Map Applications using WinForms in C#

Table of Contents

Comparison and Reflection

ASP.NET is a web development technology provided by .NET framework from Microsoft to produce dynamic webpages. It is a popular web-development framework for building web apps on the .NET platform. ASP .NET pages are text files with an .aspx file name extension. The ASP .NET runtime parses and compiles the file into a .NET framework class, and this class is used to dynamically process the incoming requests. It is interesting to note that the .aspx file is only compiled once, and the compiled type instance is subsequently reused across multiple requests. An ASP .NET page is created by simply taking an existing HTML file and changing its file name extension to .aspx. For better error messages for our asp .net pages, we can also add a file named Web.Config. The scripting language used with Asp .NET are Visual Basic .NET and C# .NET [1].

ASP .NET web forms enables us to establish an interaction to the web applications. The browser submits a web form to the web server and in response the server returns a web page [2]. The .Net framework is made of an object-oriented hierarchy. An ASP.NET web application is made of different pages. When a user requests a specific page, the IIS server assigns the page processing to the ASP.NET runtime system. The ASP.NET runtime system transforms the aspx page into an instance of a class. Hence, each ASP.NET page is an object itself [2].

XAML stands for Extensible Application Markup Language. It is a declarative language based on XML. It is a language that is normally used to instantiate .NET objects. It is mostly used for designing GUI's, and the goal of XAML is to help in creating user interface elements. All XAML files are converted into BAML at runtime by MSBuild which is an assembly present in .NET framework. BAML is the binary representation of XAML that we see in our UI. The properties of Classes defined in XAML are specified as attributes. The code-behind class is generated using partial class concept of C#. This code-behind class is responsible for functioning of WPF application.

ASP .Net is easy to execute on the IIS server and requires less coding to build complex and large applications. It gives developers freedom to integrate various APIs to build projects and implement different useful features in the applications. Moreover, it is more fitting to use ASP .NET for applications that deal with SOAP XML data. Hence, it was more fitting to use ASP .NET for this assignment purpose. Both ASP .Net and XAML give developers graphical development environment and can be used to develop web applications. There is separation of code and UI in both XAML and ASP .NET, this allows us to clearly separate the roles of designer and developer. XAML is a declarative language which is based on XML. It is mainly used to develop GUIs and can be used for other purposes such as declaraing Workflows in Workflow Foundation. Unlike ASP.NET, XAML code gets converted into BAML code at runtime.

ASP.NET libraries can be used on .NET Core. So to migrate a WPF XAML app to .NET Core for integrating with ASP .NET, the first step is to review all the app's dependencies and make sure that the references are in the format that enables us to easily migrate to .NET Core. The next step requires NuGet dependencies to be updated and the upgraded NuGet dependencies should use the <PackageReference> format. After that, the project file needs to be migrated to the new SDK-style format, and this requires us to either target .NET Core or .NET Standard compatibility. The relevant project file properties and items need to be copied to the new project file. Next, build issues need to be dealt with and this is done by adding a reference to the *Microsoft.Windows.Compatibility* package. The APL-level differences need to be spotted and fixes. Another important step would be to remove the *app.config* sections except the *appSettings* or *connectionStrings*, and then regenerated the generated code. The final step would be to confirm that the ported app works as expected by performing testing at runtime. There can be instances were exceptions like NotSupportedException pop-up, so these need to be dealt accordingly [3].

Description

The aim of this solution is to develop a simulation program that has a map as a foundation and is dependent on a person using lifelogging technology to input valuable personal information into the system, example, by uploading geotagged photos or status updates. This application can act as a kind of cognitive scaffold for Dementia sufferers supporting recall of important events, people and so on, in a person's environment. The lifelog events are stored in a SOAP XML file which would act as the database for this application.

This project deals with Windows form application development. Windows forms is a Graphical User Interface (GUI) class library available in .Net framework. Its main feature is to provide a simple interface to develop desktop applications. It is also known as WinForms, and the applications developed using this platform are known as Windows Forms Applications. WinForms can only be used to develop Windows forms applications and does not support development of web applications. WinForms provides various types of controls like textbox, drop-down lists, radio buttons, etc [4].

Specification

More than 20 events were added to a SOAP XML file with a minimum of two tracklog type events. The Windows form consists of a canvas map to display events from the SOAP XML file. The GMap.NET library was used to develop the canvas map. GMap.NET is a powerful, open-source and a cross-platform tool that enables geocoding and map presentation. GMap offers use of Google map, Yahoo map and Bing map, in this application I have implemented Google map. The events were plotted by extracting latitude and longitudes from the event information SOAP XML file. Markers have been used to represent an event on geographical location.

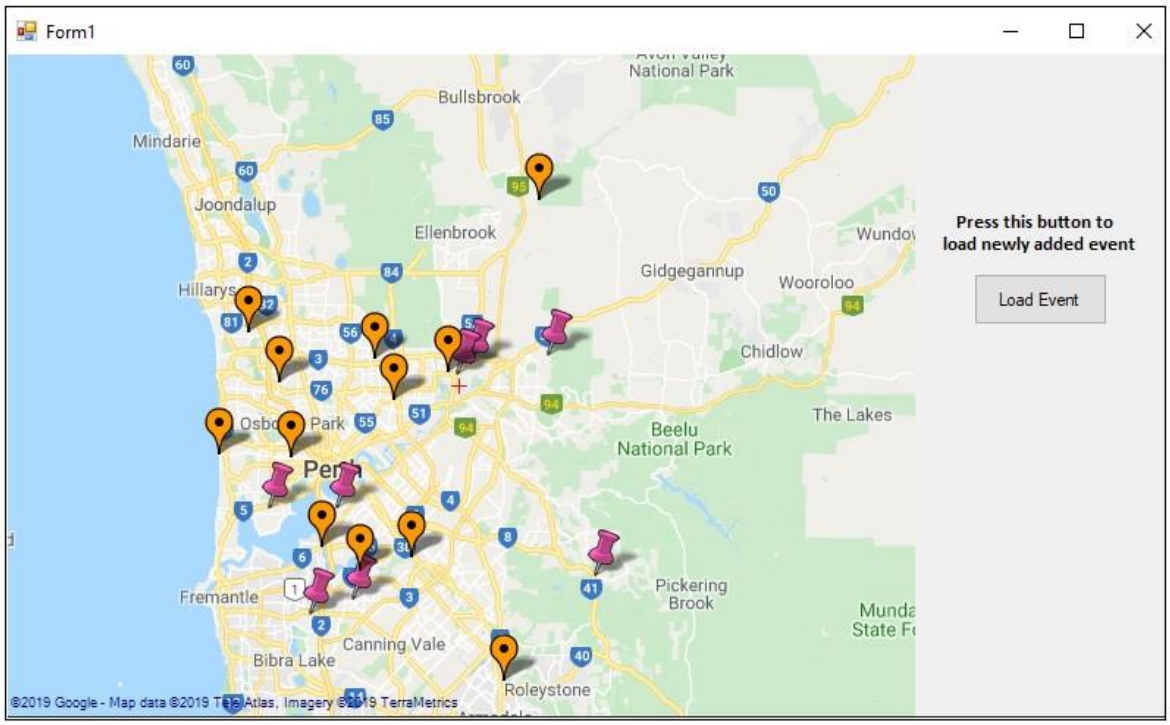

Figure 1: Displaying events on a map

Events from the SOAP file were read into a C# Dictionary using LINQ to XML. In C#, Dictionary is a collection that use to store key/value pairs. It is dynamic in nature because its size grows according to the requirement. In a Dictionary, the keys are always unique and not null as they are used to store and retrieve values from it. Only same type of elements can be stored in a Dictionary, and the size of a Dictionary is determined by the number of elements stored in it.

```
Dictionary<string, Events1> Dic1 = new Dictionary<string, Events1>();
Dictionary<string, Events2> Dic2 = new Dictionary<string, Events2>();
```

Figure 2: C# Dictionaries

The user is presented with an option to Add a new event, or Retrieve an event information by clicking on the canvas map. A pop-up box appears with buttons for making a selection to either add an event or inspect event.

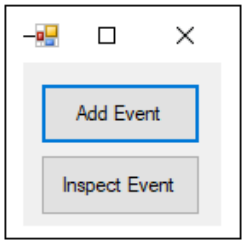

Figure 3: Choice box

When user presses the Add Event button, a drop-down list appears where the user can select the type of event, he/she wants to add. The different event types are photo, video, tracklog, facebook-status-update and tweet.

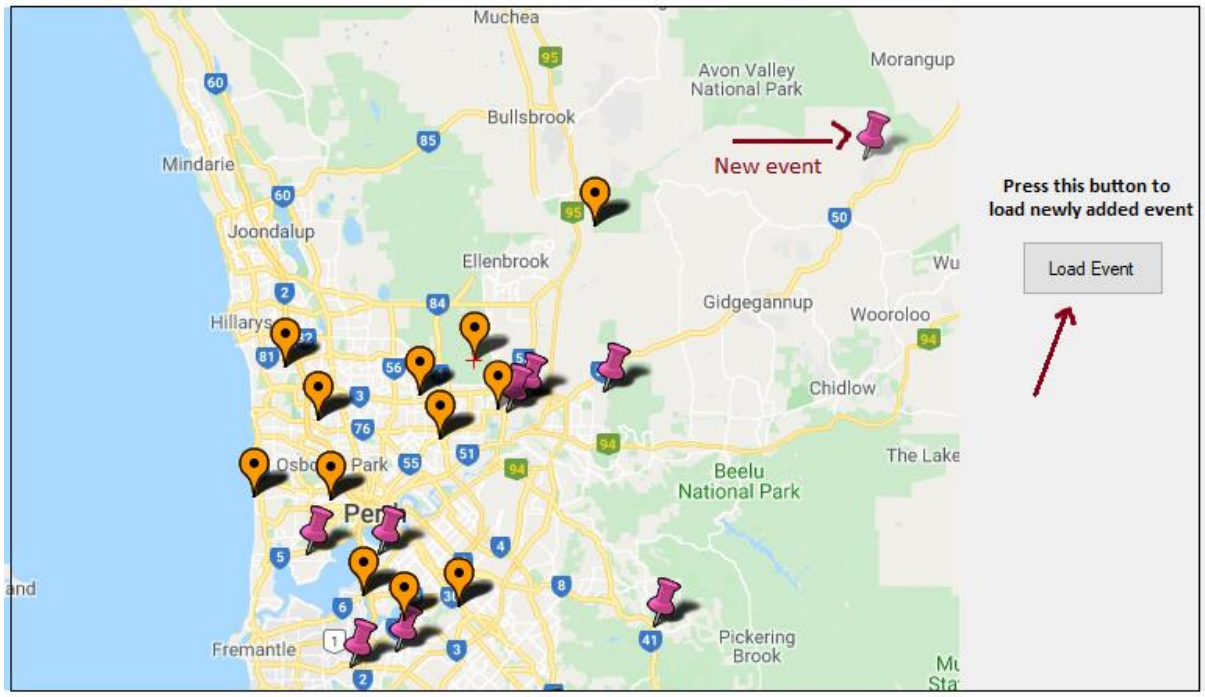

Figure 4: Adding new event

The new event information is added to Dictionary as well as to a XML file. After providing new event information, the Load Event button needs to be pressed to reload the canvas map.

```
<?xml version="1.0" encoding="utf-16"?>
<Event>
  <eventid>ID27</eventid>
  <tweet>
    <text>hello twitter</text>
    <location>
      <lat>-31.646367146916376</lat>
      <lng>116.27105712890625</lng>
    </location>
  </tweet>
</Event>
```

Figure 5: New event written to XML file

To retrieve information about an event, the user can simply click around the event and select Inspect Event button from the choice form. A sample of few event retrieval is shown in the figures below.

Event ID: ID6
Event type: tweet
Latitude: -31.981800
Longitude: 115.863710
Text: Check this tweet out
Date timestamp:

Figure 6: Tweet event

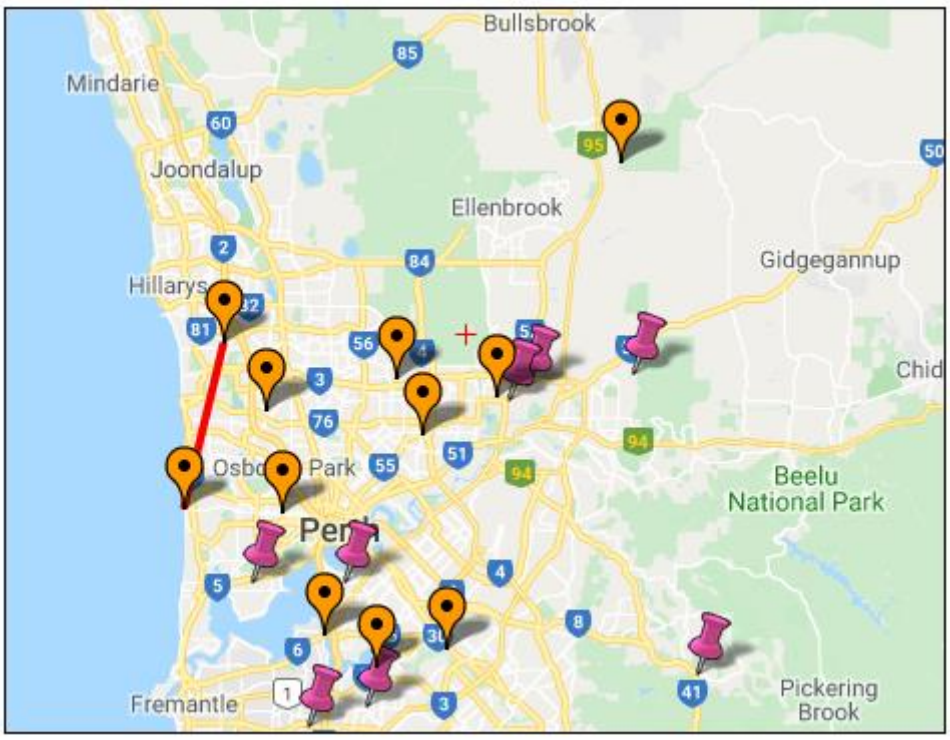

Figure 7: Tracklog event

Figure 8: Video event

The instances were LINQ to XML was used in this solution are as follows:

1) When reading event information from the SOAP XML file into the dictionary.
2) Finding least distance value from a Dictionary that contains the distance of various events from a given point on the point.
3) Writing event information to XML file.

The .NET Framework uses XML extensively, and LINQ to XML provides a way to manipulate data in XML documents using the same syntax on arrays, collections, and databases. LINQ to XML also provides a set of classes for easily navigating and creating XML documents in your code. Each XML element name is represented by a node, and a node that contains other nodes is called a parent node. Nodes that have the same parent are called sibling nodes.

The Namespace **System.Xml.Linq** contains the classes used to manipulate a DOM in .NET, referred to collectively as LINQ to XML. The **XElement** class represents a DOM element node, and the **XDocument** class represents an entire XML document. The Xdocument's Elements method can return all child elements, or only elements with a given tag name. The **Descendants** method returns all descendant elements with the given tag name, not just direct children.

There are two ways to read XML using LINQ to XML: Using the Xelement class or the XDocument class. Both the classes contain the 'Load()' method which accepts a file, a URL or XMLReader and allows XML to be loaded.

The XDocument class is used when you need to create an XML document containing XML declaration, XML Document Type (DTD) and processing instructions, comments or namespaces [6].

Evaluation

Fully implemented and fully working:

- Added at least 20 more events into the SOAP XML file.
- At least two tracklog type events added to the list of events.
- Created Windows form GUI using Visual Studio.
- Implemented the canvas map to display the vents from the SOAP file.
- Use of LINQ to XML to load the data from the SOAP file into C# Dictionary.
- Developed two different classes (Events1 and Events2) for the events.
- Displayed the events using suitable markers on the map.
- Giving user the option to Add a new event, or Retrieve information by clicking on the map.
- Use of a sub-form to obtain the relevant information about the event.
- Providing event information and associated media when retrieving an event.
- Drawing lines to represent the tracklog by reading coordinates from a GPX file.

Not fully working:

- New events are not written in SOAP file format, rather they are written simply in XML format.
- Inconsistency in adding events of type “tweet” and “facebook-status-update”. New events may not get plotted on the map in the first attempt, but works properly in the subsequent attempts. This seems to be an issue in refreshing the map because even in the first attempt the event information is added to the Dictionary but fails to plot its marker.

Part 1	**Tasks**	**Evaluation**
	Task 01	Fully Completed
	Task 02	Fully Completed
	Task 03	Fully Completed
	Task 04	Fully Completed
	Task 05	Fully Completed

References

[1] "ASP .NET Notes." http://sce2.umkc.edu/BIT/burrise/it222/notes/aspdotnet.html (accessed 27/10, 2019).

[2] tutorialspoint. "ASP.NET - Introduction." https://www.tutorialspoint.com/asp.net/asp.net_introduction.htm (accessed 27/10, 2019).

[3] Microsoft. "Migrating WPF Apps to .NET Core." https://docs.microsoft.com/en-us/dotnet/desktop-wpf/migration/convert-project-from-net-framework (accessed 27th-10, 2019).

[4] GeeksforGeeks. "Introduction to C# Windows Forms Applications." https://www.geeksforgeeks.org/introduction-to-c-sharp-windows-forms-applications/ (accessed 27th-10, 2019).

[5] GeeksforGeeks. "Introduction to ASP.NET." https://www.geeksforgeeks.org/introduction-to-asp-net/ (accessed 27th/10, 2019).

[6] C. S. U. L. Beach, "INTRODUCTION LINQ to XML," 2019.