**Assignment Due Date:** 17/9/2021 **Date Submitted:** 17/9/2021

**Your assignment should meet the following requirements. Please confirm this (by ticking boxes) before submitting your assignment.**

* All details above are complete.
* I acknowledge and agree that the assessor of this assignment may, for the purpose of assessing this assignment, reproduce this assignment and provide a copy to another academic staff member.
* I am aware of Murdoch University’s assessment policy 8.1.8 that states that “Unit Coordinators have the right to submit any assignment to the University’s plagiarism detection software if they suspect plagiarism.”
* I have read and understood the requirements for submission of assignments specified in the Unit Information and Learning Guide.
* I have read and understood the requirements for documenting and submitting this assignment that are specified in the assignment question sheet of this assignment.
* I understand that the archive file should be submitted to ICT365 Unit LMS.
* I have kept a copy of this assignment, including the archive file.

# Description of problem

The problem is to create a windows form application that displays a map containing different events to allow people with cognitive brain problems, such as dementia, to see where they have been and what they have been doing. When selected it gives information to that user to help remind them of their activities to help to give extra security and confidence to the user as well as reduce anxiety.

### Advanced features

1. Image events have custom images displaying, with the user able to choose images from file. For example, they can choose a playground image from file and set it to the event to show they were at a park. This can also be done with videos.
2. Each event can have its information altered (such as its title and/or information) when the app is running. Elements can also be deleted completely from the map. These alterations are then saved to the XML file, meaning changes can be enacted even the user restarts the program.
3. A calendar that contains all the events. When the user selects a day, it will show all the events that were created on that day. Selected that event will take you to that event on the map, then close the calendar and bring up that events information automatically.
4. Find an event by name or move by latitude and longitude.
5. The application can create a line to show a basic route from the user’s current location to an event. Will also determine the distance the current location and the event.
6. The application can create a route and link between all the different events depending on when each event was created, showing the general path that the user has taken to allow them to keep track of where they have come from and how. This gives linking to ALL events.
7. A bus stop finder. Clicking this button will take you to the nearest bus stop and display the name of the bus stop.

# Self-DIAGNOSIS, EVALUATION AND DECLARATION

**The following is a list of all the working components of the Assignment 1 program:**

- Event Storage in an XML (example XML uses 20+ events). Such events include tweets, facebook status updates, images, video and GPX data. This file is located in the Debug Folder in the solution.

- Windows Form Application that contains a working map (including movement and zoom commands)

- Loading of Events in the application from XML via LINQ, which are stored in a dictionary to be used in the application and to be displayed on maps.

- All events are displayed on the map with icons, which can be selected to display information about that event. A different pin colour is used for different events (blue for facebook, red for image, yellow for twitter and green for video, and purple for GPX)

- Each event popup will contain information about the event, such as the time and date the event was created, the name/title given to it by the user, the latitude and longitude of the event as well as the contents of that tweet.

- Double clicking on the canvas will bring up another form that the user can use to create a new event.

- Each event can have its information altered (such as its title and/or information).

- Element can be deleted completely from the map (as well as the XML file).

- Image events have custom images displaying, with the user able to choose images from file. This also applies to video events as well.

- The user can input latitude and longitude in the sidebar, for which will move the user to that location.

- The user can search for an event via its given title by the user, and the application will take map to that location and display that events data.

- The application can move the user to the nearest event.

- The application can create a line to show a basic route from the user’s current location to an event. Will also determine the distance the current location and the event.

- The application can create a route and link between all the different events depending on when each event was created.

- A calendar that contains all the events. When the user selects a day, it will show all the events that were created on that day. Selected that event will take you to that event on the map.

- A bus stop finder. Clicking this button will take you to the nearest bus stop and display the name of the bus stop.

# Solution of problem

The solution to problem was to create a windows form application that has a dictionary that can insert different type of events into it. These events are then loaded and accessed through the EventManagerSingleton. The windows form then then use calculations of the events and change the GUI depending on what class is accessed through the eventmanager. The bus finder and calendar classes also access events in this way and change GUI accordingly. These 3 different accesses will then display, show and calculate information about these events.

### DESCrIPTION and Justification OF DESIGN PATTERNS USED

1. **Inheritance (Abstract Class Event)**

The application has one abstract class called Event, which is the base class for all the different types of events. The actual specific events then inherit off this, which includes the TwitterEvent, FacebookEvent, ImageEvent, VideoEvent, GPXEvent.

The reason inheritance with an abstract class was chosen was because it is easy to create and add new event classes, as they can inherit from Event and have all the implementation of the base class, requiring less coding and work as the base implementation is already done. It also allows us to store all the different events in the same Event dictionary, and then use their implementation of the Event classes virtual methods to perform the child specific requirement (for example with add event), without knowing what type that child is when accessing it from the dictionary. Finally an abstract class was chosen over an interface as the base class still has implementation that is common throughout all the child classes.

1. **Singleton pattern (EventManagerSingleton)**

A singleton was used for the EventManagerSingleton, which manages all the Event occurrences, such as when an event needs to be accessed due to an event firing in the window forms, or whether a particular event is needed for a certain calculation.

The reason the singleton was used was because it is referenced it multiple other different classes that do not interact without each other directly, such each different windows form. It was also used as only one instance of the manager should ever exist in the program, as it handles all the multiple different events (which can have many instances). Finally, the event dictionary data needed to be available in all references to the EventManager, and as such creating new objects of the event manager in other classes will not have a correct reference to all the data of the event manager.

### DESCrIPTION and Justification OF DESIGN principles USED

There were many different design principles that were considered with the application.

1. **Open closed Principle**

The application can add new event types to the system with touching any of the other events, and those are closed to modification needed. Only the selector needs to add 1 basic if statement.

1. **Liskov Substitution Principle**

Each event, such as the twitter event, can substitute for the base event, and does not remove base class behaviour, only extend it. For example, twitter event doesn’t remove base events ability to add or delete events.

1. **Cohesion**

All the classes’ methods are related and have high cohesion. For example the eventmanagers methods only involve the managing of method, the event methods only deal with changes within itself and its corresponding data to xml and busfinder only deals with bus finding.

1. **Coupling**

Each method only deals with a particular job itself (with the help of helper methods rarely to reduce method size) . For example in the start event method, it ONLY involves searching for a particular event with no help.

### Refactoring

The code was refactored numerous times over the course of the creation of the application.

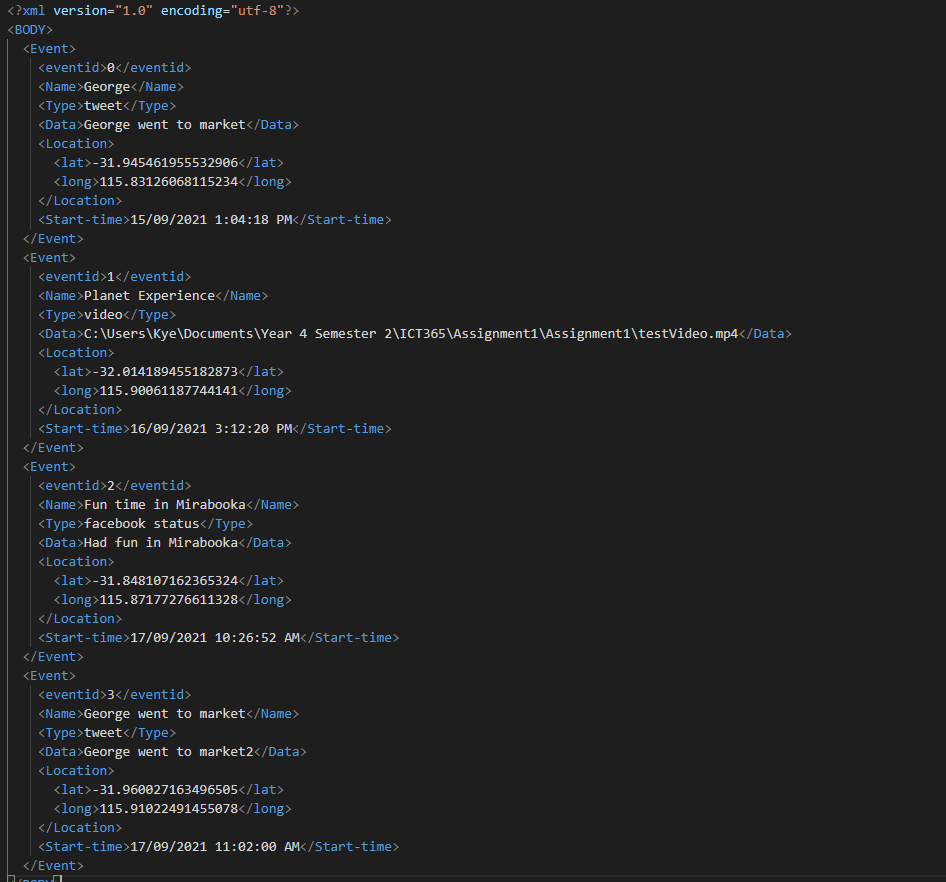
- The code was refactored when EventManagerSingleton was controlling too much of the event’s behaviour, such as when the event was created or altered it was also assigning the events individual variables. As the events themselves can handle this, the code was refactored where either the base event or the event child implementation implemented this functionality.

- The EventMangerSingletons method LoadXML and CreateEvent use to contain the hardcoded ability to determine an events type on event creation. This, one, made the methods big and, two have the same code implemented in both methods. Instead, this code was refactored out of the 2 methods into its own method, allowing it to be used multiple times.

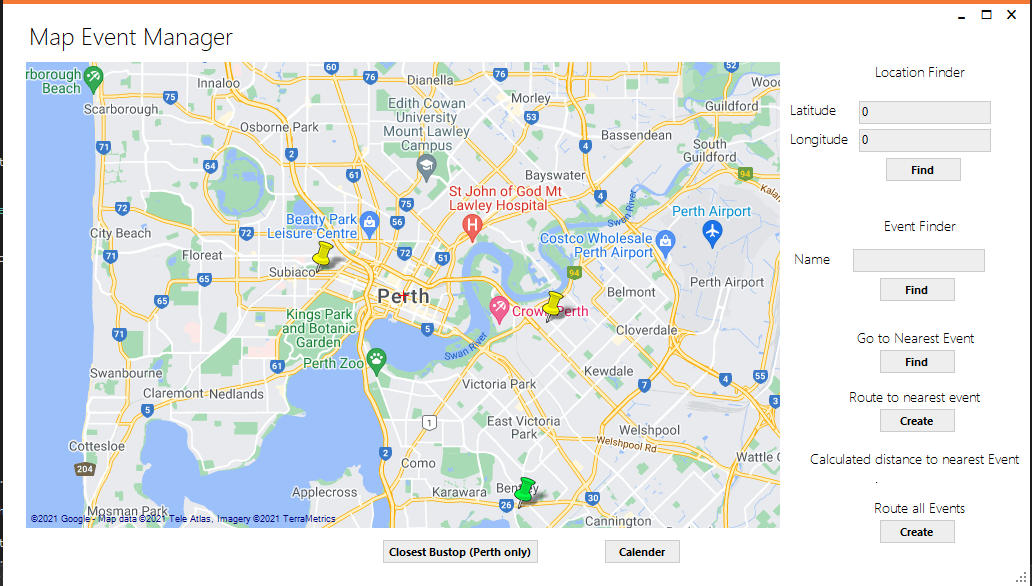
# eVIDENCE OF SOLUTION

SOME IMAGES ARE CUT OFF, THIS IS NOT TRUE FOR THE ACTUAL APPLICATION!

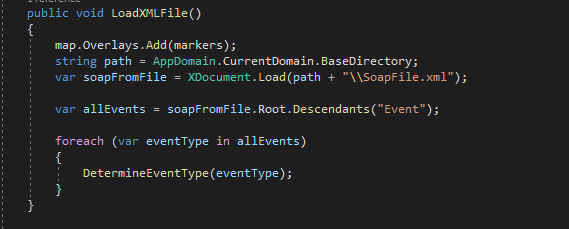
**Event Storage in an XML (example XML uses 20+ events). Such events include tweets, facebook status updates, images, video and GPX data.**

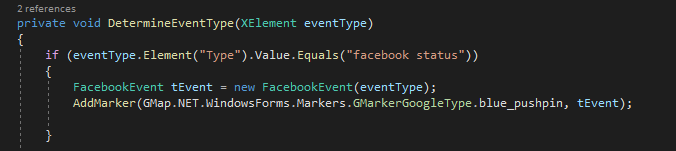


**Windows Form Application that contains a working map (including movement and zoom commands)**



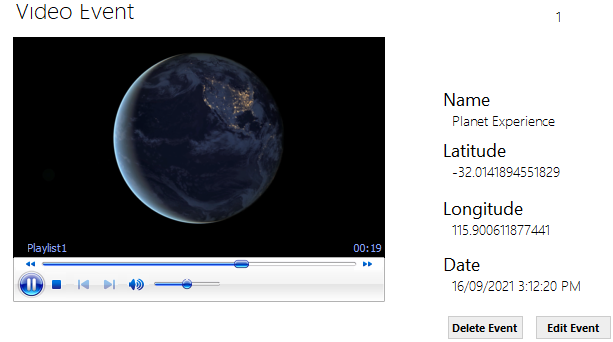
**Loading of Events in the application from XML via LINQ, which are stored in a dictionary to be used in the application and to be displayed on maps.**



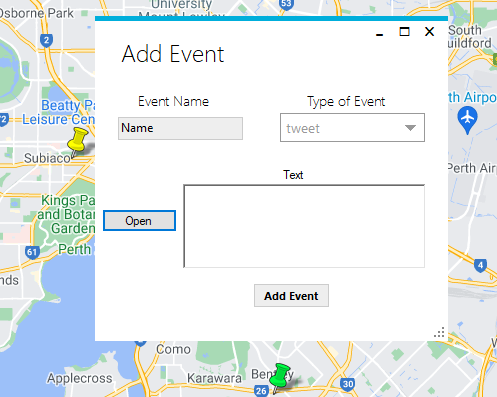




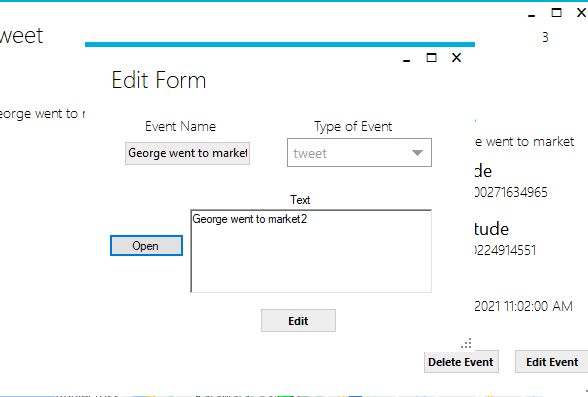
**Each event popup will contain information about the event, such as the time and date the event was created, the name/title given to it by the user, the latitude and longitude of the event as well as the contents of that tweet.**



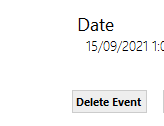
**Double clicking on the canvas will bring up another form that the user can use to create a new event.**



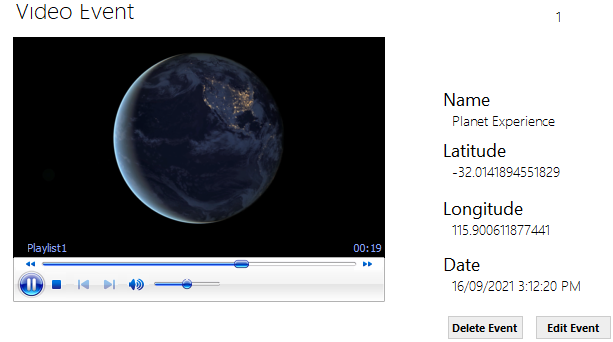
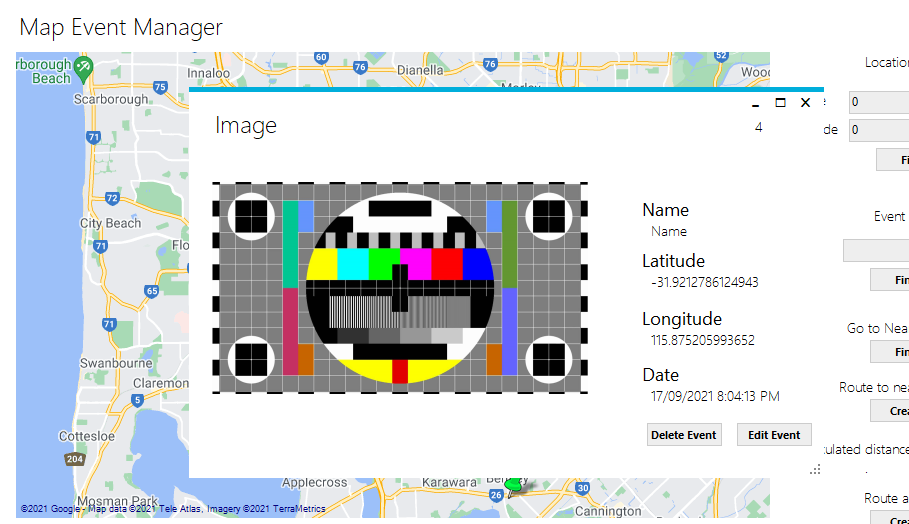
**Each event can have its information altered (such as its title and/or information).**



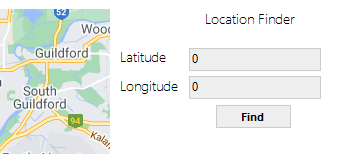
**Element can be deleted completely from the map (as well as the XML file).**



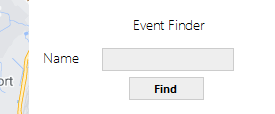
**Image events have custom images displaying, with the user able to choose images from file. This also applies to video event as well.**



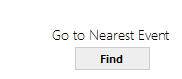
**The user can input latitude and longitude in the sidebar, for which will move the user to that location.**



**The user can search for an event via its given title by the user, and the application will take map to that location and display that events data.**

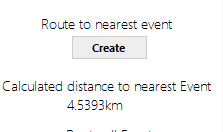


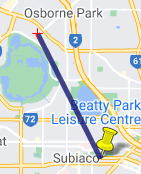
**The application can move the user to the nearest event.**



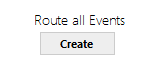
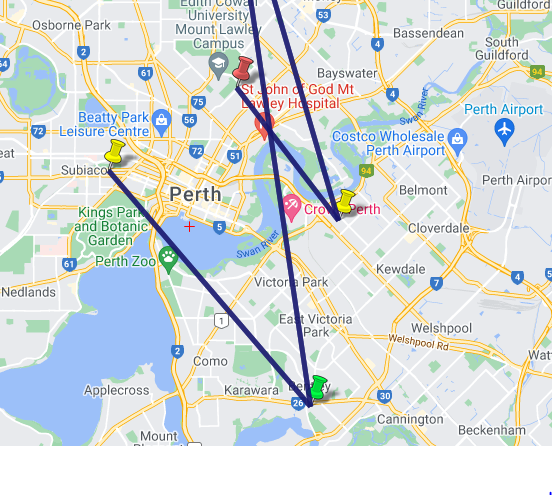


**The application can create a line to show a basic route from the user’s current location to an event. Will also determine the distance the current location and the event.**



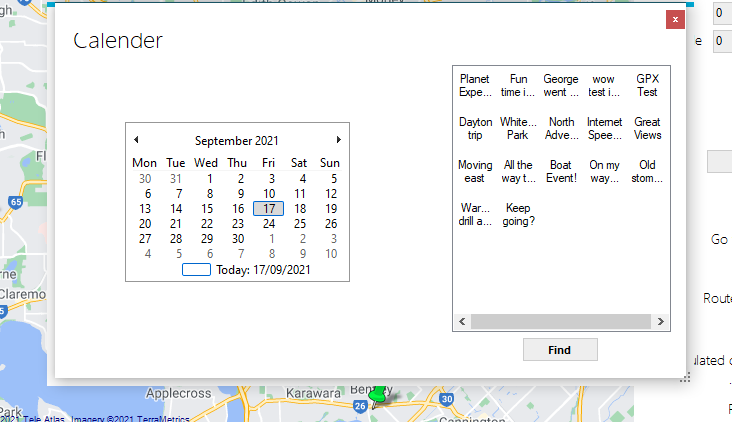


**The application can create a route and link between all the different events depending on when each event was created.**

**A calendar that contains all the events. When the user selects a day, it will show all the events that were created on that day. Selected that event will take you to that event on the map.**





**A bus stop finder. Clicking this button will take you to the nearest bus stop and display the name of the bus stop.**

