
Zusammenfassung der Arbeit

Abstract of Thesis

Fachbereich:

Department:

Electrical Engineering and Computer Science

Studiengang:

University course:

Information Technology

Thema:

Subject:

Mobile Phone Cross Platform App development using C# and Xamarin Forms

Zusammenfassung:

Abstract :

With the development and popularity of the mobile devices, cross-platform development is becoming a trend. With the support of Visual Studio, the tool Xamarin.Forms bring developers with a feasible solution to create an application which could run on Android, iOS, and UWP (Universal Windows Platforms) platforms. The purpose of this thesis is to investigate the possibilities and limitation of the tool Xamarin.Forms. To reach the final goal, a new application called “Event Manager” was created, which was developed based on previous students’ thesis result “Location Finder”. It enables event organizers to publish events with detail information and it allows the event participants to search the events according to certain conditions. Besides, some subtasks were completed to test the possibility of Xamarin from other perspectives. Furthermore, the application was published to three platforms with the functions provided by Xamarin.Forms and Visual Studio. This thesis contains the detail procedures how old application “Location Finder” is adapted and integrated, and how the application “Event Manager” is implemented according to the specification. The functional implementations are described comprehensively. In addition, the application publishing procedures are introduced. At the end of the thesis, based on the developing experience of application “Event Manager” and the successful realization of subtasks, some conclusions are drawn that Xamarin.Forms do provide developers with powerful and reliable cross-platform developing method. However, it still has some drawbacks which could affect its performance.

Verfasser:

Author:

Linhui Yang

BetreuerProfessor/in:

Attending Professor:

Prof. Dr. Jörg Bayerlein

WS / SS :

SS <2018>

Table of Content

Table of Content.....	v
1 Introduction.....	1
1.1 Topic Description.....	1
1.2 Motivation.....	1
1.3 Thesis Structure.....	1
2 Background Information and Knowledge.....	3
2.1 A brief introduction to Xamarin and its Products	3
2.2 Code Sharing Methods in Xamarin.Forms.....	3
2.3 The .NET Standard	4
2.4 Changes in Visual Studio 2017.....	5
3 Software installation and configuration.....	6
3.1 Visual Studio and MySQL Workbench Installation	6
3.2 Emulator configuration	8
3.2.1 UWP Emulator Configuration.....	8
3.2.2 Android Emulator Configuration	8
3.2.3 iOS Emulator Configuration.....	10
3.3 Connect with iOS Device	12
4 Approaches for development.....	14
4.1 Page Creation	14
4.1.1 Content page generation	14
4.1.2 Tabbed page generation	14
4.2 Layouts	15
4.2.1 Stack Layout.....	15
4.2.2 Scroll View	15
4.2.3 Grid Layout	15
4.3 View Components	16
4.3.1 Switch	16
4.3.2 Picker	16
4.3.3 Time and Date Picker.....	17
4.4 App Class	19
4.5 IDictionary interface.....	19
4.6 Navigation Methods.....	20
5 The Implementation of the Application “Event Manager”	20

Table of Content

5.1	Project Description	20
5.2	Use Case Diagram	21
5.3	Adapt and integrate previous students' result	21
5.4	Results of "Event Manager"	30
5.5	Realization details	34
5.5.1	Language preference saving	34
5.5.2	User account preference saving	35
5.5.3	Location information	37
5.5.4	Dynamic adding items into Picker	39
5.5.5	Database layer	40
5.5.6	Data operation and filtering	41
5.6	Subtask1: Pop-Up Menus	45
5.6.1	Using "DisplayActionSheet"	45
5.6.2	Rg.Plugins.Popup (1.0.4)	48
5.7	Subtask2: Publishing Application	51
5.7.1	Publishing Via Google Play	51
5.7.2	Publishing Via Apple Store	56
5.7.3	Publishing Application Privately in Win10	65
5.8	Errors reporting	68
6	Evaluation	70
7	Summary	71
7.1	Possibilities and limitations of Xamarin.Forms	71
7.2	Conclusion and Outlook	71
	Acknowledgment	73
	Appendix 1 List of Figures	74
	Appendix 2 List of Table	76
	Appendix 3 List of Code	76
	Appendix 4 Content of USB Sticker	77
	Appendix 5 Reference	77

1 Introduction

1.1 Topic Description

In this thesis, the Xamarin.Forms are used to program a Cross-platform application which based on C#. The app tends to be running on three different platforms: Windows10 UWP (Universal Windows Platform), Android and iOS devices.

The final goal of this thesis is to find the possibility and limitation of Xamarin in cross-platform application developing. To reach this task, previous students' thesis result "Location Finder" was loaded and modified. Based on the previous application, the new application "Event Manager" was developed. This application could be divided into two parts, which have two target groups. The first part is for Event Organizer. The Event Organizer should register for the application, and they could easily publish the events with detail information. The second part is for event participants. The event participants do not need to register an account. They could search for the events according to the requirements (time, place, type of events). During the development phase, some subtasks are investigated and applied in the application. After the development, the method for publishing the application is researched. In this thesis, the methods for distributing application via Google Play/ Apple Store will be introduced.

1.2 Motivation

Cross-platform application development has some natural features. Combined with Visual Studio and Xamarin some more possibilities could be discovered. Several reasons can be given to explain why to choose Xamarin to program application.

Firstly, the developers only need to program one user interface, and it can be displayed in three platforms. What's more, the interfaces are native appearance. No need to program in the platform-specific language. Besides, one code logic implementation may run on multiple platforms could save developing time on a large scale.

Secondly, the platforms supported by Xamarin are popular now. Android and iOS take a large part of the mobile device market shares. In another word, the majority of the people in the world have a mobile device which uses these operating systems. So, the applications built by Xamarin can be widely used.

Thirdly, the difference of programming on different platforms can be compared. To some extends, it can also lay the foundation for my future studying.

Fourthly, Xamarin.Forms allow developers to develop the application with C#. For some developers who are familiar with Java, C, and C++ the learning costs could be rather low.

Furthermore, C# is suitable for mobile app development. Some features of C# are powerful. For example, it provides Language-Level Async (Asynchronous programming) which the overall responsiveness of the application can be enhanced[1].

1.3 Thesis Structure

This thesis is divided into several parts to help the reader understand what the author has researched and implemented.

In chapter 1 and chapter 2 a short impression of the thesis is provided, and the background knowledge of Xamarin is introduced.

Then in chapter 3, some application installation steps and software configuration procedures are introduced.

In chapter 4, the approaches designed to construct the application are demonstrated. It could help readers to understand the primary usages of Xamarin and Visual Studio and make readers have a rough understanding of the application “Event Manager”.

In chapter 5, the detail implementation of the application “Event Manager” and other subtasks will be introduced. Besides, some problems occurred during the implementation will also be described.

Finally, in the chapter the possibilities and limitation of the Xamarin.Forms are analyzed, and some further conclusions are drawn.

2 Background Information and Knowledge

2.1 A brief introduction to Xamarin and its Products

Xamarin is a software company which is owned by Microsoft. [2] In 2014 they published the Xamarin.Forms which is a powerful cross-platform programming tool integrated into the Microsoft Visual Studio.[2] Moreover, Xamarin.Forms have several features.

Firstly, Xamarin.Forms is a cross-platform natively backed UI toolkit abstraction that allows developers to easily create user interfaces that can be shared across Android, iOS, Windows, and the Universal Windows Platform.[3] Xamarin.Forms provide native controls of the target platforms which are used to create the user interface.[3] That means the application can share a significant portion of their user interface code. What's more, the native look and feel of the target platform will still be retained.[3] So, the developers can create natural appearance iOS, Android, and UWP applications.

Secondly, one code logic could be implemented on three platforms. A layer of C# encapsulation made by Xamarin.Forms which allows developers to call the native API of iOS and Android, which means an application could be developed in C# language environment and also could run in iOS, Android and UWP platform. The developers don't need to have deep understanding in Java or Objective-C but also can develop the iOS or Android Applications.[3]

Thirdly, two techniques are provided by Xamarin.Forms for developers to generate User Interface. The first technique is to use complete C# source code for user interface design. The second technique is to use XAML (Extensible Application Markup Language) to describe the User Interface.[3]

There are some differences between XAML and C#. C# lays more emphasis on defining a process while markup language focus on defining a state. What's more, the markup language has some intrinsic drawbacks when compared with code: no loops, no flow control, no algebraic calculation syntax and no event handlers. [4]

In this thesis, the XAML is not the point we want to focus. Compared with C#, XAML lays more emphasis in the Presentation Layer. It is convenient for developers to create front-end part, but the business logic part still needs to be implemented by C#. To maintain the uniformity, this thesis focuses on coding by C#.

2.2 Code Sharing Methods in Xamarin.Forms

As a Cross-Platform programming tool, sharing code is an important feature. In this part, the concept of these three methods and the advantages and disadvantages will be introduced according to the definition given by Microsoft.

Three strategies are provided by the Xamarin: Shared Asset Project (SAP), Portable Class Libraries (PCLs) and .NET Standard Libraries. [5]

In SAP, a shared project will be created and the source code will be used on different platforms. The source code contains the majority implementation part of different platforms. For platform-specific implementation, platform compiler `#if __(platform)__` could be used. However, when a platform wants to compile the project, the whole shared project will be compiled into an assembly which will be unnecessary. Besides, the code inside the compiler directives will not be updated by refactoring. [5]

In PCL, a portable project will be generated, and the source code will be shared on different platforms. The shared source code is used for User Interface. Some business logic implementation part is realized in specific-platform projects. A shared interface created in the portable project, and it will be implemented in these platform projects. However, the compiler directives cannot be used in PCL. What's more, only the subset of .NET Framework can use PCL. [5]

In .NET Standard Libraries, similar to PCL, a portable project will be generated and the source code will be shared in different platforms. It has the advantages of the PCL and has fewer limits than PCL. The code can be shared on different platforms, and the refactoring operation could update the all related reference. Besides, the .NET base library BCL is much powerful than PCL. [5]

2.3 The .NET Standard

With the development of the Xamarin and the .NET, the .NET Standard has replaced the PCL as the main-stream Code Sharing Strategy. In the latest version of Visual Studio 2017(15.6.6), only Shared Projects or .NET Standard could be chosen as the Code Sharing Strategies when a cross-platform Project is created.

The Application “Event Manager” is created by strategy .NET Standard. So, in this section, the information about .Net Standard will be introduced.

According to the definition of .NET Standard given by Immo Landwerth, the .NET Standard is a set of APIs that all .NET Platforms have to be implemented. The latest .NET Standard 2.0 will be implemented by .NET Framework, .NET Core, and Xamarin. It includes a compatibility shim for .NET Framework binaries, which means that the set of libraries can be referenced from .NET Standard libraries are increased on a large scale. What's more, the PCL will be replaced by .NET Standard as tooling story for building multi-platform .NET libraries. [6]

In previous years, the .NET platforms forked a lot[6]. For example, in framework .NET Core and Xamarin, they have different base libraries. The .NET Core has Core Library as its base library, and Xamarin has Mono Class Library. To some extends, the platform-specific base libraries make it easy for the developer to create an application in different platforms, but it disobeys the original intention for unity. So, the .Net Standard was introduced. This library support three Framework .NET Framework, .NET Core, and Xamarin. Thus, the uniformity is enhanced.[6]

Compared with PCLs, the .NET Standard guarantees the consistency in the APIs. [6]The PCLs uses APIs which support multiple platforms, but there is a problem. The available APIs are the intersection set of these platform-specific platforms. What's more, it is hard to define the API when it comes to a specific combination of platforms. However, the .NET Standard is a comprehensive API. To a certain degree, .NET Standard can be deemed as the union set of those specific-platform APIs. It means that all .NET platforms could execute the Application which uses the .NET Standard library.[6]

The Version of .NET Standard

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0

.NET Framework (with .NET Core 2.0 SDK)	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299
Windows	8.0	8.0	8.1					
Windows Phone	8.1	8.1	8.1					
Windows Phone Silverlight	8.0							

Table 2.3.1.NET Implementation Support [7]

This is the table shows the generations of the .NET Standard. The elements which below the .NET Standard is the name of the frameworks, the following numbers are the versions of the corresponding frameworks which are supported by the .NET Standard. [7]

With the development of the .NET Standard, the size of the APIs supported by .NET Standard is increasing, but the supported platforms are decreasing. It is a problem to the developer. As a result, before creating an application, the version of the .NET Standard should be considered. Otherwise, the incompatibility problem will occur. [7] For example, the Microsoft decided to stop supporting the development of Windows Phone, Windows Phone Silverlight and Win 8.1. So, starting from .NET Standard 1.3, the APIs for these three platforms are removed. [7]

2.4 Changes in Visual Studio 2017

Obsolescence of Components

When the Application “Location Finder” was loaded in the new version of Visual Studio 2017 Professional, the Components section in each Platform project is missing. An alert window jumped out and claimed that the current solution is using Xamarin Components which is no longer supported.

According to the explanation by Joseph Hill, with the development of the .NET ecosystem, there have been tens of thousands of packaged .NET libraries in the NuGet. Thus, the functionality of the Xamarin Component Store decreased on a large scale. As a result, it has been integrated into the NuGet which has better adaptivity in .NET Ecosystem.[8]

3 Software installation and configuration

3.1 Visual Studio and MySQL Workbench Installation

The Visual Studio 2017 and MySQL workbench are needed for developing an application in the thesis. The following pictures show the version and the components which are needed to be installed.



Figure 3.1.1 Version of Visual Studio 2017

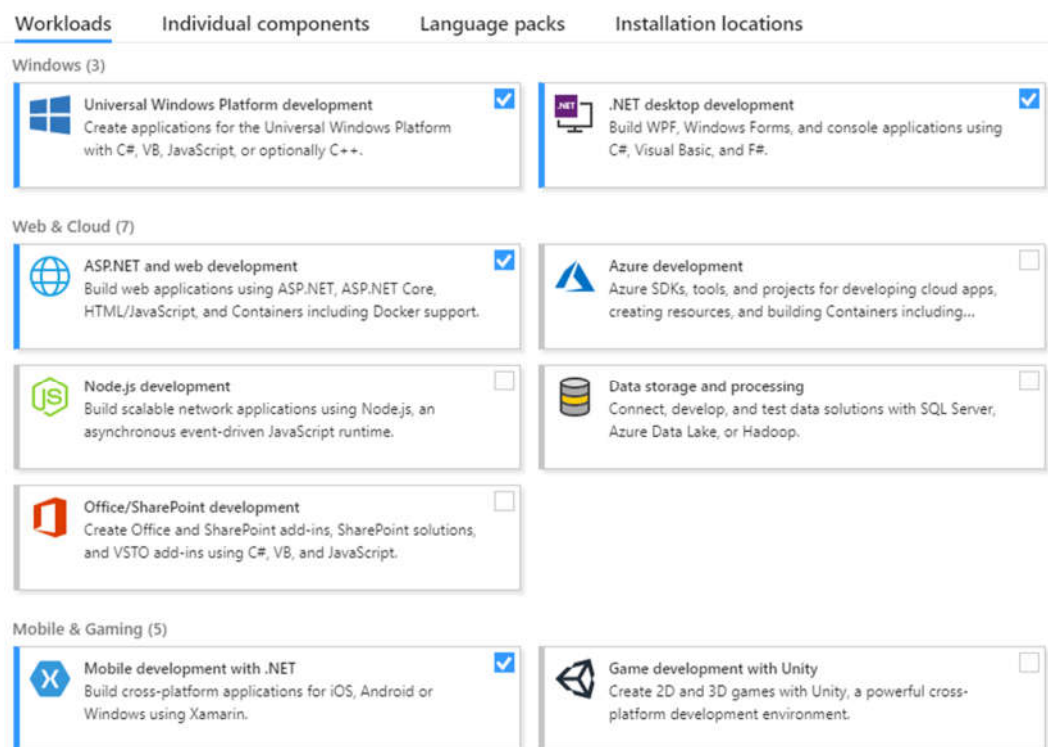


Figure 3.1.2 Installed Component in Visual Studio 2017

The Visual Studio 2017 professional is available at the website [9]

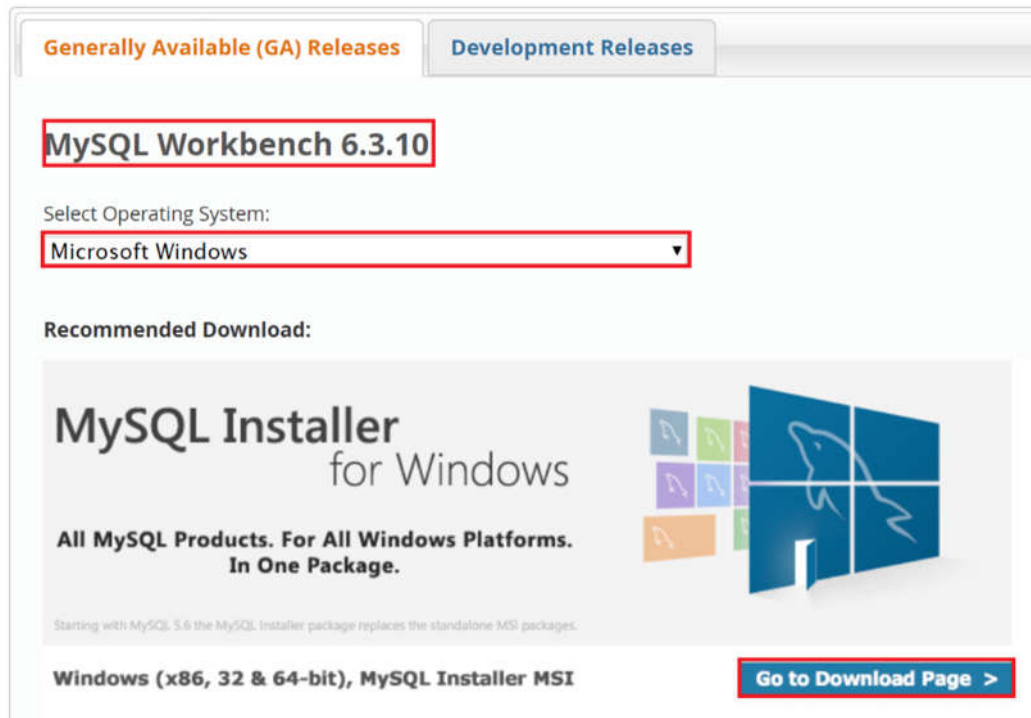


Figure 3.1.3 MySQL Workbench Installation

The MySQL Workbench is available from the website:[10].

In this thesis, a remote database is used. In the following steps, the database connection will be introduced.

First, add a new MySQL connection. Then enter the connection information of the database.

The places marked should be filled.

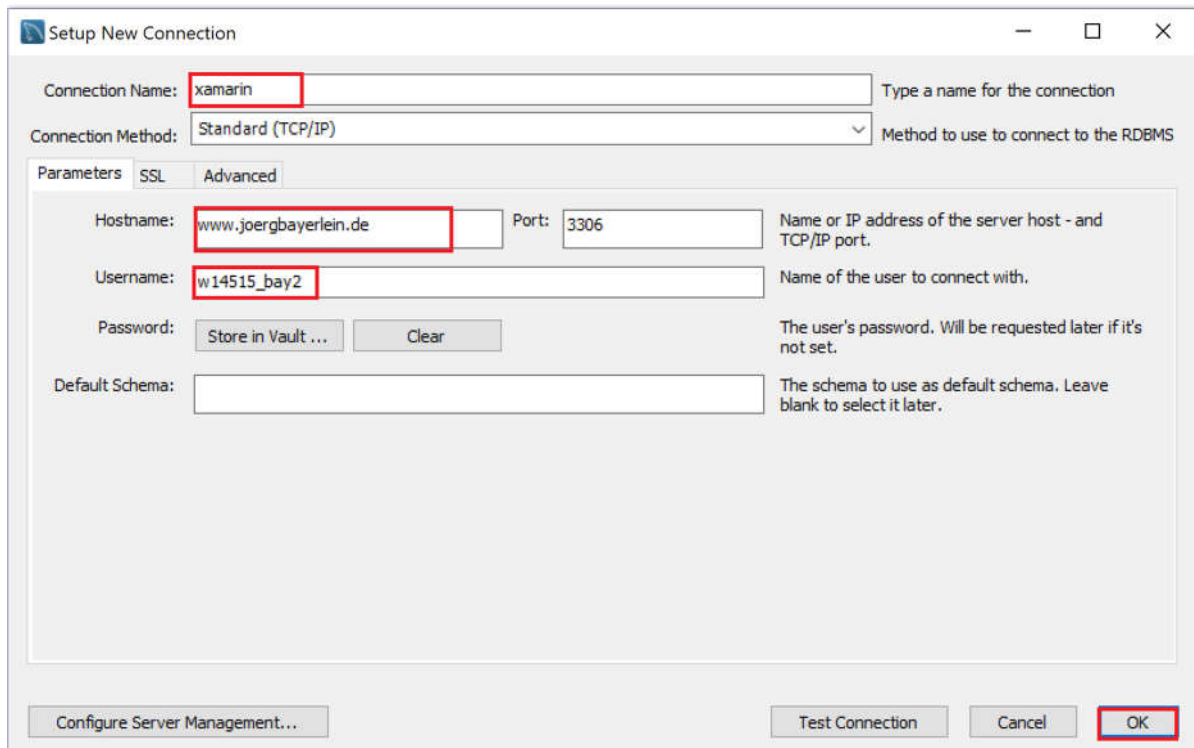


Figure 3.1.4 MySQL database Connection

3.2 Emulator configuration

3.2.1 UWP Emulator Configuration

In the previous students' result, the Win Mobile Emulator was installed for debugging the application. However, with the development of the .NET Standard and the change of the Microsoft strategies, the Win Mobile has been abandoned. Moreover, in the latest Windows 10 Fall Creator Update (10, Build 16299), the APIs for Window Mobile have been deleted. As a result, in the debugger bar, there are no emulators for Window phone. If the developers still want to use the Windows Phone Emulator, it is possible to use the Win Mobile Emulator by changing the Min Version of the target SDK to Window 10 Creators Update (10, Build 15063).[11]

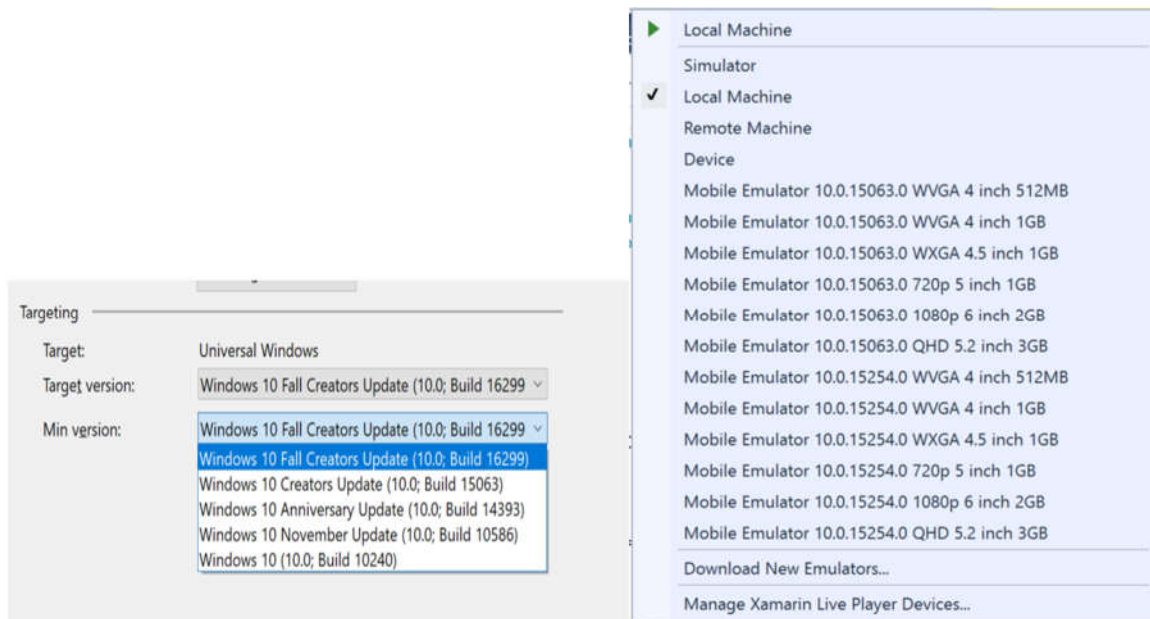


Figure 3.2.1 UWP Targeting SDK

Figure 3.2.2 UWP Emulator (Win 10 SDK 15063)

In this thesis, the current latest version of Win 10 SDK 16299 is used, so the Win Mobile Emulator will not be used in this thesis. Instead, the emulator provided by local machine is used which is available in build 15063 or higher.

3.2.2 Android Emulator Configuration

In the visual studio, there are two kinds of Android Emulators. The first one is HAXM-based Android emulator. The second one is the Hyper-V-based Android Emulator. The first way requires CPU to support the HAXM hardware acceleration, and the Hyper-V should be stopped when HAXM is running which means the UWP emulator can't run. The second way is better than the first way because of the easier configuration. So, in this thesis, the Hyper-V based Emulator installation procedure will be discussed.

In the Visual Studio, the android emulator has its manager application. It can be download in Visual Studio Installer. Go to the "Individual Component" section, searching the emulator and choose "Visual Studio Emulator for Android" then upgrade the Visual Studio.

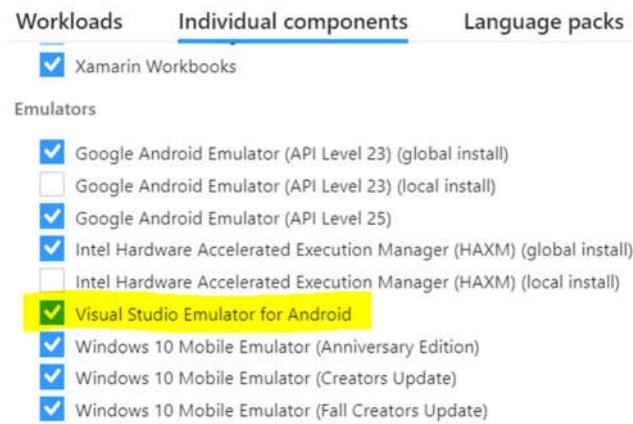


Figure 3.2.3 Visual Studio Emulator for Android installation

Open the Visual Studio Emulator for Android, and then download the Android Emulator. In this thesis, Emulators with API 19 and API 23 has been downloaded for debugging. (API 23 is the highest version for Android Emulator in this emulator management application)

Because it is a Hyper-V based emulator, so there are some configurations need to be set in the Hyper-V manager. After the emulators have been installed, the emulator will appear in the menu of Hyper-V manager.



Figure 3.2.4 Hyper-V manager

The version of the Emulator might have some conflicts with the operating system. It will result that the Android debugger crash during the process of deploying the application. To avoid this problem, go to the emulator setting pages, click the Processor and then go to the "Compatibility". In the "Compatibility", the checkbox should be checked.

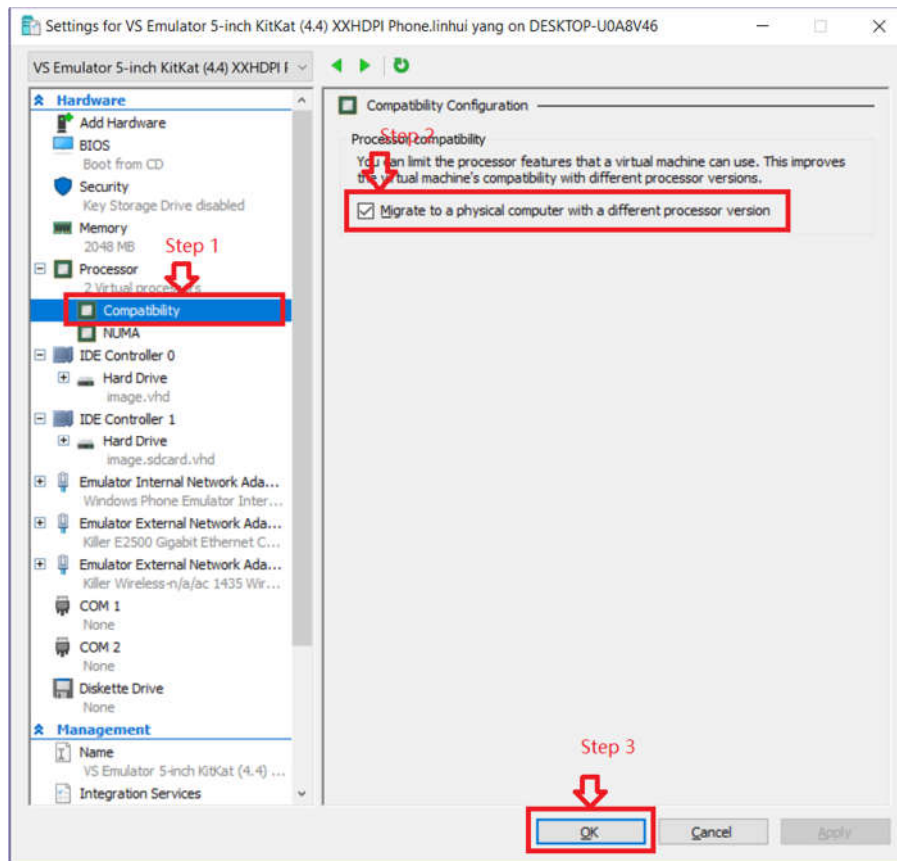


Figure 3.2.5 Hyper-V manager Emulator Setting

3.2.3 iOS Emulator Configuration

To use an iOS emulator to debug the application, a Mac is needed as the iOS application compiler. Some configuration steps should be completed in both Mac and window machine.

Step 1. Installation of Xcode

It is available in the apple store. The version of the Xcode should be Version 9 or higher. Besides, the release of the operating system should be OS10 or higher.

Step 2. Set permission for connection

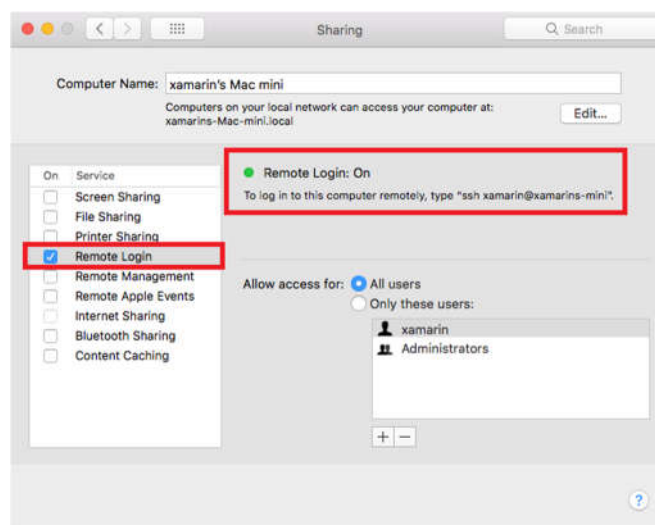


Figure 3.2.6 Set Permission for Remote Login

In Mac choose Sharing in the preference, allow remote login.

Step 3. Connect the Visual Studio 2017 with the Mac

Click the “Pair to Mac” button in the iOS Toolbar. Then choose a mac to connect. If there is not available mac on the list, adding the IP address of the mac manually.

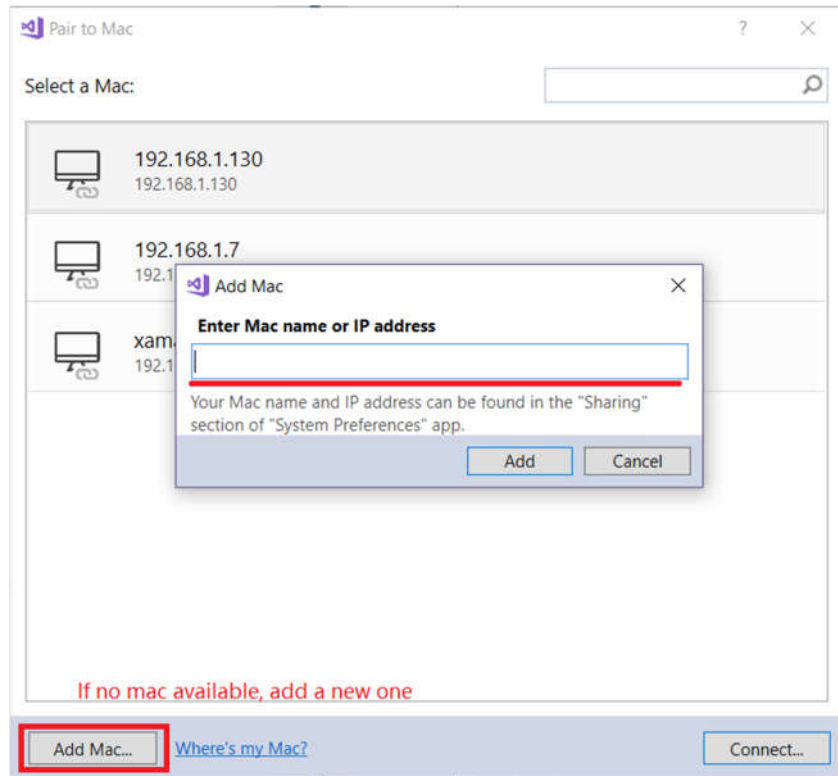


Figure 3.2.7 Pair to Mac

Remote simulator to windows

With the improvement of the Visual Studio, the developer could run Remote Simulator on Windows. Actually, the Visual Studio projects the iOS simulator on MAC to the Windows when a MAC is paired to the Visual Studio on Windows. It provides some convenience for the developer because the debugging procedure could be completed in one operating system. There is no need to operate a MAC. However, there is still one problem. Opening iOS simulator on the Windows takes quite long time. What's more, the operation in Windows is not as smooth as in OS. There exists a massive delay due to the operating system and network connection.

The way to open iOS simulator on Windows will be introduced.

Firstly, go to the “Tool” in the Window Visual Studio toolbar, then click the Options.

Secondly, go to the “iOS Setting”.

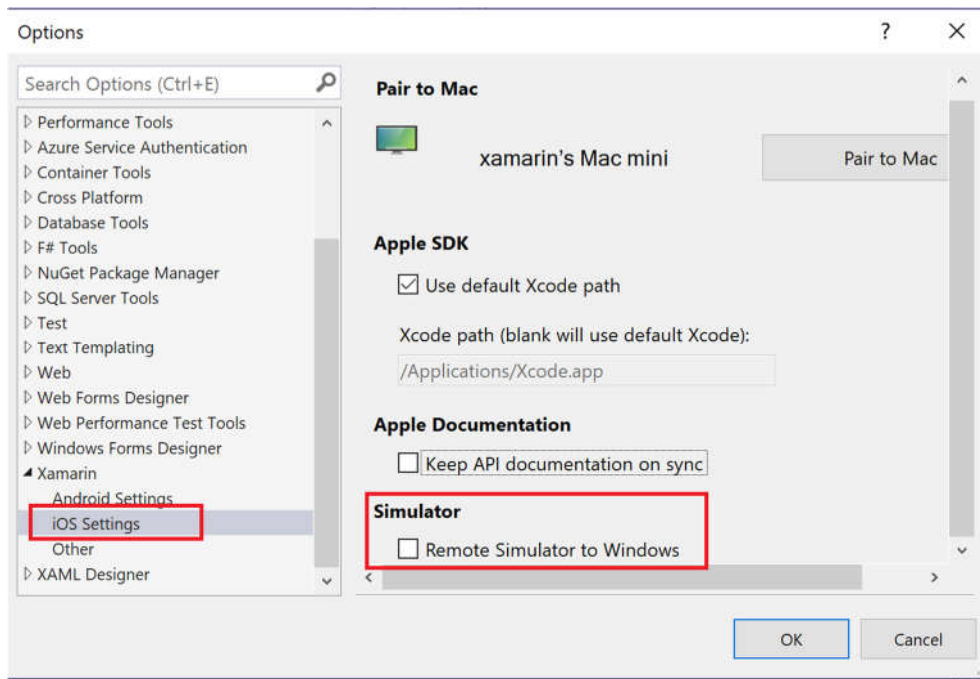


Figure 3.2.8 iOS Setting for Remote Simulator to Windows

Check the Remote Simulator to Windows box; the simulator will show in the Windows when the application is about to be debugged.

3.3 Connect with iOS Device

A physical device is needed for debugging and seeing how the application run on a real device. Some configurations should be set to connect an iOS device. The sample application “ReuterIdent” will be used to introduce the connection procedures.

Step 1. Create a blank project in Xcode

Step 2. Pair the Bundle Identifier

Firstly, open the *Info.plist* in the Visual Studio. Set the application name and Bundle Identifier. In a default situation, the application name is identical to project name. Then go to the project in the Xcode. Make sure the Bundle Identifiers in both projects the same.

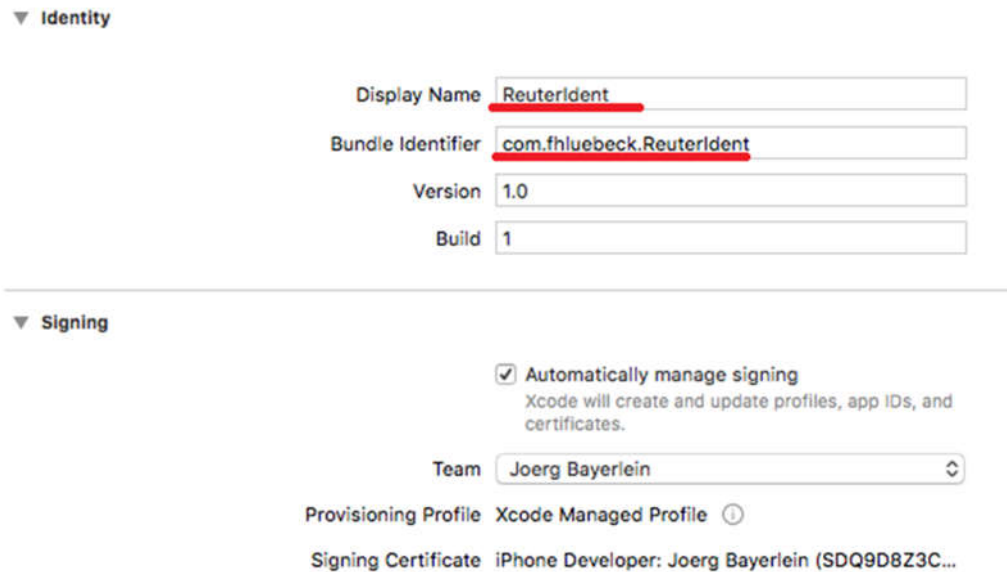


Figure 3.3.1 Bundle Identifier in Xcode

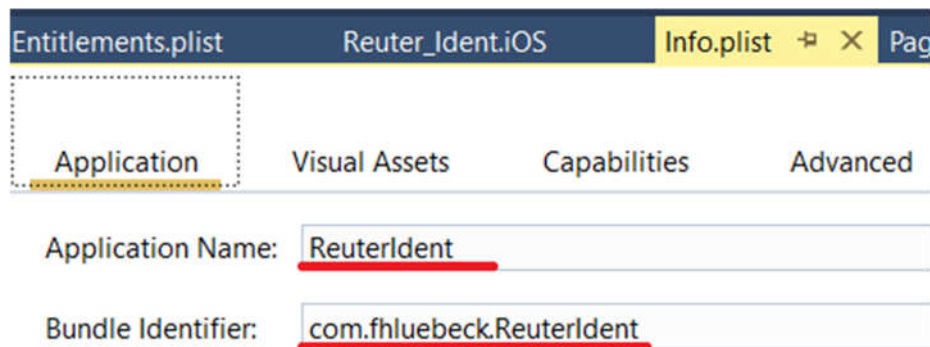


Figure 3.3.2 Bundle Identifier in Visual Studio

Step 3. Signing the project

An apple account is needed for signing the project. In this thesis, an apple developer account was registered. Besides, a signing certificate was necessary. In the Xcode, go the Account, then choose an Application ID. After that, click the button “Manage the Certificate” and add an iOS developing certificate manually. Finally, restart the Xcode to ensure the certificate is active. After that, the device connected with Mac can be shown in the toolbar of the Visual Studio.

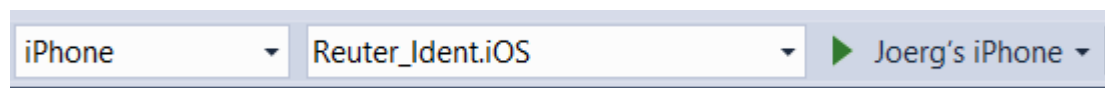


Figure 3.3.3 Connect real device

Click the run button; the application will be installed on the device. To debug the application, the debug permission on the device should be allowed. In the Setting of the iPhone, choose the application, then enable the debug operation.

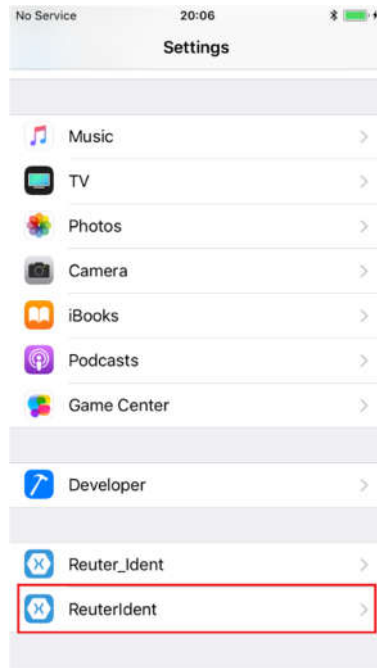


Figure 3.3.4 Enable Device Debug 1

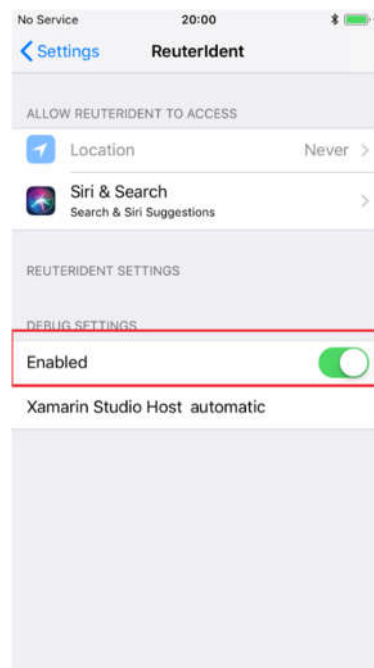


Figure 3.3.5 Enable Device Debug 2

Then the application can be run and debugged on the real device.

4 Approaches for development

4.1 Page Creation

Xamarin provides several types of pages: Content Page, Master Detail Page, Navigation Page, Tabbed Page, Templated page and Carousel page. The Content page and Tabbed page are decided to be used in developing phase. In this part, the simple syntax for creating these two kinds of pages will be introduced.

4.1.1 Content page generation

The content page is a standard page used for representing the information and for user interaction.[12] The Visual Studio provides methods for a developer to generate a Content page automatically.

Firstly, right-click the project, then click the add. After that choose New Item. Secondly, select the Content page in the menu. Then create it.

4.1.2 Tabbed page generation

A tabbed page will be used in this project. It is a page which has several tags on the top or button of the pages (depends on platforms). The user could go to different pages by clicking the navigation tags. Each tag represents one Content page, what's more, an icon could be added at the tag bar. [13]

Firstly, create a simple Content page. Secondly, change the Content Page class to Tabbed Page class. Finally, adding content page item to the Tabbed page.

```
1. public class test_tabbedPage : TabbedPage
2. {
3.     public test_tabbedPage ()
4.     {
5.         var navigationPage = new NavigationPage(new Page1());
6.         navigationPage.Icon = "resource.png";
7.         navigationPage.Title = "Tag1";
8.
9.         Children.Add(navigationPage);
10.        Children.Add(new Page2());
11.    }
12. }
```

Code 4.1.1 Create a tabbed page [13]

In the tabbed page, the icon and the title of the tab page could be defined. [13]

4.2 Layouts

In the mobile application, choosing the right layout is important. The proper layout should be decided to fit in the screen of the mobile devices. In the following parts, the layouts used in the application will be shortly introduced. [14]

4.2.1 Stack Layout

In the stack, a baseline is set in the center of the screen. The views can be set horizontally and vertically based on that stack line. It is a simple linear layout interface that could nest other layouts to create a complex view. [15]

4.2.2 Scroll View

The scroll view is designed to guarantee that the large view can be shown on the small screen. It allows all layout views to automatically move to the visible portion of the screen when the user scrolls the screen by their finger or some other peripherals. [16]

4.2.3 Grid Layout

Grid layout divides the view into rows and columns. The row and column have proportional size and absolute size. It helps the developer to arrange the position of the items in the view. For example, the position of the confirm button and cancel button should be placed at opposite position which can be perfectly arranged by grid layout. What's more, it can guarantee the size of different items is identical. It can make the view tidy. [17]

4.3 View Components

The mainly used view components in the application will be introduced in the following parts to give readers a basic concept of the application “Event Manager”.

4.3.1 Switch

According to the definition given by the Charles Petzold, the switch is a view component required by the application to get the Boolean input from users. The switch only defines one property *IsToggled* of type *bool*. The property’s change will trigger the events. In the following implementation phase, the usage of the switch will be described in detail. [18]

4.3.2 Picker

The Picker is used to show a list of items and user could pick an element in this view list. [19]

```

1. //get a Dictionary from other resources, or create a Dictionary in the f
   following
2. static Dictionary<int, string> sample;
3. ...
4. picker = new Picker
5. {
6.     Title = "Test Picker",
7.     HorizontalOptions = LayoutOptions.Center,
8.     VerticalOptions = LayoutOptions.Center,
9. };
10. };
11. for (int i = 1; i < sample.count()+1; i++)
12. {
13.     picker.Items.Add(result[i]);
14. }
15. //set the default item
16. picker.SelectedIndex = 0;
17. //remove item by its name
18. picker.Items.Remove("items name");
19. //item's change trigger the event
20. picker.SelectedIndexChanged += Picker_SelectedIndexChanged;

```

Code 4.3.1 Sample code of Picker

The basic method of application will be introduced by example code.

A Picker contains a title, it can be instantiated in the constructor, and the position of the title can also be set in the constructor. In the implementation phase, the item resources come from Dictionary. If the number of items in the dictionary is unknown, it can be added to picker by a loop. The item resources can also be obtained from a string array. After that, the default items could be set. Otherwise, it will be left blank in the picker. The item in the picker can also be removed by the name of items. Besides, when the selected item changed the event could be triggered. For example, a new page will be pushed. [19]

4.3.3 Time and Date Picker

In previous student's results, the time picker part was not implemented, and in the database, the starting time of the event and the ending time of the event are left blank. The time information of the event is one of the most important details of the "Event Manager". So, in the following part, the method to create Time and Data Picker will be discussed.

The Time Picker can be divided into two parts: the first part is how user choose the time. The second part is how to get the data from the presentation layer.

Screenshots of Time/Date Picker

1. UWP

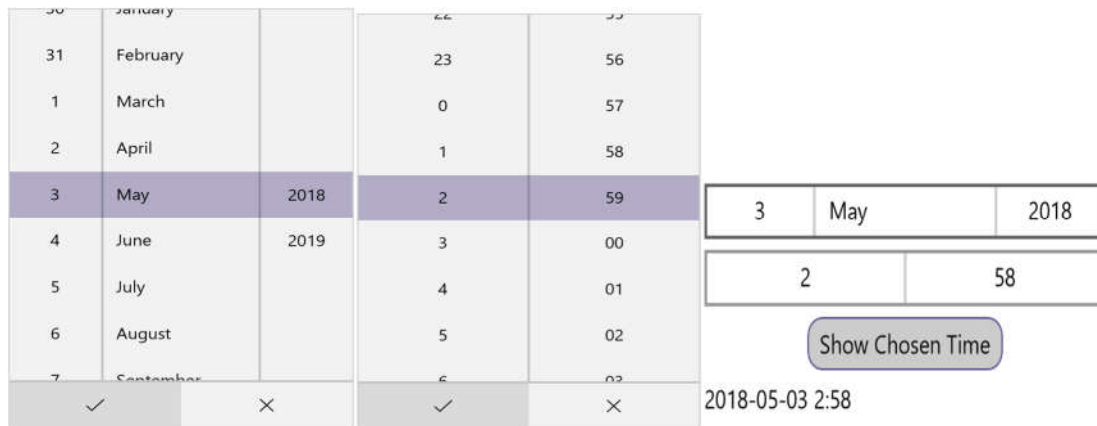


Figure 4.3.1 UWP DatePicker

Figure 4.3.2 UWP TimePicker

Figure 4.3.3 UWP time format

2. Android

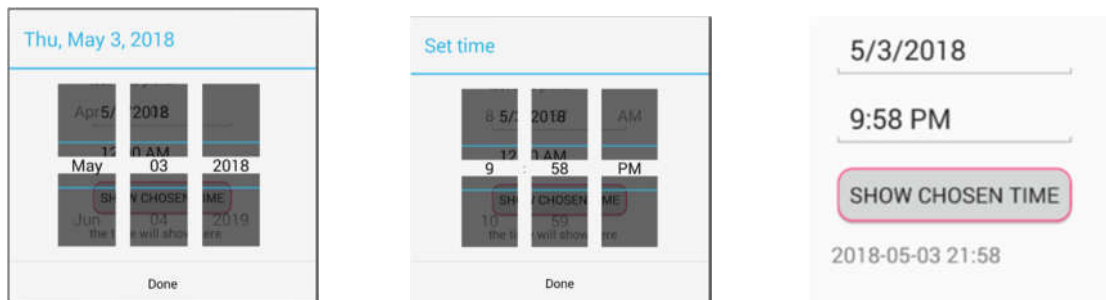


Figure 4.3.4 Android DatePicker

Figure 4.3.5 Android TimePicker

Figure 4.3.6 Android time format

3. iOS

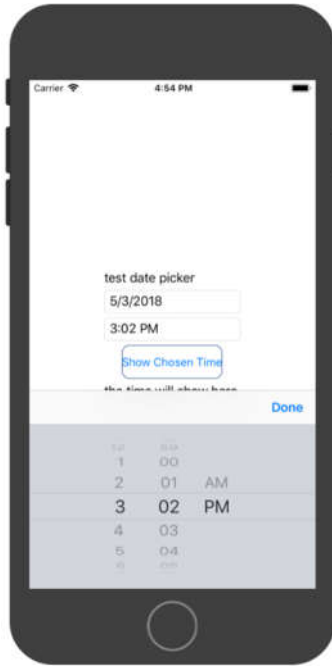


Figure 4.3.7 iOS TimePicker

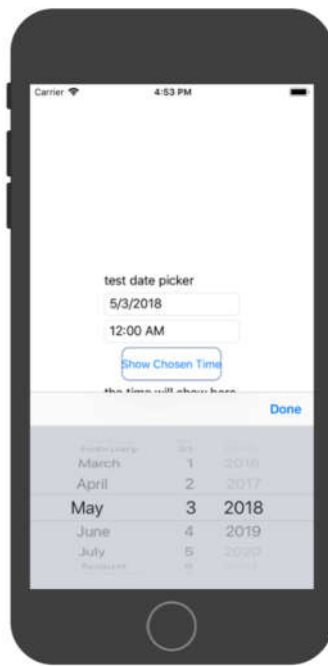


Figure 4.3.8 iOS DatePicker



Figure 4.3.9 iOS time format

In the screenshot, there are two kinds of time picker. The first one is Date Picker; the second one is Time Picker. Then there is a button which has the function to get the chosen time and value the label below. A label is below the button, which is used to show the value of the chosen time.

The C# syntax for DatePicker:

```
1. DatePicker datePicker = new DatePicker
2. {
3.     MinimumDate = new DateTime(2018, 1, 1),
4.     MaximumDate = new DateTime(2018, 12, 31),
5.     Date = new DateTime(2018, 6, 21)
6. };
```

Code 4.3.2 Syntax for DatePicker [20]

Once the picker has been selected, the date chosen item will be shown. The user could choose the year, month and day. The starting date and end date should be initialized in the beginning. The default date could be set in the instantiation procedure. After the date has been selected, the confirm button should be clicked to stop the date choosing.

The C# syntax for TimePicker

```
1. TimePicker timepicker = new TimePicker
2. {
3.     Time = DateTime.Now.TimeOfDay
4. }
```

Code 4.3.3 Syntax for Time Picker

In the constructor of the Time Picker, the default time could be set at the current time. What's more, when time picker has been selected, the time picker items will be seen, and it will be closed after the confirm button has been clicked.[20]

There are some differences between platforms. In UWP, the default format of the time picker is the 24-hour system. In Android and iOS are 12-hour system with AM/PM. So, to test the return format of the time, a button “Show Chosen Time” and the click event is set.

```

1. private void ShowTime_Clicked(object sender, EventArgs e)
2.     {
3.         time_Hour = tp.Time.Hours.ToString();
4.         time_Min = tp.Time.Minutes.ToString();
5.         date = dp.Date.ToString("yyyy-MM-dd");
6.         ChosenTime.Text = date + " " + time_Hour + ":" + time_Min + ":00";
7.     }

```

Code 4.3.4 Get value from TimePicker and DatePicker

According to the screenshot, the format of the date is “YYYY-MM-DD hh: mm”. The seconds part is omitted in the emulators, but it does not affect the data comparison in the database.

4.4 App Class

The Application class has three properties: *MainPage*, *Properties*, *Current*. [21]

Property	Description
MainPage	Used to set the initial page for the application.
Properties	A persistent dictionary is used to store simple values across lifecycle state changes.
Current	It contains a reference to the current application object.

Table 4.4.1 App class properties [21]

4.5 IDictionary interface

In this thesis, the IDictionary interface is used to store a collection of data. The syntax for its implementation is:

```

1. using System.Collections.Generic;
2. ...
3. Dictionary<TKey, TValue> instance = new Dictionary<TKey, TValue>();

```

Code 4.5.1 Syntax for IDictionary Interface

The value stored in the dictionary is in the form of Key-Value pairs. The TValue could be an integer, a string or a string array. The TKey could be an integer or string value.[22] Several methods are provided for this interface:

Method	Description
<code>public void Add (TKey key, TValue value);</code>	Add a key-value pair into the instance
<code>public bool ContainsKey (TKey key);</code>	Return a Boolean value whether the instance contains specific key

<code>public bool Remove (TKey key);</code>	Remove the key-value pair from the instance
---	---

Table 4.5.1 Methods in IDictionary Interface [22]

4.6 Navigation Methods

In Xamarin, two navigation methods are provided. One is called Modal page navigation, and the other one is called Modeless page navigation.

```
1. Task PushAsync(Page page)
2. Task PushModalAsync(Page page)
3.
4. Task<Page> PopAsync()
5. Task<Page> PopModalAsync()
```

Code 4.6.1 Page Navigation Method [23]

The Push methods are used to navigate to new pages, and meanwhile, the pop methods are used to go back to the previous page. The push and pop method should correspond. Otherwise, there will be an error message. The biggest difference between a modal page and the modeless page is that there is no self-contained back button in the modal page because the developer doesn't want the user to go back to the previous page without finishing the operation on the current page. In this thesis, the majority of the page is navigated by modal page. [23]

5 The Implementation of the Application “Event Manager”

5.1 Project Description

To discover the possibilities and the limitation of the Xamarin, some tasks have been made. The main task is to create an application “Event Manager” which is based on the application “Location Finder” [24][25]. The “Event Manager” is designed for two target groups which are the event organizers and participants. During the realization procedure, this application requires to integrate two separate parts of “Location Finder” into a whole and transplant the code to the latest version of Visual Studio 2017.

In the business logic part, this application could be divided into two parts. In the first part, event organizers could register an account for publishing events and enter the detail information of the events in the application. Besides, the type of the events could be added dynamically. In the second part, event organizers could search the events according to the range, time and event type conditions. The detail information of the events will be retrieved from the database. What's more, the structure of the MySQL database needs to be adapted for a data operation.

After the primary task has been realized, some subtasks are needed to be accomplished to discover the possibilities and limitations of Xamarin.Forms from other perspectives.

Firstly, the user preference setting should be stored in a local file.

Secondly, the utilization of Pop-Up menus should be investigated and applied to the “Event Manager”.

Thirdly, the methods for application publishing to different platforms will be investigated. A sample application will be published to Google Play store and Apple Store, and meanwhile, this

application will be published to another Win 10 machine and be launched without running Visual Studio.

5.2 Use Case Diagram

The functional requirements of the application according to the specification are abstracted into use cases. Some part of the use cases is shared with organizers and participants. Both could see the detail of the application and chose the language. In organizer parts, the majority of the functions is to operate on the events. For example, the organizer could publish the events and modify the detail of the events. In participant part, the majority of the functions is to search the events according to certain conditions. The most significant difference is the organizer is required to register an account and log in.

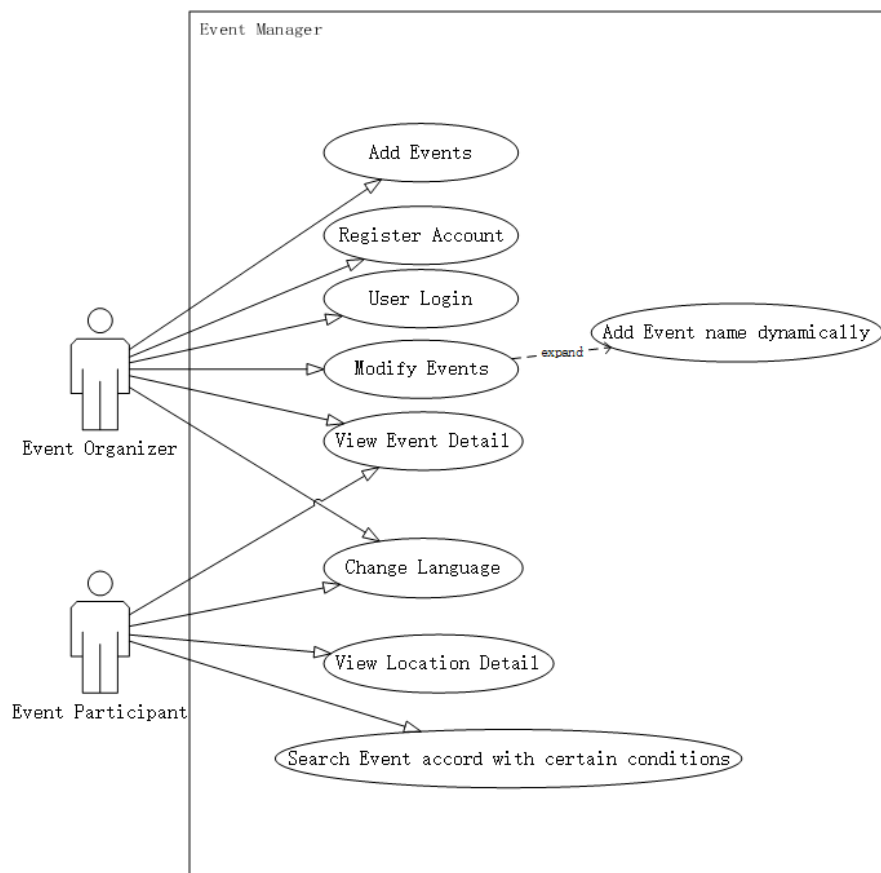


Figure 5.2.1 Use case diagram for Event Manager

5.3 Adapt and integrate previous students' result

In the latest version of Visual Studio 2017, the .NET Standard have replaced the PCL, because there are no PCL strategies when a blank app is created. As a result, it is impossible to run the old version of the application which uses PCL libraries without any version conflicts. To avoid this problem, a blank project should be created for old application installation. In this thesis, the application called “Location Finder”[24][25] which was created by Mr. Haocheng Liu and Mr. Yongsheng Huang was used and modified. Mr. Yongsheng Huang created owner part, and Mr. Haocheng Liu created the user part of the “Location Finder”[24][25]. The source code application “OwnerBay” was used to do the transplantation. It will be contained in the resource USB sticker.

Step 1. Create a new Project and generate a blank app

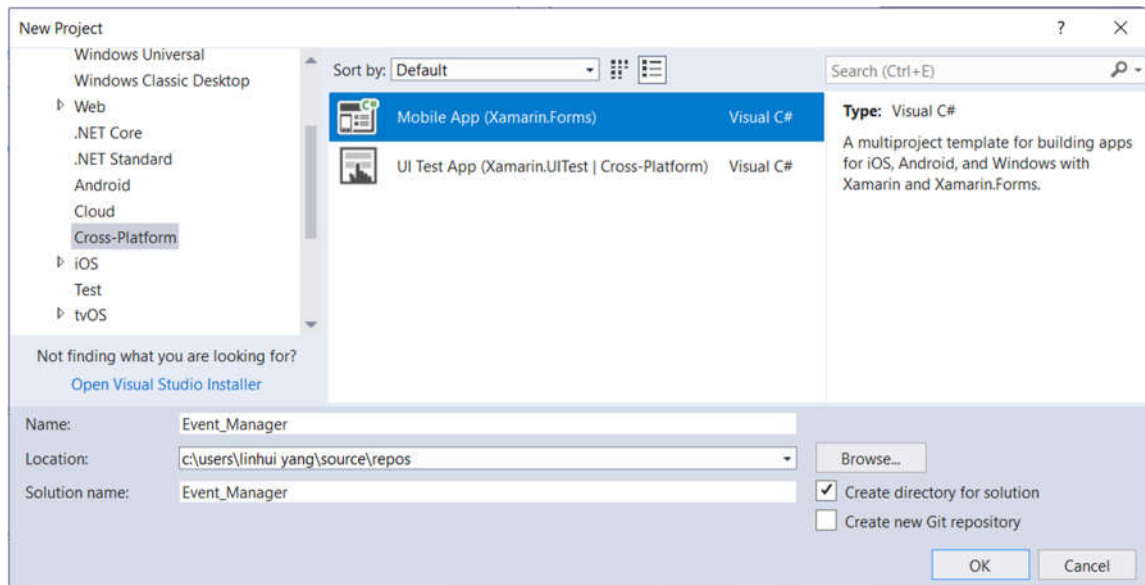


Figure 5.3.1 Create a new Project

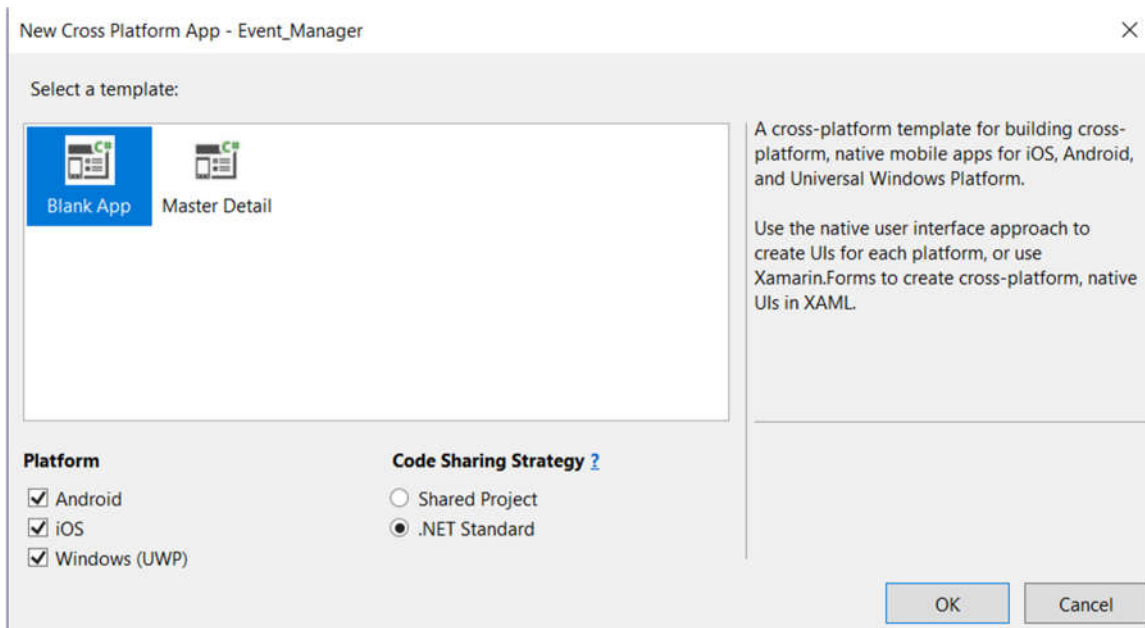


Figure 5.3.2 Generate a Blank App

In the figure 5.3.1, a new cross-platform project is created. After the creation, in the submenu, the item Blank App should be chosen. Then the .NET Standard should be selected as Code sharing method.

Step 2. Copy files from “OwnerBay” to a new project.

Firstly, go to source folder of “OwnerBay” and select following .cs files.

sktop > Chinesen 2017 Adaptiert auf 2018 > OwnerBay > OwnerBay > OwnerBay > OwnerBay

<input type="checkbox"/> Name	Date modified	Type	Size
bin	13/04/2018 15:54	File folder	
obj	13/04/2018 15:54	File folder	
Properties	13/04/2018 15:54	File folder	
App	18/12/2017 11:10	Windows Markup ...	1 KB
App.xaml.cs	18/12/2017 11:31	Visual C# Source F...	1 KB
<input checked="" type="checkbox"/> CommentPage.cs	18/12/2017 11:27	Visual C# Source F...	14 KB
<input checked="" type="checkbox"/> DataTypeTransfer.cs	18/12/2017 11:27	Visual C# Source F...	4 KB
<input checked="" type="checkbox"/> EventPage.cs	18/12/2017 11:27	Visual C# Source F...	42 KB
<input checked="" type="checkbox"/> EventRegistrationPage.cs	18/12/2017 11:27	Visual C# Source F...	6 KB
<input checked="" type="checkbox"/> HomePage.cs	18/12/2017 11:27	Visual C# Source F...	1 KB
<input checked="" type="checkbox"/> IDAL.cs	18/12/2017 11:27	Visual C# Source F...	8 KB
<input checked="" type="checkbox"/> IPicturePicker.cs	18/12/2017 11:27	Visual C# Source F...	1 KB
<input checked="" type="checkbox"/> LanguageChoosePage.cs	18/12/2017 11:27	Visual C# Source F...	3 KB
<input checked="" type="checkbox"/> LoginPage.cs	18/12/2017 11:27	Visual C# Source F...	6 KB
MainPage	18/12/2017 11:10	Windows Markup ...	1 KB
MainPage.xaml.cs	18/12/2017 11:10	Visual C# Source F...	1 KB
<input checked="" type="checkbox"/> MapPage.cs	18/12/2017 11:27	Visual C# Source F...	6 KB
<input checked="" type="checkbox"/> MealPage.cs	18/12/2017 11:27	Visual C# Source F...	55 KB
<input checked="" type="checkbox"/> Multilingual.cs	18/12/2017 11:27	Visual C# Source F...	16 KB
OwnerBay	18/12/2017 14:48	Visual C# Project fi...	6 KB
packages.config	18/12/2017 14:48	XML Configuration...	1 KB
<input checked="" type="checkbox"/> Page1.cs	18/12/2017 11:27	Visual C# Source F...	40 KB
<input checked="" type="checkbox"/> ProfilePage.cs	19/12/2017 08:04	Visual C# Source F...	48 KB
<input checked="" type="checkbox"/> RegisterPage.cs	15/04/2018 16:53	Visual C# Source F...	43 KB

Figure 5.3.3 Copy .cs file from “OwnerBay”

Secondly, paste these files into the new project. These files should be added to code sharing project.

Another part of “Location Finder”[24][25] should also be copied and pasted. However, in the specification, the restaurant part is not necessary. Moreover, the restaurant part was the main body of the user part of the “Location Finder”[24][25]. As a result, it is unnecessary to integrate any files from that app.

Step 3. Change the namespace

Because these .cs file are copied from another project, so the namespace should be changed to match the reference.

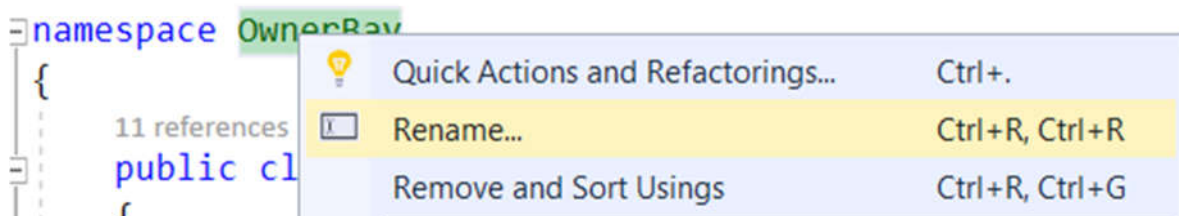


Figure 5.3.4 Rename Namespace 1

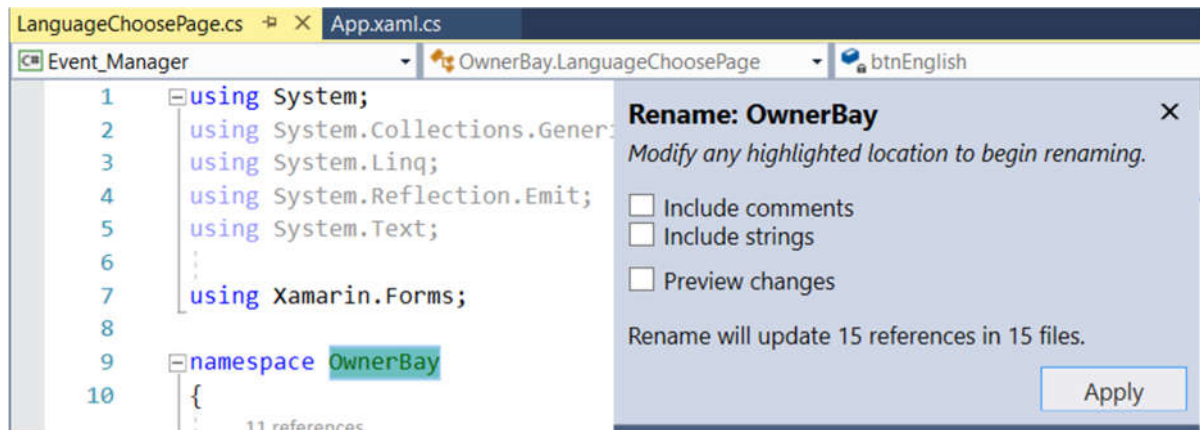


Figure 5.3.5 Rename Namespace 2

Open one of the .cs files which from “OwnerBay”, right-click the namespace. Then choose Rename. After that change the “OwnerBay” to “Event_Manager”.

Step 4. Copy implementation .cs files into each platform-specific project

In each platform-specific project (UWP, Droid, iOS), there are some implementation files which implement the interface defined in the code sharing project. However, there is some difference in each platform, so the procedures are different. Taking UWP project as an example, Firstly, copy file *DAL.cs* and *PicturePicker.cs* from *OwnerBay.UWP*, then paste these files into *Event_Manager.UWP*. This procedure can be done by selecting the UWP project in Visual Studio and press CTRL+V.

Secondly, change the namespace reference, which is the same with Step 3.

Thirdly, change the Dependency service in the *DAL* and *PicturePicker.cs* file. It should be corresponding to the namespace of code shared project. It can be seen in the Figure 5.3.6.1.

```
[assembly: Dependency(typeof(Event_Manager.UWP.PicturePicker))]  
  
namespace Event_Manager.UWP
```

Figure 5.3.6 Change Namespace and Dependency Service

The operation procedure in other two platforms is the same.

Step 5. Adding MySQL Data Plugins and references

In the application “OwnerBay”, several plugins are required for application launching. In the previous version of Visual Studio 2017, these plugins were managed in the Component. Besides, in the source folder of the project, all plugins and references were stored in the Packages folder which locates in the root folder of the project. However, in the latest version of Visual Studio 2017, the Component has been canceled, and there is no folder called “Package” in the root directory. According to the announcement given by the Microsoft, the libraries of Component has been integrated into NuGet package. So, the plugins and references should be reloaded manually into NuGet Packages.

Before installing plugins into platform-specific projects, the plugins in the Code Shared project should be installed, and the version should be unified.

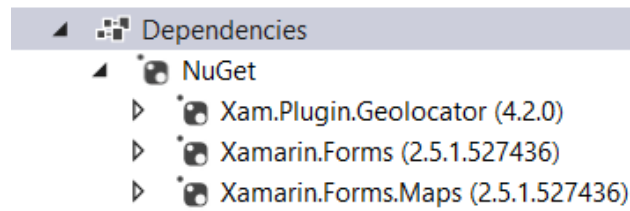


Figure 5.3.7 Plugins in Code Shared Project

These plugins could be obtained in the NuGet Package. It can be open by click the Project ->Manage NuGet Packages.

In this application, three plugins are missing in platform-specific projects: MySQL Data Plugin, Geocoder, and Xamarin.Forms.Maps. However, the MySQL data plugin required by platforms are different. Therefore, the detail installation steps will be introduced.

In UWP

The MySQL.Data plugin could be installed in the NuGet package. Go to the NuGet package for UWP, then enter the MySQL.Data in the search bar. Install the plugin into the project.

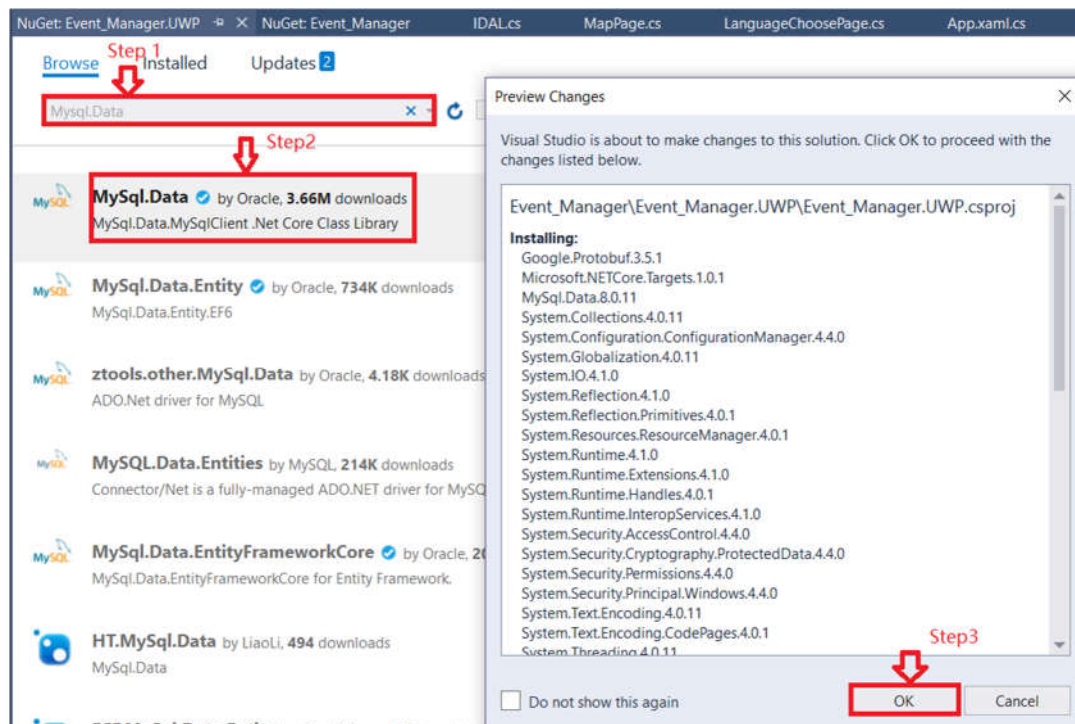


Figure 5.3.8 Install MySQL.Data into UWP Project

In Android and iOS

In the previous vision, the MySQL plugin for iOS and Android was from Component Store. However, in the current version of Visual Studio, the Component Store is not available. In the package of “OwnerBay”, the MySQL plugin which was installed from the Component Store could be found in the “Component” folder.

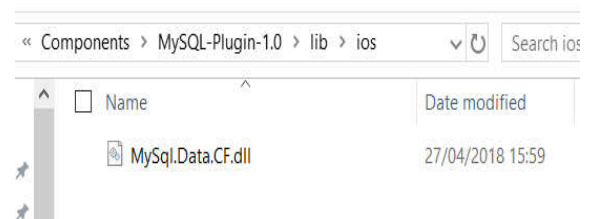


Figure 5.3.9 MySQL Plugin for iOS and Android

Firstly, adding this component into the reference of the Android and iOS projects manually.

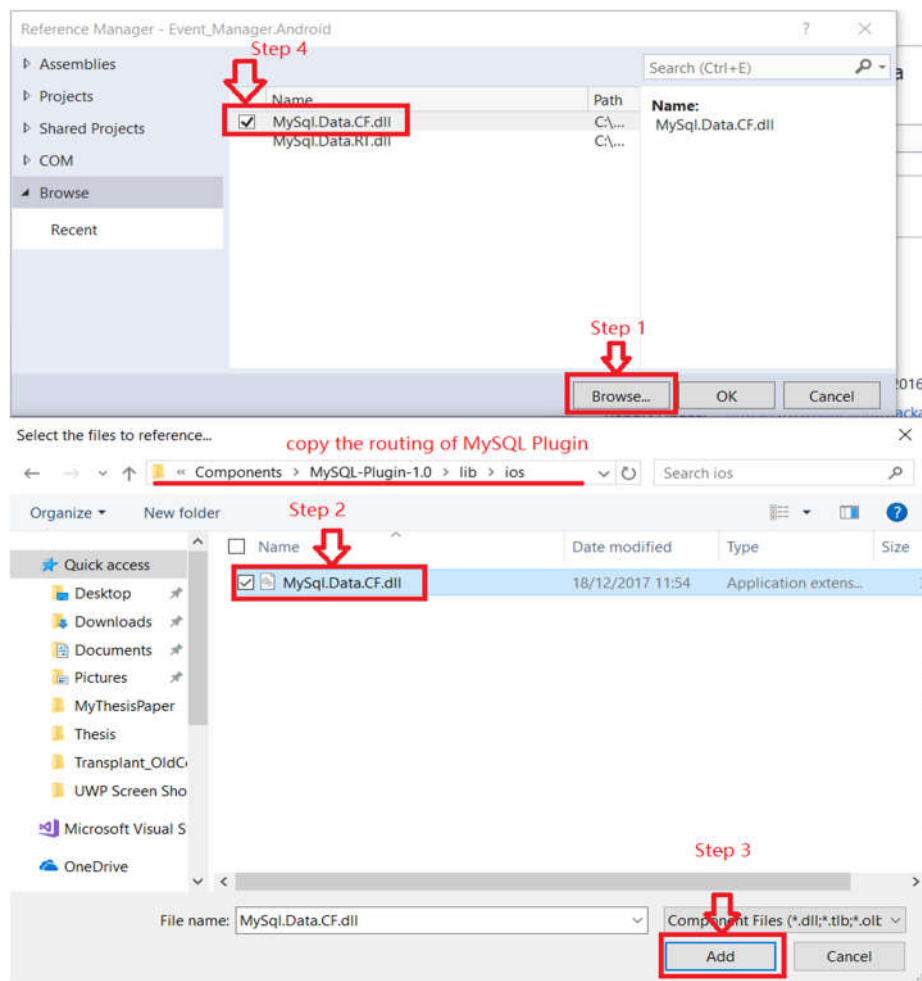


Figure 5.3.10 Adding MySQL reference manually

Besides, to add MySQL plugin reference, the System.Data reference should also be added.

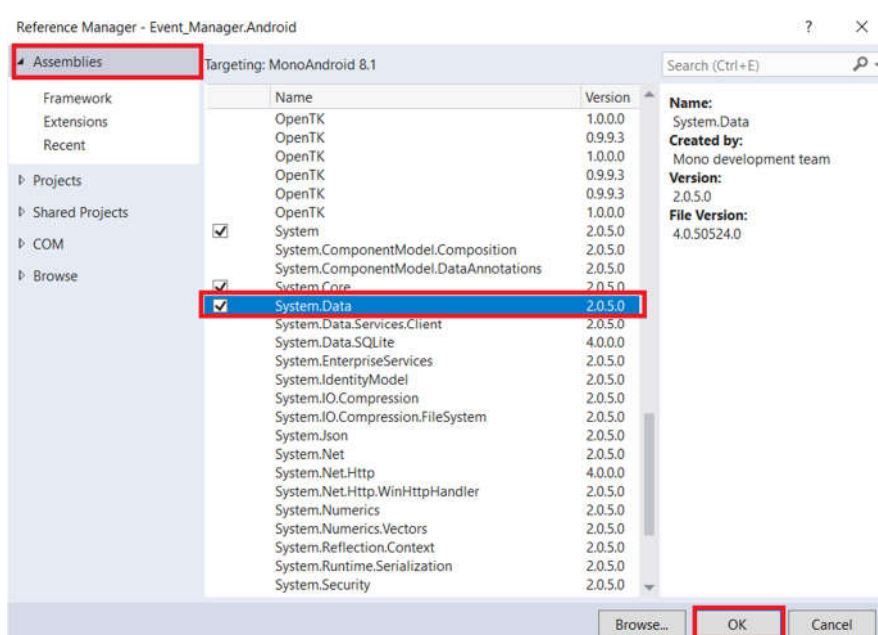


Figure 5.3.11 Adding System.Data reference

Finally, the MySQL plugins have been installed successfully. The procedure for adding a reference in Android and iOS is the same.

Step 6 Adding Maps, Geocode, and Geolocator

These plugins should be added to all projects. What’s more, the version of these plugins should be corresponding with the version of Xamarin.Forms. Moreover, some other plugins should also be installed to assist the functions of these plugins. In the following table, the necessary plugins are listed.

Project	Plugin name and version
Android Project	Xamarin.Forms.Maps v2.5.1.527436 Xamarin.Forms v2.5.1.527436 Xam.Plugin.Geolocator v4.2.0 System.ObjectModel v4.3.0 Plugin.Permission v2.2.1 Plugin.CurrentActivity v2.1.0.2 Xamarin.GooglePlayServices v27.0.0 Xamarin.GooglePlayServices.Maps v60.1142.1
iOS Project	Xamarin.Forms.Maps v2.5.1.527436 Xamarin.Forms v2.5.1.527436 Xam.Plugin.Geolocator v4.2.0 Plugin.Permission v2.2.1
UWP Project	Microsoft.NETCore.UniversalWindowsPlatform v6.0.8 Xamarin.Forms.Maps v2.5.1.527436 Xamarin.Forms v2.5.1.527436 Xam.Plugin.Geolocator v4.2.0 Microsoft.BingMaps.V8.TypeScript v2.0.3

Table 5.3.1 Plugins needed for each project

In the old version of the application, the Geocoder is used in projects. However, in the new version, this plugin is incompatible with Xamarin.iOS and monoAndroid. What’s more, if the Geolocator is used in the project, the Geocoder will conflict with the syntax of Geolocator. As a result, this plugin will not be installed.

Step 7 Initialize Maps in projects

After the map components have been installed, they should be called and initialized in the projects.

In UWP

Firstly, adding following code into the *MainPage.xaml.cs* file.[27]

```
1. Xamarin.FormsMaps.Init("INSERT_AUTHENTICATION_TOKEN_HERE")
```

Code 5.3.1 Initialize Maps in UWP [27]

Secondly. Insert the map authentication token into the initialization code. The code is:

3OKUqTtrqaXZ7QnAxUwJ~1KtxBPWVyAH4Vud5CIy5yA~AsdwTmY8jx6jPkdzeC90ejLfCEt5jT0mg2ubSWMeKA3SXLVjWQVFxhpiyT60zdC

The Bing Map Authentication Key can be obtained from[26]. However, an account for Bing Map should be created. Then in the “MyKey” under the “MyAccount” to create a key.

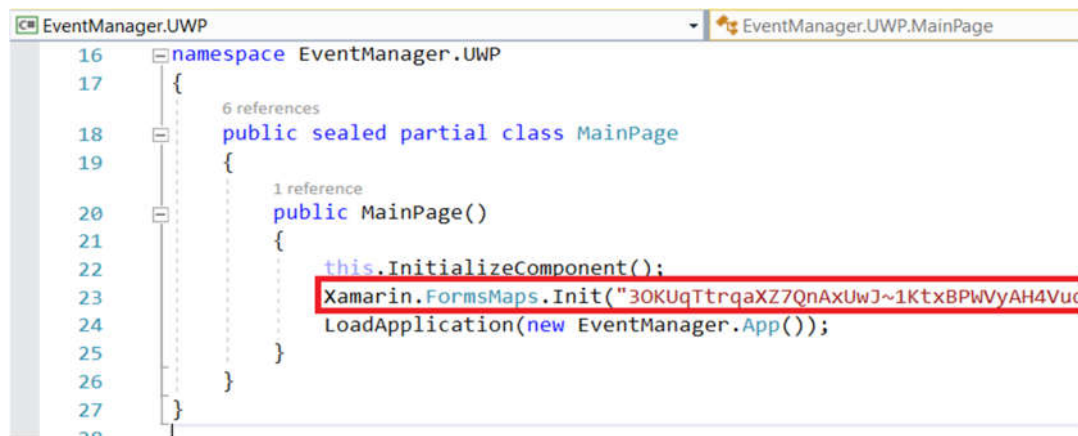


Figure 5.3.12 Initialize Maps in UWP

In iOS

Firstly, adding following code into the *AppDelegate.cs* file. It should be added into the *FinishedLaunching()* method. [27]

```
1. global::Xamarin.FormsMaps.Init();
```

Code 5.3.2 Initialize Maps in iOS [27]

```

public override bool FinishedLaunching(UIApplication app, NSDictionary options)
{
    global::Xamarin.Forms.Forms.Init();
    global::Xamarin.FormsMaps.Init();
    LoadApplication(new App());

    return base.FinishedLaunching(app, options);
}

```

Figure 5.3.13 Initialize Maps in iOS[27]

Secondly, adding following codes into the *Info.plist* file. Open this file with XML editor, and then add these codes close to the end tag of plist. [27]

```
1. <key>NSLocationAlwaysUsageDescription</key>
2. <string>Need location for geolocator plugin.</string>
3. <key>NSLocationAlwaysAndWhenInUseUsageDescription</key>
4. <string>Can we use your location</string>
5. <key>NSLocationWhenInUseUsageDescription</key>
6. <string>We are using your location</string>
7. <key>NSPhotoLibraryUsageDescription</key>
```

Code 5.3.3 Initialize Maps in Info.plist [27]

In Android

Firstly, adding following code into the *MainActivity.cs* file. [27]

```
1. global::Xamarin.Forms.Forms.Init(this, bundle);
```

Code 5.3.4 Initialize Maps in Android [27]

```
3 references
public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
    0 references
    protected override void OnCreate(Bundle bundle)
    {
        TabLayoutResource = Resource.Layout.Tabbar;
        ToolbarResource = Resource.Layout.Toolbar;

        base.OnCreate(bundle);

        global::Xamarin.Forms.Forms.Init(this, bundle);
        LoadApplication(new App());
    }
}
```

Figure 5.3.14 Initialize Maps in MainActivity.cs

Secondly, configure the Permission of the Emulator in the Android properties -> Manifest. The permissions which are related to the locating should be allowed. Besides, the INTERNET option should also be enabled. [27]

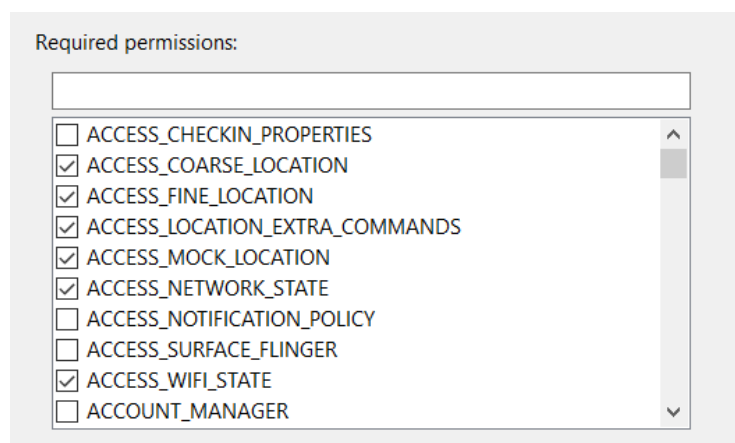


Figure 5.3.15 Configure the permissions for Emulator

5.4 Results of “Event Manager”

In this section, the screenshots of the application “Event Manager” will be shown. What’s more, the connection between pages will be introduced. However, some detail explanation of functions and database connection methods will be explained in the following “Realization details” part. The primary function of this part is to show the reader how this application looks like and how to operate it.

This app developed by Xamarin.Forms provide three different platforms (Android, iOS, and UWP). So, for the whole project, there are three platform-specific projects. Whereas, it has been mentioned in the previous section that these projects share most of the user interface code. As a result, the appearance of this application in different platforms is similar. So, only the screenshots in UWP are used to explain.

The application “Event Manager” set an entrance for organizers and participants, after the language preference, they could choose their identity.

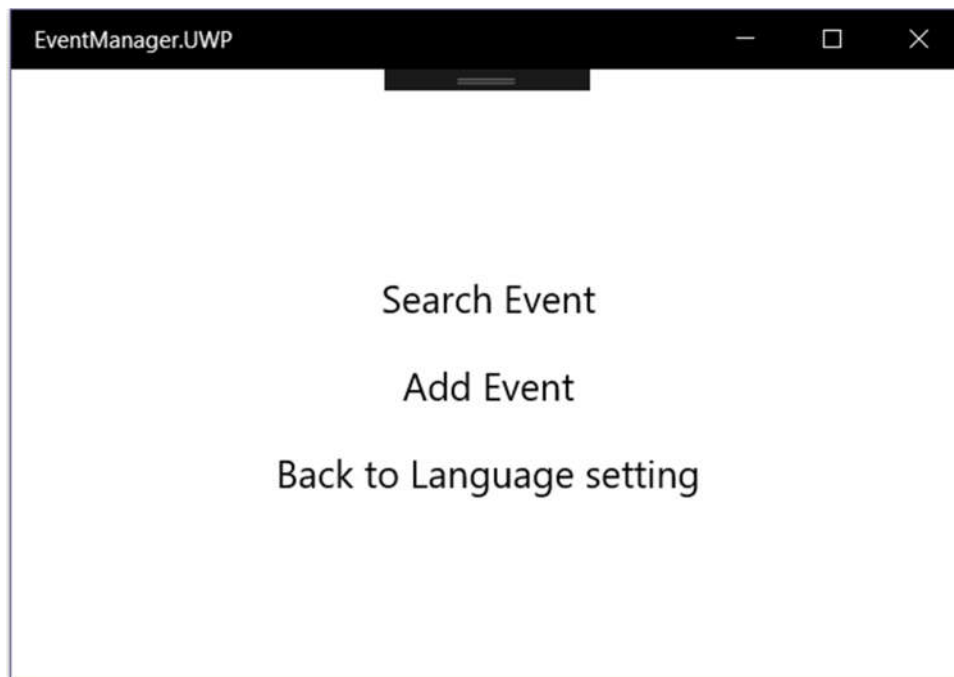


Figure 5.4.1 Choose an Identity in Event Manager

There are three buttons on this page. First, two buttons share one button click event. The formal parameter will be passed in and checked. If the “Search Event” button has been clicked, the organizer page will be pushed. Else, if the button “Add Event” has been clicked, the participant page will be pushed. The last button has the function to go back to the language choose page. So, this application will be introduced in two parts.

I. Organizer part

In the event organizer part, a login page will be shown.

The figure consists of two side-by-side screenshots of the EventManager.UWP application. The left screenshot shows the login page with fields for Email (containing 'tt') and Password (containing two dots), and buttons for Register, Login, and LoadInfo. The right screenshot shows the registration page with fields for Name, Email, Password, Town, Zip Code, Street, Street Number, Telephone, Telephone2, Homepage, ChartLink, and Description.

Figure 5.4.2 Organizer Login Page

Figure 5.4.3 Register an account for Organizer

To access the event publishing page, an account is needed. If the organizer doesn't have an account, he/she should click the “Register” button to create an account.

In the register page, there is some information should be filled in. Some basic personal information like name, email and password should be entered. The “Email” will be used as the account for accessing the event publishing page. In the following entries, some location information should be entered. After the information has been entered and checked, the save button should be clicked. Then a confirm box will jump out. After that, the register page will be closed and go back to the previous page.

After the user enters the account and password, the event home page will be pushed.

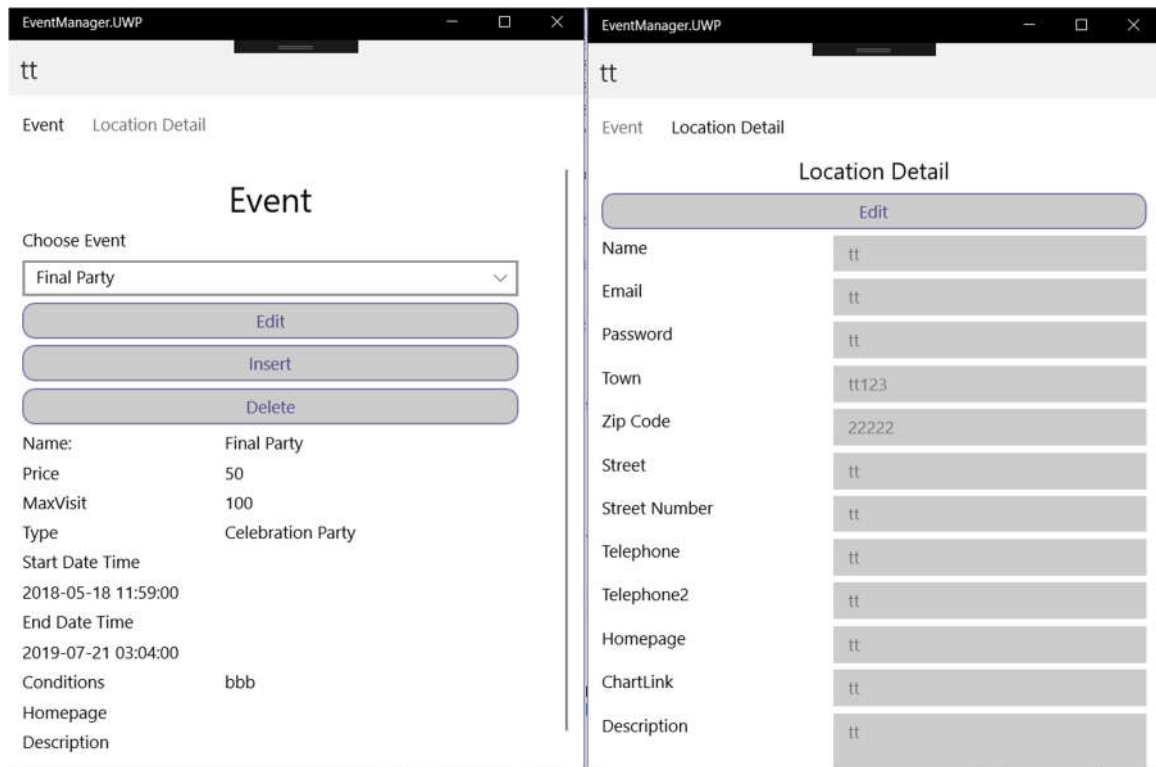


Figure 5.4.4 Organizer Event Page

Figure 5.4.5 Organizer Location Page

The organizer event page and location page are the collection of pages in a tabbed page. In the title bar, the name of the organizer will be shown as title. These pages can be shifted by clicking the tags which located in the tag bar. The left one is the event page. The organizer could choose the event in the event picker. Once the event picker item has been changed, the information in the following table will also be changed. Some modifications to the event can be operated by clicking the edit button. The table below threes buttons will be changed.

Figure 5.4.6 Edit the event

The detail information of the event will be replaced by entry and pickers. What’s more, the organizer could add a new type of the event dynamically. It will be explained comprehensively in the following parts. After the modifications have been completed, click the confirm to save the value into the database. Similarly, the insert function provides the same user interface to enter the detail of an event. One difference is that there is no value in each entry. Besides, the delete function will delete the event in the database.

II. Participant part

In the event participants part, login is not necessary. However, the “Event Manager” is developed based on the previous application “Location Finder”. To main the uniformity, the login page and register page have been kept. However, on the login page, a button called “later” can skip these operations and directly go to the event search page.

Figure 5.4.7 Event search page

In the event choose page, several filters are provided in the form of the picker. Firstly, the current position of the participant will be shown. Then three conditions can be set. In the range condition, there are six conditions: “2KM”, “5KM”, “10KM”, “20KM”, “50KM” and “Show All”. After the range condition, the date which participant want to attend the event should be set. If the participant does not want to set, the default value has already set as the current date and time. Finally, the event type should be chosen. Once the conditions have been decided and the search button has been pressed, the event page will be pushed into the screen.

Figure 5.4.8 Event detail page

Figure 5.4.9 Event Location Detail

To make the user interface simple and clear, all events are put into the picker. Beneath the event picker, a “back” button is set if there are no desired events under certain conditions. If the participant has interests in this event, he/she could click the “Location Detail” button to get more information. In the Figure 5.4.10, the contact information and location details are presented. Then the participant could contact the organizer and attend the event.

5.5 Realization details

5.5.1 Language preference saving

According to the specification, two operation language options should be provided. The user could choose English or Deutsch as operation language. The language preference will be asked at the beginning time when the user opens the application; then the preference will be saved into the local file. The user will not be asked to choose language again when the next time he/she opens the application. However, the users can change the language in the submenu.

Screenshot of language choose page

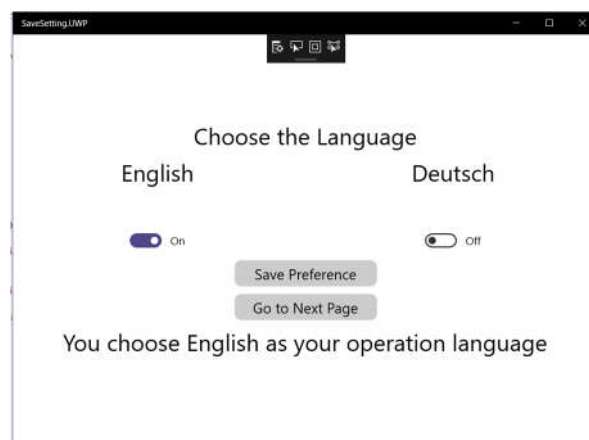


Figure 5.5.1 UWP Language Setting Page

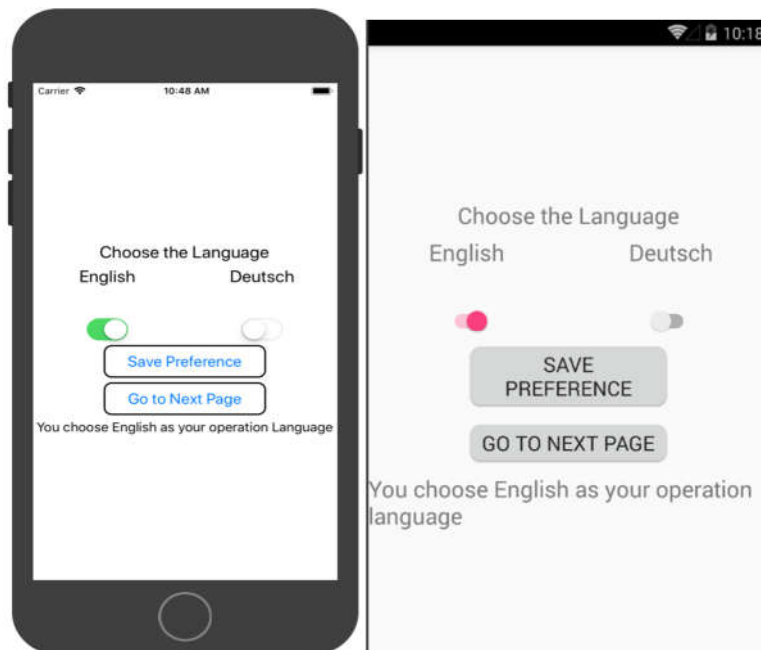


Figure 5.5.2 iOS Language Setting Page Figure 5.5.3 Android Language Setting Page

In the screenshots, two switches are used to represent the chosen language. After language choosing, click the button *Save Preference*, the language preference will be saved into the local file. Then click the button *Go to Next Page*, after that a new page will be pushed.

In the realization part of the button clicked event, a simple validation will be checked. If no switch is toggled, an alert window will pop up. Similarly, if both switches are toggled, an alert window will pop up. Each switch corresponds to a language token. In the code, an integer number is set to mark the language. If English switch has been toggled, the language token will be valued 0. Similarly, if “Deutsch” switch is toggled, then the language token will be valued 1. After the validation, the language setting token will be saved into a local file. Once the language preference has been saved, the application will skip the step to choose a language.

The user could open this language setting page in the submenu. The switch will maintain the situation which it has been set before. In the code part, the status of the switch is defined by the language token.

```
1. //retrive the properties
2. IDictionary<string, object> properites = Application.Current.Properties;
3. if (Application.Current.Properties.ContainsKey("Language"))
4. {
5.     String current_token = Application.Current.Properties["Language"
6.     ].ToString();
7.     if (current_token == "1")
8.     {
9.         De_Switch.IsToggled = true;
10.    }
11.    if (current_token == "0")
12.    {
13.        Eng_Switch.IsToggled = true;
14.    }
15.    };
```

Code 5.5.1 Retrieve switch status

In the second line of the code 5.5.2.1, the value stored in the local file will be retrieved into a Dictionary.

5.5.2 User account preference saving

Once the user successfully enters the account and the password, these values will be saved into a local file. These values will be auto-filled into the account and password entry by a button click action when the next time user want to log in. It has been applied to the Organizer Login Page. The realization procedure will be shown in the following.

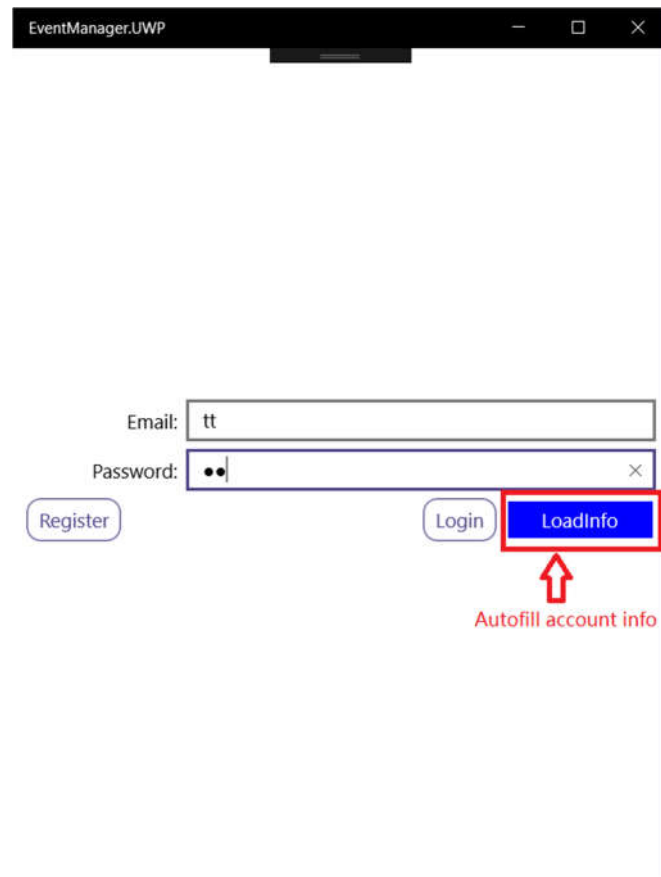


Figure 5.5.4 Autofill account information

Step 1. In the following code, the entry for Email and Password are initialized. Besides, the instance of the local file should be initialized at the beginning.

```
1. ...
2. //set the entry for User name and password
3. Entry etEmail, etPassword;
4.
5. public OwnerRegisterPage()
6. {
7.     //initialize the local file, instantiate the properties instance
8.     IDictionary<string, object> properites = Application.Current.
    Properties;
9.     //instantiate the entry
10.    etEmail = new Entry { };
11.    etPassword = new Entry { };
12.    ...
13. }
```

Code 5.5.2 Entry instantiation of Username and Password

Step 2. In the button click event, store the parameter value into the local file. The method *SavePropertiesAsync()* could be used at anywhere of the page. As a result, it can guarantee the permanent storage.

```
1. private async void BtnLogin_Clicked(object sender, EventArgs e)
2. {
3.     ...
4.     //save property
5.     Application.Current.Properties["UserName"] = etEmail.Text;
6.     Application.Current.Properties["Password"] = etPassword.Text;
7.     await Application.Current.SavePropertiesAsync();
8.     ...
9. }
```

Code 5.5.3 Store the parameter into the local file

Step 3. Retrieve the account information from the local file. It can be done by a button click event. A “LoadInfo” button has been set in this page.

```
1. private void LoadInfo_Clicked(object sender, EventArgs e)
2. {
3.     etEmail.Text = Application.Current.Properties["UserName"].ToString();
4.     etPassword.Text = Application.Current.Properties["Password"].ToString();
5. }
```

Code 5.5.4 Load User Account Information

5.5.3 Location information

In the “Event Manager”, the position of the user is an important attribute to filter the events. Since the plugin Geolocator has been installed, so the current position of the user could be retrieved by the GPS sensor. In this thesis, this method has been implied in several pages.

In event organizer part, the method is called in the page *OwnerRegisterPage*. In the button click event *BtnCurentPositon_ClickedAsync()*, the double-type Latitude and Longitude are valued by current position. These values are obtained from the GPS locator. In the UWP project, an alert window will jump out and ask for the permission to get the current location. In the page *OwnerMapsPage*, the value of latitude and longitude are obtained from the Map which user could enter the street name and get the value form the map. In the button click event *OnBtnFind_ClickedAsync()*, the method *GetPositionsForAddressAsync(string streetname)* will search the name of a street in the map, then get the exact value of latitude and longitude. After that, the value will be stored in an instance, in which the latitude and longitude value could be extracted.


```

1. public async void getCurrentPosition()
2. {
3.     var locator = CrossGeolocator.Current;
4.     locator.DesiredAccuracy = 50;
5.
6.     var location = await locator.GetPositionAsync();
7.     latitudeValue = location.Latitude;
8.     LongitudeValue = location.Longitude;
9.     currentPositionValue.Text = "Latitude is:" + latitudeValue.ToString(
10. ) + " "
11.         + "Longitude is:" + LongitudeValue.ToS
    tring();
12. }

```

Code 5.5.5 Get Location Information

In the participant part, the location information will get in the page *UserEventSearchPage*. The user's current position will be obtained. Then the value will be used to choose the range condition. In the button click event *getCurrentPosition()*, an instance of plugin Geolocator will be instantiated. Then the method *GetPositionAsync()* will be called by the instance. It will take some time to get the position value because it is an asynchronous method. Then the data will be passed to the label parameter. The Latitude and Longitude are Double type value.

During the development phase, there was one configuration problem which resulted in the data format error. In the database, the location detail of the position is “XX.xxxxx”. However, the data get from the function is “XX,xxxxx”. As a result, it is impossible to exert the range filter.

This application is developed in the English operation language. However, the computer is activated in Germany region. The solution is to configure the radix point in the system.

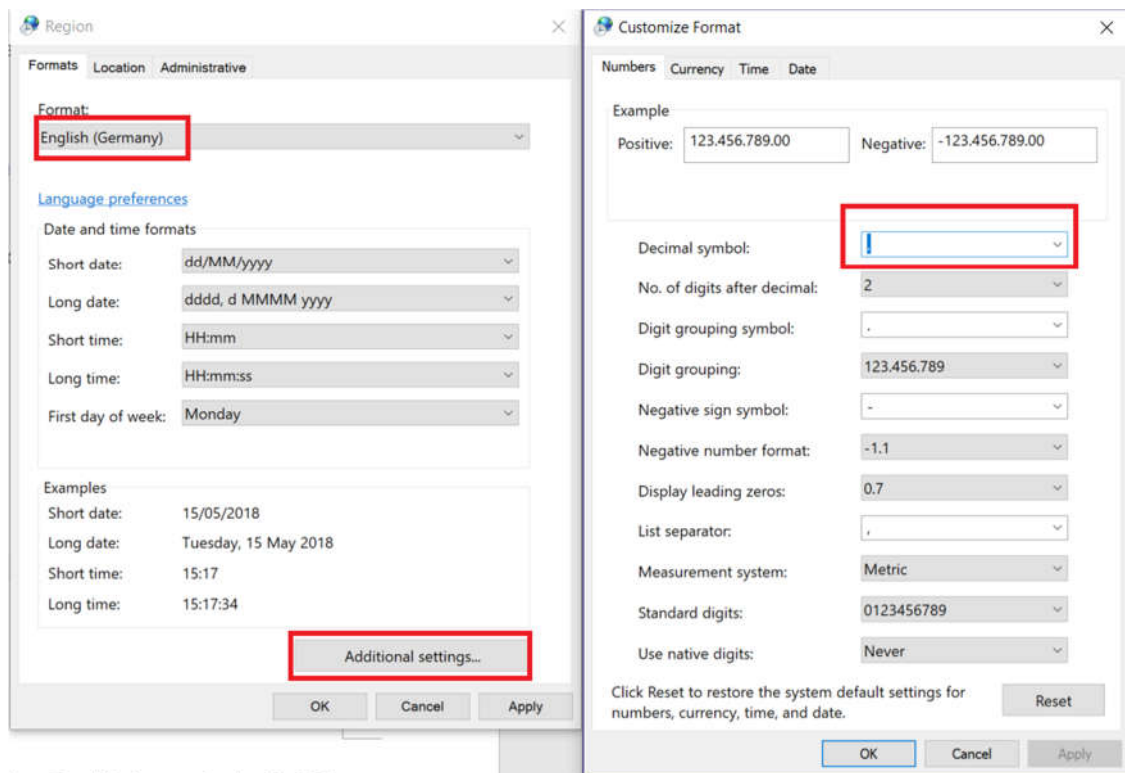


Figure 5.5.5 Change Decimal symbol

Firstly, go the setting->region. Then click the Additional setting. After that change the decimal symbol from “,” to “.”.

5.5.4 Dynamic adding items into Picker

According to the specification, if there is no event type which matches the type of the event, Event Organizers can create a new type of event. Adding an event type entry page by a button click action is not user-friendly. For the mobile device user, it will interrupt their mind. So, a dynamic event picker is created.

In the event picker, once the Organizer chooses the picker item “Enter Other Event”, a new page will be generated. The Organizer could enter the name of the event into the entry.

Screenshots:



Figure 5.5.6 UWP Dynamic Event Picker

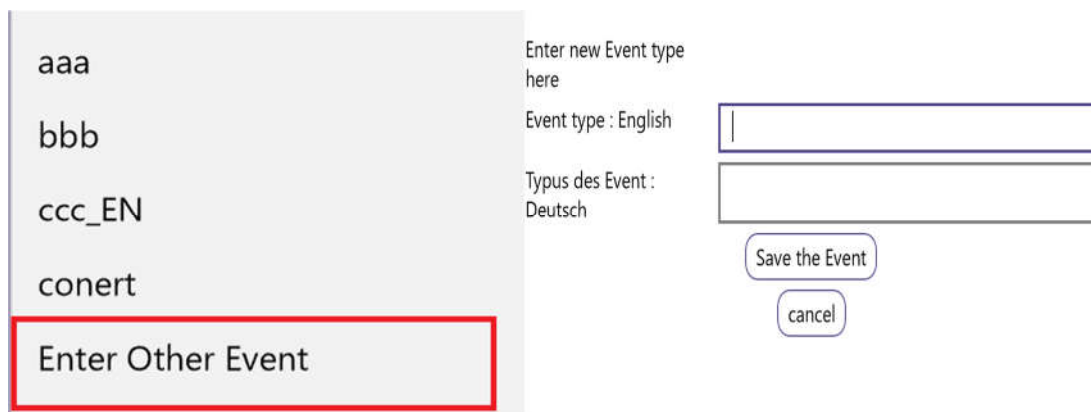


Figure 5.5.7 UWP Dynamic Event Picker Items

Figure 5.5.8 New Event type entry page

As has been mentioned in the specification, once the user chooses the operation language, the language choose will not be asked in the following operation. So, in the event entry page, the Organizer could choose to fill one of the event type entry. There exists the situation that the Organizer only know one of the languages, so the other kind of language could be left blank. Furthermore, a new database table and database connection class should be created.




Table Name:

Schema:
w14515_locals

Collation:
utf8 - default collation

Engine:
InnoDB

Comments:




Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 number	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 EventType_EN	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 EventType_DE	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Figure 5.5.9 Detail of Event Type table

	number	EventType_EN	EventType_DE
	1	aaa	aaa
	2	bbb	bbb
	3	ccc EN	ccc DE
	4	conert	konzert
	5	Football match	Fussball
	NULL	NULL	NULL

Figure 5.5.10 Event Type Table Sample

There are three columns. The first column is the primary key column “number”, and the number of the primary key is in a sequence which starts from 1. The following two columns store the information of the event types. One in English and the other in Deutsch. Considering the situation that the organizer will leave another part blank so that event type can be set as null.

5.5.5 Database layer

A database is needed for data storage. In this thesis, all event details are stored in MySQL database. The name and the functions of the data table are presented in the following table.

Table Name	Description
Event	The detail information of events is stored in this table Foreign keys: <i>Type</i> , <i>LocLink</i>
EventType	The type name of the events is stored in this table; the main key is used to connect to Event table. Primary key: <i>number</i>
Location	The location detail information is stored in this table. Mainly used by event organizers, and the main key is used to connect to <i>Event</i> table. Primary key: <i>Keynb</i>

Table 5.5.1 Database Table Description

The database layer should be separate from the business logic and view layer. As a result, the database connection and some data operations are realized in the platform-specific classes. Then these classes can be connected to the code sharing project by using dependency services.

Step 1. Create an interface in the code sharing project, declare the name of the functions.

Step 2. Add dependency connection in each implementation class. It should be added before the namespace.

Android	1. [assembly: Dependency(typeof(EventManager.Droid.DB))]
iOS	1. [assembly: Dependency(typeof(EventManager.iOS.DB))]
UWP	1. [assembly: Dependency(typeof(EventManager.UWP.DB))]

Table 5.5.2 Sample of adding assembly in the implementation class

Step 3. Use MySQL plugin in the database implementation class.

```
1. using MySql.Data.MySqlClient;
```

Code 5.5.6 using MySQL Plugin namespace

Step 4. Instantiate the interface in code sharing project and call the data operation methods when needed. After that, the data layer could be connected to the view layer.

5.5.6 Data operation and filtering

The data operation can also be divided into two parts, organizer part and participant part.

I. Organizer part

The organizer part is based on the application “Location Finder”[24][25], and on this basis, some functions were altered, and new functions were added. What’s more, in the following part detail explanations will be given to help readers understand how organizer get and change event details.

1. Account registration

In the *OwnerRegisterPage*, after the registration information has been entered, the user could press the register button to create an account. In the *DAL.cs* which is the data operation implementation class, the data will be stored in the database. The following code which locates in the *DAL.cs* will show how to insert the data into the database. [24]

```

1. public void RegisterMainInfo(string name, string email, string password, string
   town, int townnb, string street, string streetnb)
2. {
3.     //provide the base class for an encoding provider
4.     EncodingProvider ppp;
5.     ppp = CodePagesEncodingProvider.Instance;
6.     Encoding.RegisterProvider(ppp);
7.
8.     //SQL database connection statement, then connect the database
9.     string connsqlstring = "Server='www.joergbayerlein.de';" +
10.        "Port=3306;" +
11.        "User Id=w14515_bay2;" +
12.        "Password=bay35984;" +
13.        "Database=w14515_loicals;" +
14.        "charset=utf8;" +
15.        "SslMode=none";
16.     MySqlConnection conn = new MySqlConnection(connsqlstring);
17.     conn.Open();
18.     //SQL insert statement
19.     string sqlstring = "insert into Location (Keynb, Name, Email, password,
   Town,Townnb, Street,Streetnb)" +
20.        " values (@Keynb,@name,@email,@password,@town,@townnb,@street,@streetnb)";
21.     MySqlCommand cmd = new MySqlCommand(sqlstring, conn);
22.     cmd.Parameters.AddWithValue("@keynb", Convert.ToString(GetMaxLocationKeyI
   d() + 1));
23.     cmd.Parameters.AddWithValue("@name", name);
24.     cmd.Parameters.AddWithValue("@email", email);
25.     cmd.Parameters.AddWithValue("@password", password);
26.     cmd.Parameters.AddWithValue("@town", town);
27.     cmd.Parameters.AddWithValue("@townnb", Convert.ToString(townnb));
28.     cmd.Parameters.AddWithValue("@street", street);
29.     cmd.Parameters.AddWithValue("@streetnb", streetnb);
30.
31.     cmd.ExecuteNonQuery();
32.     conn.Close();
33. }
```

Code 5.5.7 Register Main information [24]

Firstly, from line 3 to 6 a base class is provided for an encoding provider. Then connect to the database. After that execute the query statement. Finally, the data will be stored in the database. [24]

2. Get event/location detail

After the organizer login, the organizer account will be passed to the *EventPage* which serves as a token to get the corresponding events. Then the function *GetAllEventSet(accountId)* will be called and return event detail value. These value will be stored in a *Dictionary<int, string[]>*. The following codes will show how it implements in the *DAL.cs*.

```

1. public Dictionary<int, string[]> GetAllEventSet(int accountId)
2. {
3.     //database connection
4.     ....
5.     //initialize a dictionary for event detail storage
6.     Dictionary<int, string[]> result = new Dictionary<int, string[]>();
7.     string sqlstring = "select * from Event where locLink = @keynb";
8.     MySqlCommand cmd = conn.CreateCommand();
9.     cmd.Parameters.AddWithValue("@keynb", Convert.ToString(accountId));
10.    cmd.CommandText = sqlstring;
11.    MySqlDataReader rd = cmd.ExecuteReader();
12.    int count = 1;
13.    while (rd.Read())
14.    {
15.        string[] record = new string[13];
16.        record[0] = rd.GetInt32("Keynb").ToString();
17.        record[1] = rd.GetString("Name");
18.        record[2] = rd.GetString("description");
19.        record[3] = rd.GetInt32("Type").ToString();
20.        record[4] = rd.GetDateTime("Startdatetime").ToString("yyyy-MM-
dd hh:mm:ss");
21.        record[5] = rd.GetDateTime("Enddatetime").ToString("yyyy-MM-
dd hh:mm:ss");
22.        record[7] = rd.GetInt16("Flagonline").ToString();
23.        record[8] = rd.GetInt16("MaxVisit").ToString();
24.        record[9] = rd.GetFloat("Price").ToString();
25.        record[10] = rd.GetInt16("locLink").ToString();
26.        record[11] = rd.GetString("conditions").ToString();
27.        record[12] = rd.GetString("Homepage").ToString();
28.        result.Add(count, record);
29.        count++;
30.    }
31.    conn.Close();
32.    return result;
33. }
```

Code 5.5.8 Get Event Detail Information [24]

Similarly, this procedure is used to get the location information. In the *OwnerprofilePage*, the account will be obtained from *OwnerLoginPage*. Then methods *GetLocation*(accountId)* will be called to obtain the location detail. [24]

3. Modify event details

The organizer could modify the existed event details by clicking the edit button and save the modification by clicking the save button. In the save button, method *UpdateEvent*(accountId, parameter)* will be called to update the data. An example to update the event name will be shown in the following code.

```

1. public void UpdateEventName(int keynb, string name)
2. {
3.     //database connection
4.     ...
5.     string sqlstring = "update Event set Name = @name where Keynb = @keynb";
6.     MySqlCommand cmd = new MySqlCommand(sqlstring, conn);
7.     cmd.Parameters.AddWithValue("@keynb", keynb.ToString());
8.     cmd.Parameters.AddWithValue("@name", name);
9.     cmd.ExecuteNonQuery();
10.    conn.Close();
11. }

```

Code 5.5.9 Update Event Detail [24]

The *keynb* is the account of the organizer, and the parameter *name* is the new name of the event. After the database connection and SQL statement execution, the name of the event will be replaced.

“*GetLocation*()*” represents the methods starting with *GetLocation*. For example, “*GetlocationEmail(int keynb)*”, “*GetLocationPassword(int keynb)*” and so on.

“*UpdataEvent*()*” represents the methods starting with “*Update*”. For example, “*UpdateEventName(int keynb, string name)*”, “*UpdateEventDescription(int keynb, string str)*” and so on.

II. Participant part

In the participant part, the main task is to create a searching page which contains three filters.

1. Time filter

The time will be obtained from the data and time picker. The format is “YYYY-MM-DD hh:mm”. In the SQL searching sentence, the chosen time should be constrained in the domain of the start and end time of the events.

```

1. select * form Event where Startdatetime < "UserChosenTime" and "UserChosenTime" < Enddatetime

```

Code 5.5.10 SQL Time filter searching sentence

2. Type filter

In the *UserEventSearchPage*, the user could select the type of the event. Then the type name will be searched in the “*EventType*” table. The “number” in the table will be returned which can relate the “*EventType*” table with the “*Event*” table.

```

1. select number from EventType where EventType_EN="userChosenType" or EventType_De="userChosenType"

```

Code 5.5.11 SQL Type filter searching sentence

3. Range filter

Some range options are provided in the range picker in the *UserEventSearchPage*. The parameter in the SQL sentence is decided by the range picker.

```

1. if (userChosenRange == "Show all")
2. {
3.     RangeCondition = "";
4. }
5. else if (userChosenRange == "2km")
6. {
7.     double minLongitude = userLongitude - 0.02;
8.     double maxLongitude = userLongitude + 0.02;
9.     double minLatitude = userLatitude - 0.01796945;
10.    double maxLatitude = userLatitude + 0.01796945;
11.    RangeCondition = " Longitude<" + maxLongitude.ToString()
12.                    + " and Longitude>" + minLongitude.ToString()
13.                    + " and Latitude<" + maxLatitude.ToString()
14.                    + " and Latitude>" + minLatitude.ToString();
15. }
16. else if (userChosenRange == "5km")
17. {
18.     double minLongitude = userLongitude - 0.05;
19.     double maxLongitude = userLongitude + 0.05;
20.     double minLatitude = userLatitude - 0.04492363;
21.     double maxLatitude = userLatitude + 0.04492363;
22.     RangeCondition = " Longitude<" + maxLongitude.ToString()
23.                     + " and Longitude>" + minLongitude.ToString()
24.                     + " and Latitude<" + maxLatitude.ToString()
25.                     + " and Latitude>" + minLatitude.ToString();
26. }
27. ...
28. //return the locLink in the Location table
29. string[] locLink= db.SearchLocation(RangeCondition);

```

Code 5.5.12 Range filter code fragment [25]

The range filter was altered based on the result of the previous student Haocheng Liu. [25] In the filters there are five conditions: 2KM, 5KM, 10KM, 20KM, and 50KM, here only two are represented. If the range is 2 km, the longitude variation is 0.02[25], and the latitude variation is about 0.01796945 [25]. In the following condition, the range variations are increased proportionally directly.

After the range filtering, the parameter *RangeCondition* will be passed to database implementation class. In the *DB.cs* file, there is a function called *SearchLocaiton(string RangeCondition)* . After the filtering, the “Keynb” will be returned.

```

1. if (RangeCondition == "")
2. {
3.     sqlstring = "select Keynb from Location";
4. }
5. else
6. {
7.     sqlstring = "select Keynb from Location where" + RangeCondit
8.     ion;

```

Code 5.5.13 SQL Range Filter Searching Sentence

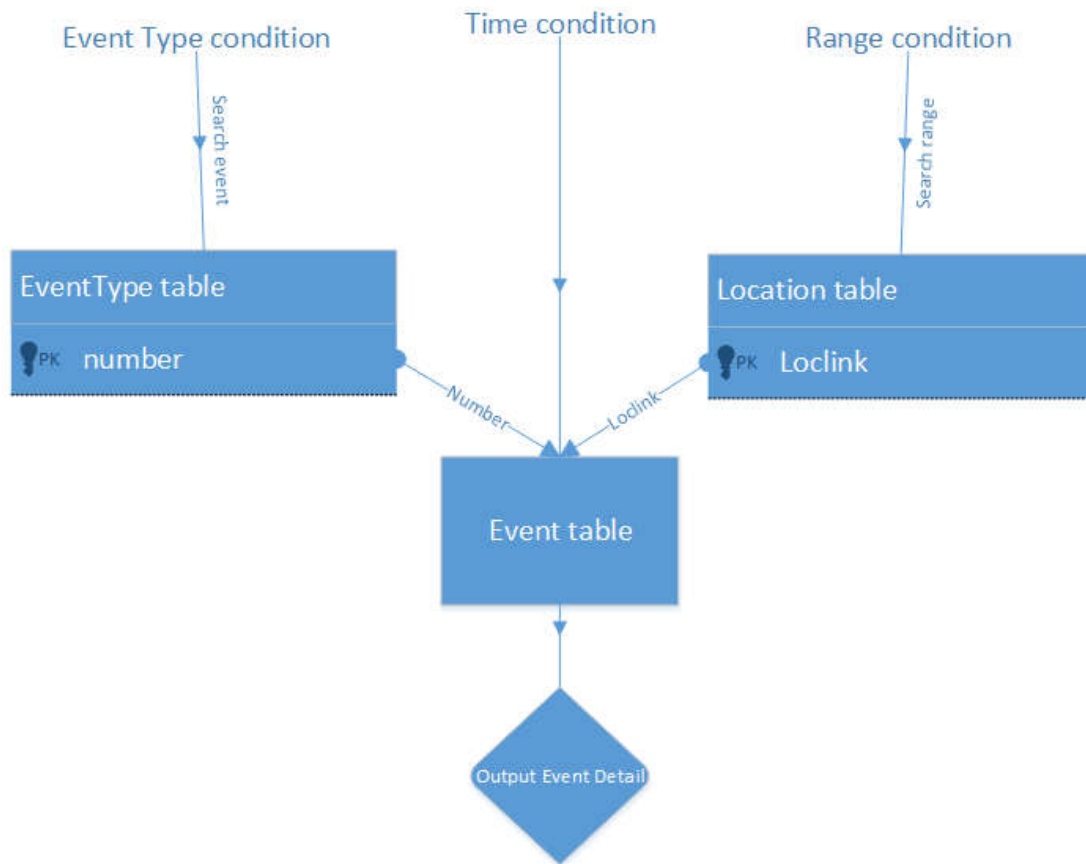
Get Event Detail in *UserEventDetailPage*

Figure 5.5.11 Event Filter Realizing Logic

Three tables should be queried because three conditions locate in different tables. The searching logic is depicted in the figure 5.5.6.1. The type condition is searched in the *EventType* table; then the number will be returned as the event type searching token. Similarly, *Loclink* will be return as the range searching token in the table *Location*. Finally, the desired event detail will be extracted from the *Event* table.

5.6 Subtask1: Pop-Up Menus

In the mobile application, a popup menu is very useful. Because the screens of the mobile devices basically are not so large, so some superfluous function buttons should be hide when they are not called. The Popup Menus could solve that space-using problem. It makes the user interface tidy and clean and meanwhile it provides equal functionalities like the comprehensive toolbar in other devices like Windows.

5.6.1 Using “DisplayActionSheet”

In Xamarin.Forms, there is a built-in method called “DisplayActionSheet”. It is an alert window which user could click the button in this window. After the button click, a popup window will show.

In C# code, when the “DisplayActionSheet” method is called, some string parameter will be passed to the Alert window. [28]

```
1. string popupMenu = await DisplayActionSheet("PopUp menus","Cancel",null, "item1","item2","item3" );
```

Code 5.6.1 Initialize a Popup Menu by DisplayActionSheet

The first parameter is the title of the alert window. The second and third parameter is the formatted function button in the alert window. The second is cancel button; the third is the destroy button. If there is no need for destroy function, it can be left null. Then the followings are the items in the alert windows. [28]

As Xamarin.Forms self-contains method, it provides the native looking in all three platforms.

Screenshots for three platforms:

1. Android



Figure 5.6.1 Android DisplayActionSheet 1

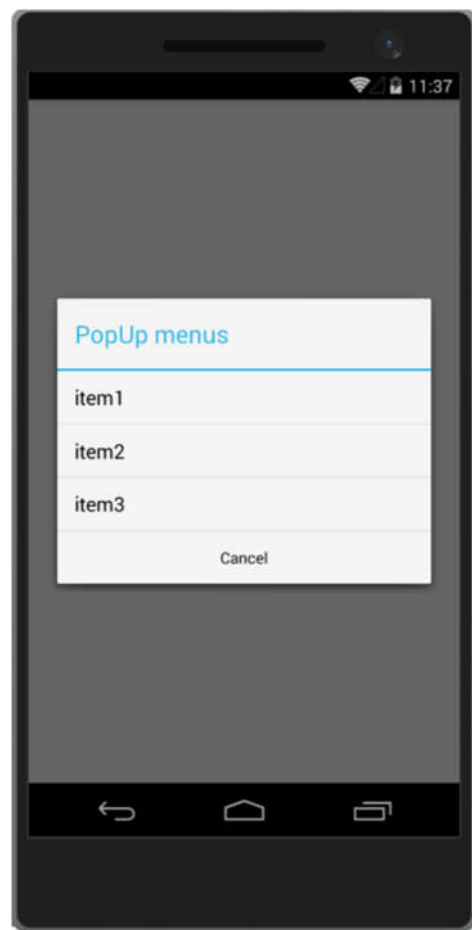


Figure 5.6.2 Android DisplayActionSheet 2

2. UWP

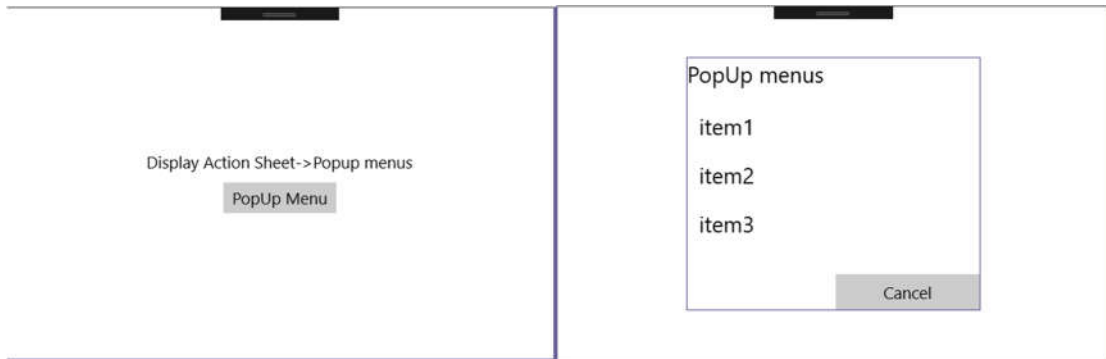


Figure 5.6.3 UWP DisplayActionSheet 1

Figure 5.6.4 UWP DisplayActionSheet 2

3. iOS



Figure 5.6.5 iOS DisplayActionSheet 1

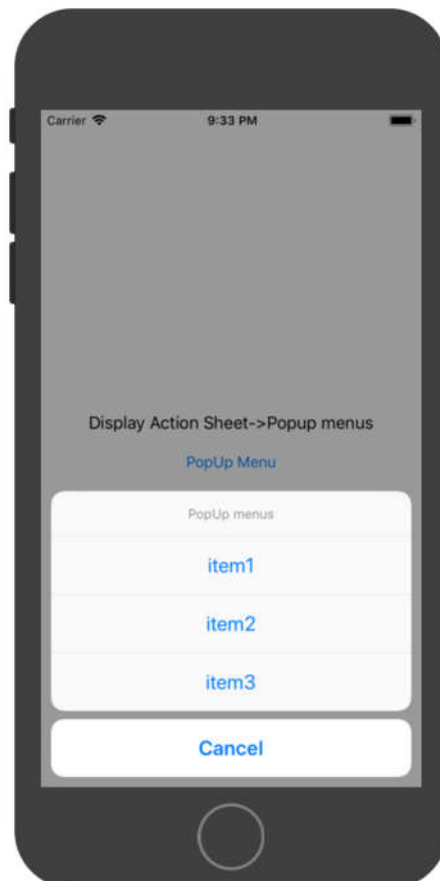


Figure 5.6.6 iOS DisplayActionSheet 2

The self-contained method provides a formatted pop-up window which has the title, two function buttons, and item lists. And the position of the pop-up window is fixed according to the platforms. In UWP and Android, the pop-up window is shown in the center of the screen. However, in the iOS, the pop-up window is shown in the bottom of the screen, which is the same with the system design of iOS.

5.6.2 Rg.Plugins.Popup (1.0.4)

There is another method for Pop-Up menus. A plugin can be used to create Pop-Up menus on all three platforms with more functions and fewer limitations. The plugin is called “Rg.Plugins.Popup(1.0.4)”.

It is a cross-platform plugin for Xamarin.Forms which enable the users to open new pages as pop-up pages. iOS, Android, and UWP are supported. Moreover, meanwhile, the plugin provides some methods for simple and flexible animation for showing pop-up pages. [29]

It is a free plugin, and it can be retrieved in NuGet package. In the following steps, the procedure for installing and utilizing will be described.

Step 1. Installation

In the Visual Studio navigation bar, click the Project, then go to the Manage NuGet Packages, enter the name of the plugin in the search bar. After that install the plugin into each project.

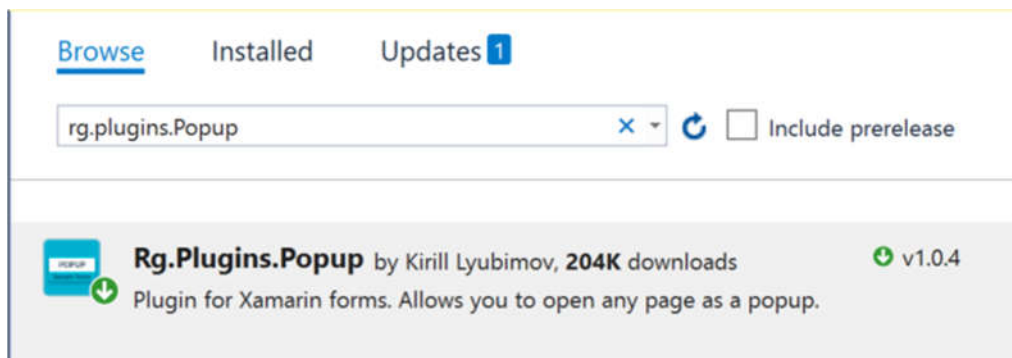


Figure 5.6.7 Install Rg.Plugins.Popup

Step 2. Using plugin’s namespace

Adding the namespace of the plugin into the .cs file which pop-up method is needed.

```
1. using Rg.Plugins.Popup.Services;
```

Code 5.6.2 Using Plugin’s Namespace

Step 3. Create a new Pop-Up page

A new Pop-Up page should be created. It can be customized by the developer. The developer could design the layout or the structure of the Pop-Up window. Before the page detail design,

```
1. using Rg.Plugins.Popup.Services;
2. ...
3. public class PopUpPage : Rg.Plugins.Popup.Pages.PopupPage
4. {
5. ...
6. }
```

Code 5.6.3 Create new Pop-up Page

The namespace of the plugin should be added at the top of .cs file. Then a reference to “Rg.Plugin.Popup.PopupPage” class should be added.

Step 4. Design Pop-Up Page

In the Pop-Up pages, the developer could design the layout and the structure of the Pop-Up window, and meanwhile, the animation could be added when the Pop-Up page is about to show.

Screenshots for three platforms:

1. Android



Figure 5.6.8 Android PopUp1

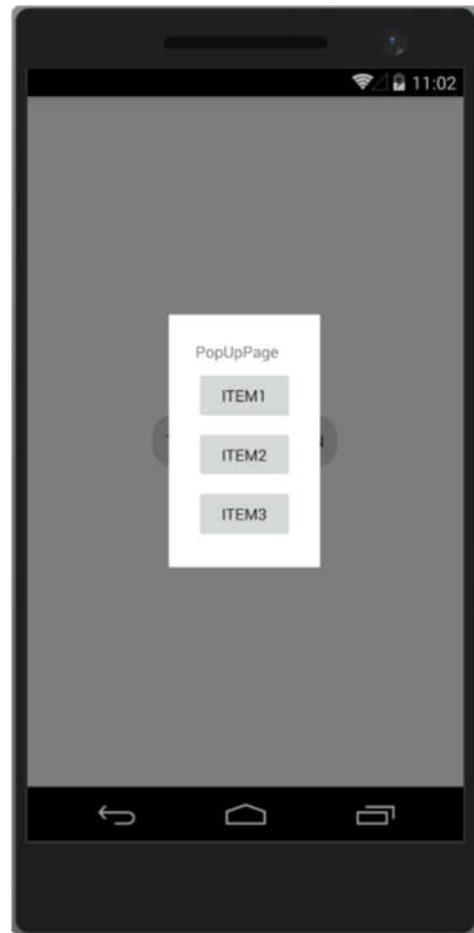


Figure 5.6.9 Android PopUp2

2. UWP



Figure 5.6.10 UWP PopUp1



Figure 5.6.11 UWP PopUp2

3. iOS



Figure 5.6.12 iOS PopUp1

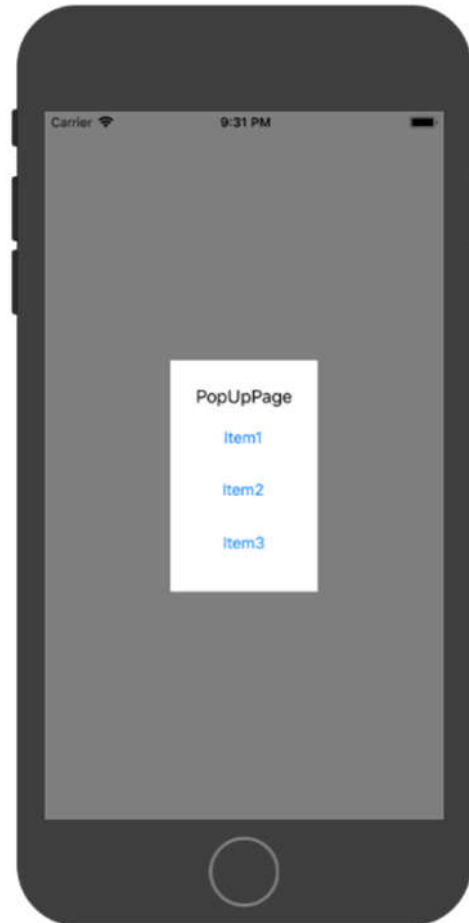


Figure 5.6.13 iOS PopUp2

By using the plugin, the pop-up window will show in the center of the screen. The developers have more options when designing the pop-up window.

The left picture is the Pop-Up trigger button, click the button, and a popup page will show. In the code part, that button name “PopUp” and exists in the *test_PopUp.cs* file.

The right picture shows the popup window. In the popup window, there are three item buttons. Click the “item1”, the popup window will close, and a label on the previous page will be shown. Clicked the “item2”, the popup window will be closed, and a new page will be pushed. Clicked the “item3”, the popup window will close and the language setting page for the “Event Manager” will be pushed.

In UWP, the user could click the grey background to cancel the Pop-Up window. But in the Android, the user should press the self-contained back button to cancel the Pop-Up window.

5.7 Subtask2: Publishing Application

5.7.1 Publishing Via Google Play

To investigate the method for publishing application via Google Play, a simple application “ReuterIdent” will be published to the Google Play store. The detail publishing procedure will be explained. The following steps were executed following the instruction is given by the Microsoft. For more detail information, it can be retrieved at the website [30].

A Google developer account is needed for application publishing. Since the developer account has been registered, the following publishing work could be executed.

Step 1. Specification the Application Icon [30]

An Android Application without an icon is not allowed to be published in some application markets. Go to the Android Properties -> Android Manifest, select “@drawable/icon” in the Application icon. [30]

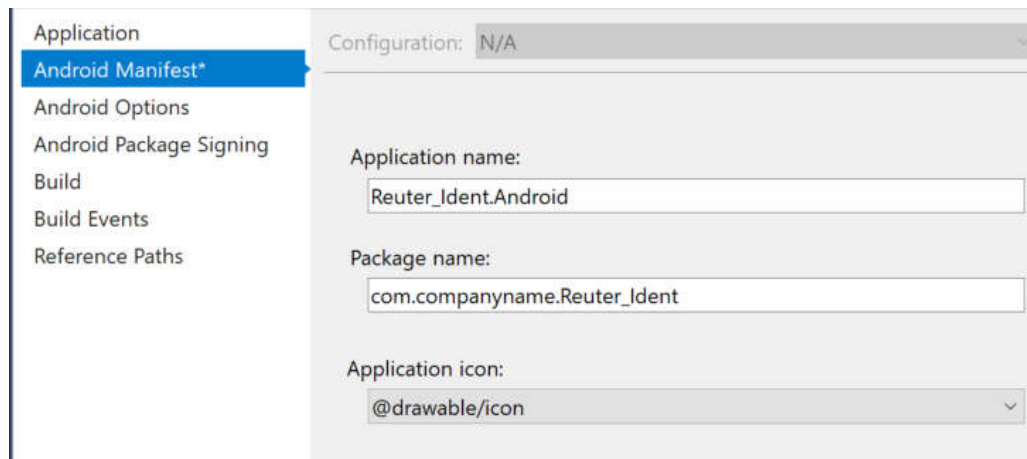


Figure 5.7.1 Android Application icon configuration

Step 2. Setting application version

In the Android Manifest, set the version of application you want to publish. The “Version Number” represents the version of the application.[30] It should be integer and will be used internally by Android and application. Basically, it starts with 1 and is incremented with each build. [30]

The “Version Name” is used specifically for users. It could be String or Number. It helps users to identify the build which has been installed in their devices. [30]



Figure 5.7.2 Set the Version Number

Step 3. Shrink the APK [30]

The size of the APK could be compressed by removing unnecessary managed code and unused Java bytecode. That can be done by using Xamarin.Android linker and ProGuard tool. [30]

In the Android Properties-> Android Options, choose “Sdk Assemblies Only” in the Linking.



Figure 5.7.3 Shrink the APK[30]

Step 4. Application Protection [30]

When the application has been published, it should not be debugged by the user. To Protect the application from the hacker, adding following codes into in the Android Project *AssemblyInfo.cs* file. [30]

```
//Protect the Application
#if DEBUG
[assembly: Application(Debuggable = true)]
#else
[assembly: Application(Debuggable=false)]
#endif
```

Figure 5.7.4 Protect the Application from unapproved debugging

Step 5. Setting Package Properties [30]

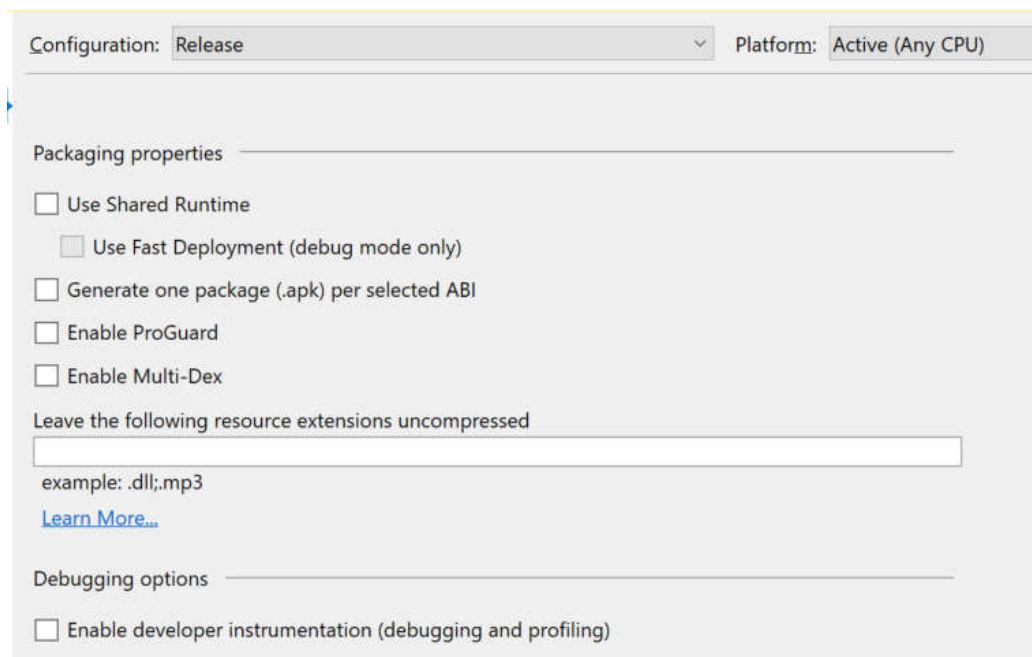


Figure 5.7.5 Set Package Property for archiving

Because it is a relatively small application, no needs to apply these properties. In Android Properties- > Options, Choose Release Configuration, clear all checkboxes. [30]

Step 6. Archiving the Package

In the Visual Studio toolbar, select “release”, then build the Android Project. After the successful build, archiving the Project.

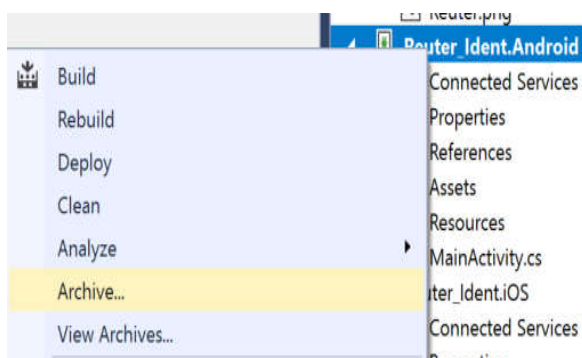


Figure 5.7.6 Archive the Android application

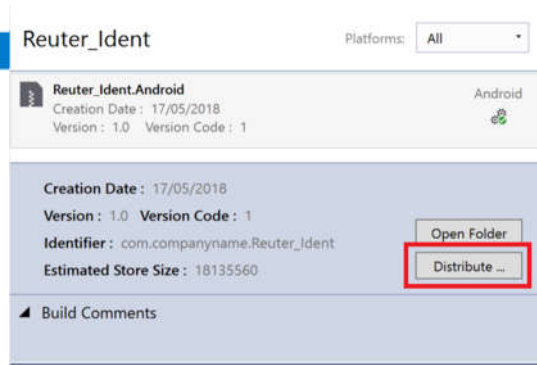


Figure 5.7.7 Archived result

The archived APK file will be stored in the directory: “C:\Users\YourUserName\AppData\Local\Xamarin\Mono for Android\Archives”. The developer could also upload the APK file manually to the Google Play Console to publish it. [30]

Step 7. Distribute the application

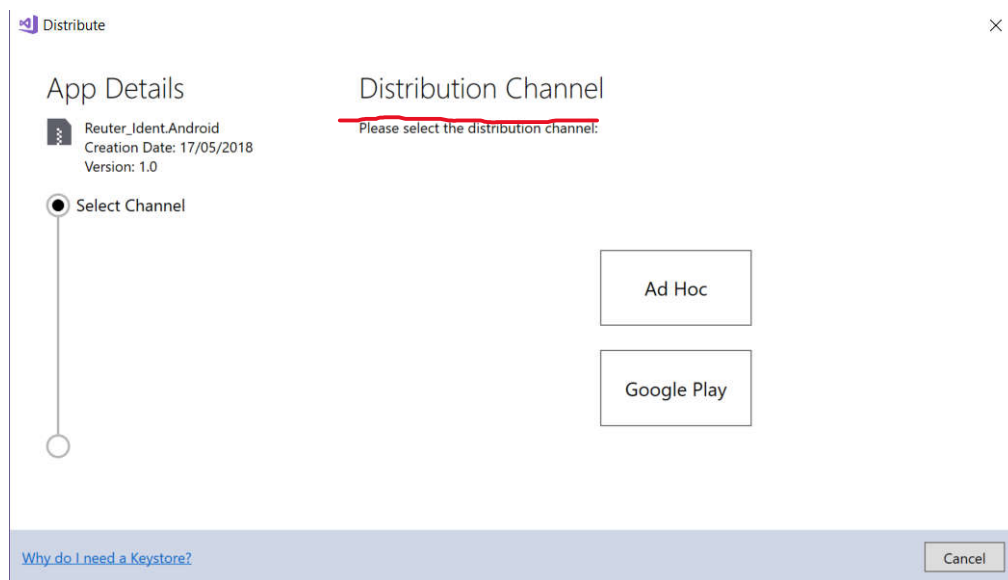
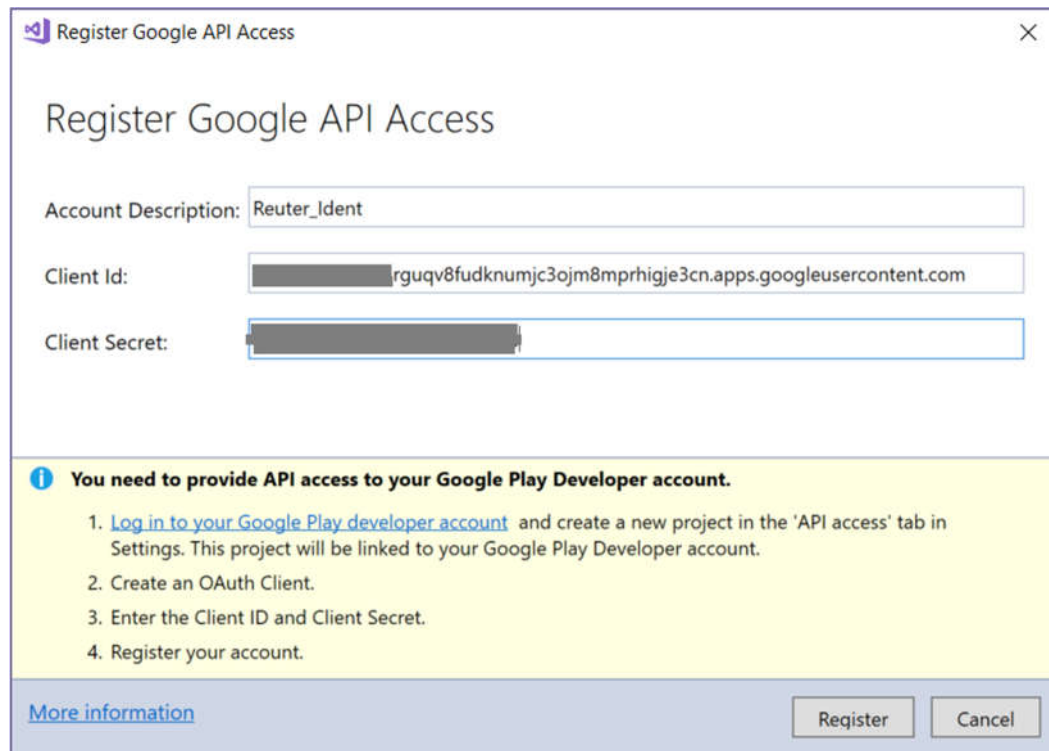


Figure 5.7.8 Select Distribute channel

Choose Google Play as the publishing channel. Then, an Android Key store needed to be created for storing the basic information of the creator. After the creation, the Google Play developer account is needed to connection the Google Play services. [30]



The image shows a Windows-style dialog box titled "Register Google API Access". It contains three input fields: "Account Description" with the text "Reuter_Ident", "Client Id" with a long alphanumeric string, and "Client Secret" which is currently empty. Below these fields is a yellow information box with a blue 'i' icon and the text "You need to provide API access to your Google Play Developer account." followed by a numbered list of four steps. At the bottom left is a blue link "More information", and at the bottom right are two buttons: "Register" and "Cancel".

Register Google API Access

Account Description: Reuter_Ident

Client Id: [redacted]rguqv8fudknumjc3ojm8mprhigje3cn.apps.googleusercontent.com

Client Secret: [redacted]

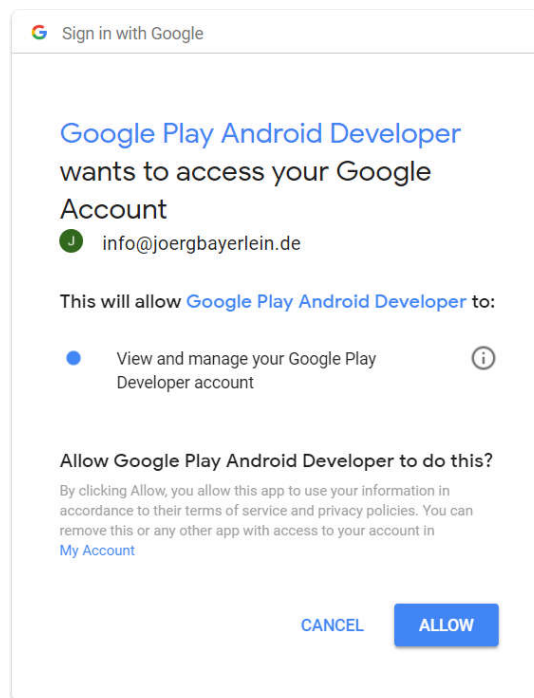
i You need to provide API access to your Google Play Developer account.

1. [Log in to your Google Play developer account](#) and create a new project in the 'API access' tab in Settings. This project will be linked to your Google Play Developer account.
2. Create an OAuth Client.
3. Enter the Client ID and Client Secret.
4. Register your account.

[More information](#) Register Cancel

Figure 5.7.9 Register Google API Access

The Google API Access should be related. This procedure serves the function to connect current application with your google play developer account. The Client Id and Secret can be created in the Google Play Console. Go to the Google Play Console, click the setting, then click the API access. After that click the button “Create OAuth Client”. Finally, copy the client Id and Secret from the OAuth Id. [30]



The image shows a web browser confirm window titled "Sign in with Google". It features the Google logo and the text "Google Play Android Developer wants to access your Google Account". Below this is a profile icon and the email "info@joergbayerlein.de". A section titled "This will allow Google Play Android Developer to:" contains a single permission: "View and manage your Google Play Developer account". At the bottom, it asks "Allow Google Play Android Developer to do this?" and provides a brief explanation of what clicking 'Allow' does. There are "CANCEL" and "ALLOW" buttons at the bottom right.

Sign in with Google

Google Play Android Developer wants to access your Google Account

info@joergbayerlein.de

This will allow Google Play Android Developer to:

- View and manage your Google Play Developer account

Allow Google Play Android Developer to do this?

By clicking Allow, you allow this app to use your information in accordance to their terms of service and privacy policies. You can remove this or any other app with access to your account in [My Account](#)

CANCEL ALLOW

Figure 5.7.10 Web Browser Confirm window

After registration, a web browser confirm window will jump. Allow the operation.

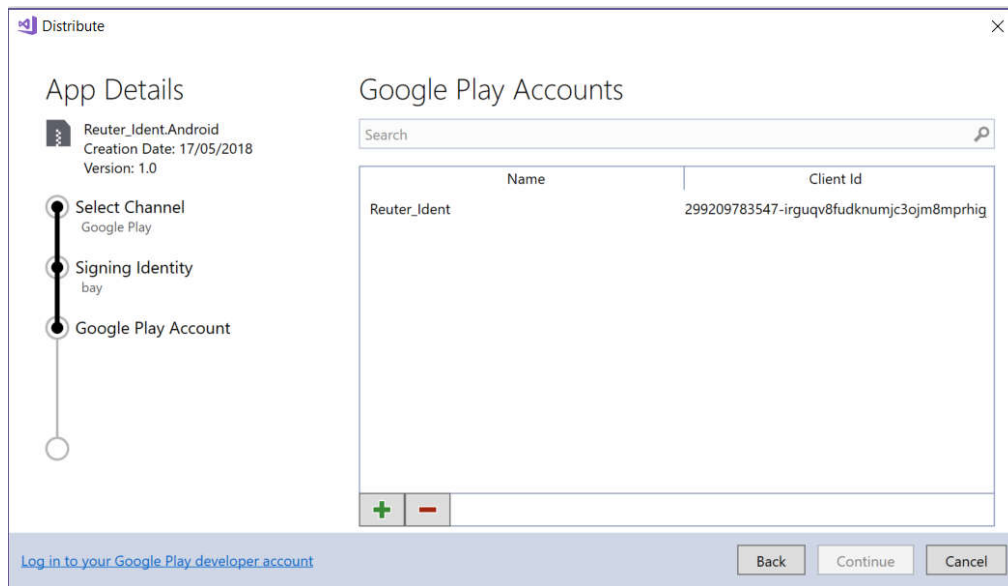


Figure 5.7.11 Choose Google Play Account

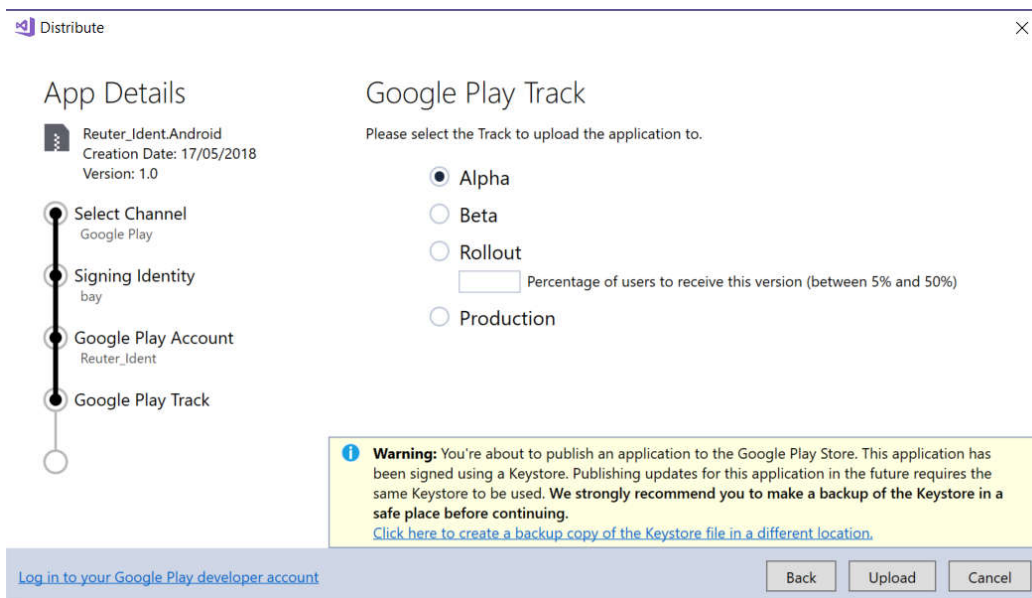


Figure 5.7.12 Select Track to Upload Application

Choose the Alpha version to upload the application.

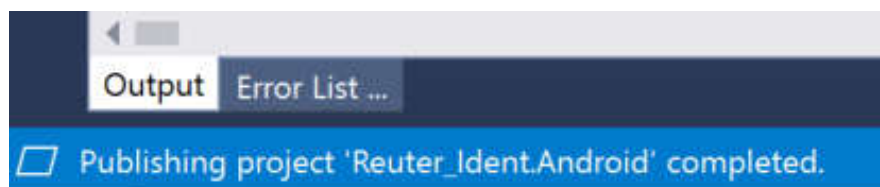


Figure 5.7.13 Successful Publishing

After this operation, the APK file has been submitted on to the Google Play Console. The APK file will be listed in the “Closed Track”. Then the developer could publish it to the “Production Track” in the Google Play Console.

5.7.2 Publishing Via Apple Store

An Apple developer account should be created before the apple application publishing. Then in Xamarin.iOS could package the application on the Visual Studio. However, the further publishing steps should be operated on a Mac with the Xcode. In the following steps, the application packaging procedure and publishing methods will be introduced. What’s more, the sample application “ReuterIdent” will be published through following steps. The procedures can be divided into four parts: Provision the application, Configuration in Visual Studio, Configuration in iTunes Connect and Upload the application to iTunes Connect for distribution. Further information is available from [31].

I Provision the application [31]

To publish an iOS application to App Store, some pre-configuration should be set. The first step is to provision the application for App store distribution. Provision an application means that a digital signature will be generated for application release and installation on devices. It should be completed in the Mac. To provision an application requires a Distribution Profile and a Distribution Certificate. The Distribution Profile contains an App ID. To create these files, the developers should log in to the official website of apple developer. [31]

After login in the apple developer account, choose the Certificate, Identifiers & Profiles.

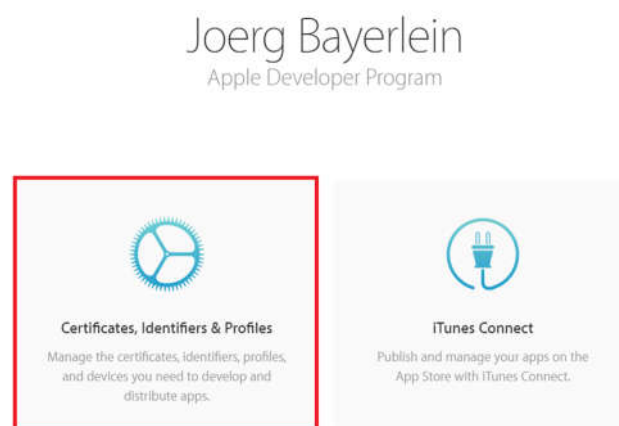


Figure 5.7.14 Login Apple Developer Account

Then the menu page of Certificate, Identifiers & Profiles will be shown. In this page, two files should be created by the developer. The first one is the Distribution Certificate. The second one is Distribution profile.

Firstly, create a Distribution Certificate in the Certificate option. Under the Certificate section, select the Production. Then click the “+” on the right corner. After that choose App Store in the Distribution section. [31]

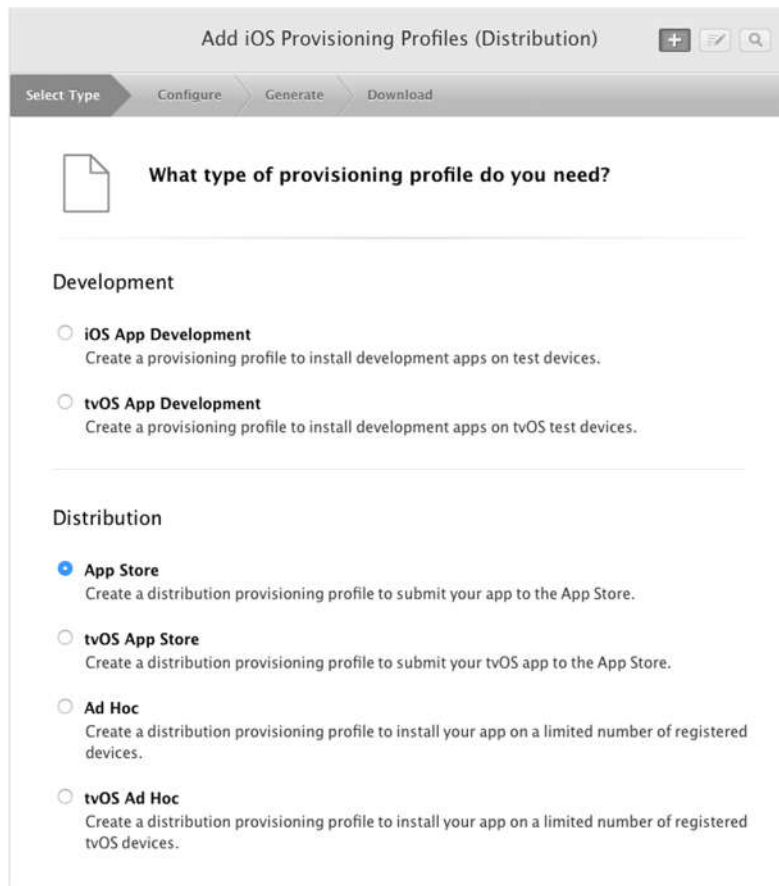


Figure 5.7.15 Create a Distribution Provision File

Click the continue button. Then a CSR (Certificate Signing Request) file is required. It should be created in Mac. [31]

Open the Keychain Access on the Mac, click the submenu and choose the Certificate Assistant. Then choose “Request a certificate from a Certificate Authority”. Then create a certificate. The blank “User Email Address” and “Common Name” should be filled. Then save the certificate into the disk.[32]

After the CSR has been generated, upload it to the web page. Then generate a Certificate. Download the Certificate on the Mac and double-click it to installed it in the Keychain Access. [32]

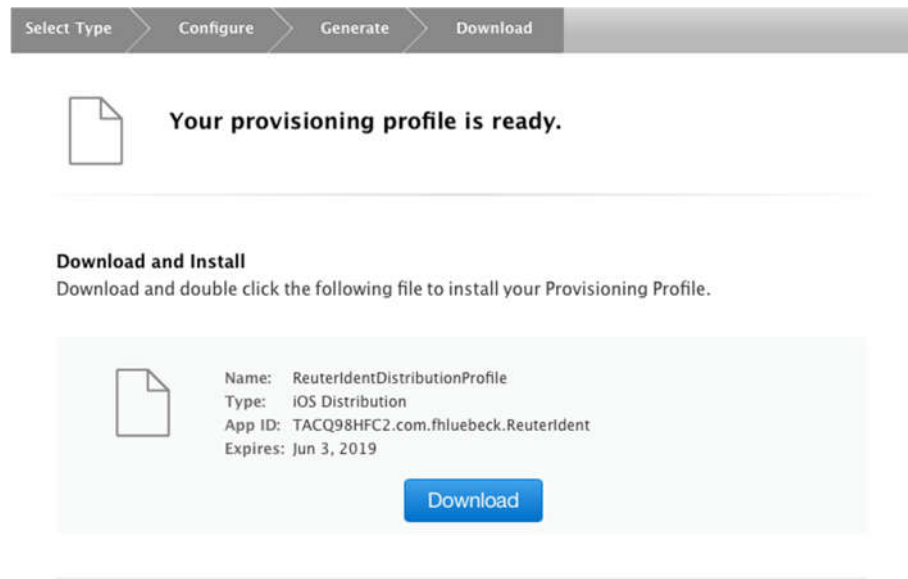


Figure 5.7.16 Download the Certificate



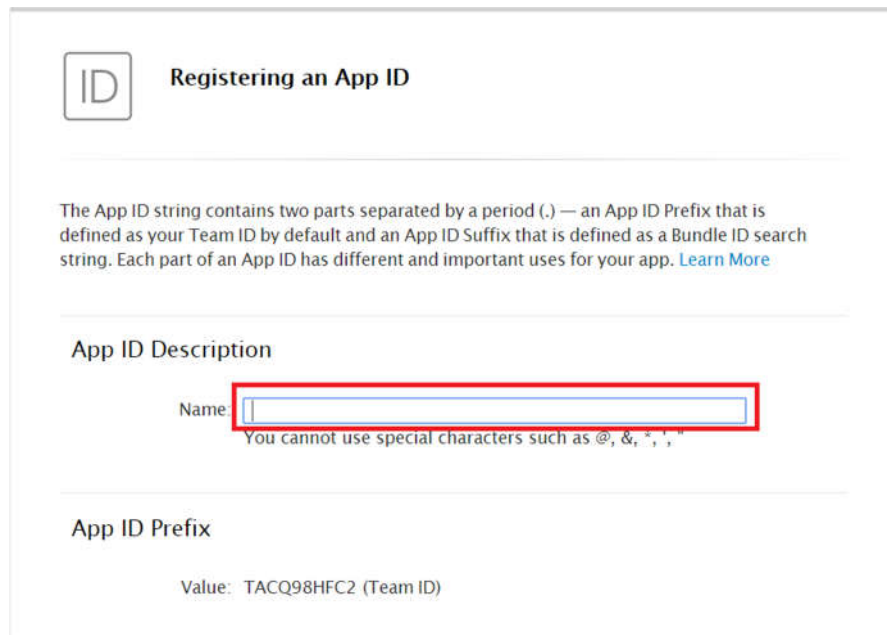
Figure 5.7.17 Install Certificate into Keychain Access

If the certificate installed into the Mac, it can be seen in the Keychain Access.

Secondly, create a Distribution profile.

Before its creation, an App ID should be created. The definition of App ID should be distinguished from Bundle ID. The App ID contains two components, a Team ID and a Bundle ID. It is used to represent one or more applications. However, the Bundle ID is just a unique identifier for an application [33]

To create an App ID, under the Identifiers section, click the App IDs. Then click the “+” to create a new ID.



ID **Registering an App ID**

The App ID string contains two parts separated by a period (.) — an App ID Prefix that is defined as your Team ID by default and an App ID Suffix that is defined as a Bundle ID search string. Each part of an App ID has different and important uses for your app. [Learn More](#)

App ID Description

Name:

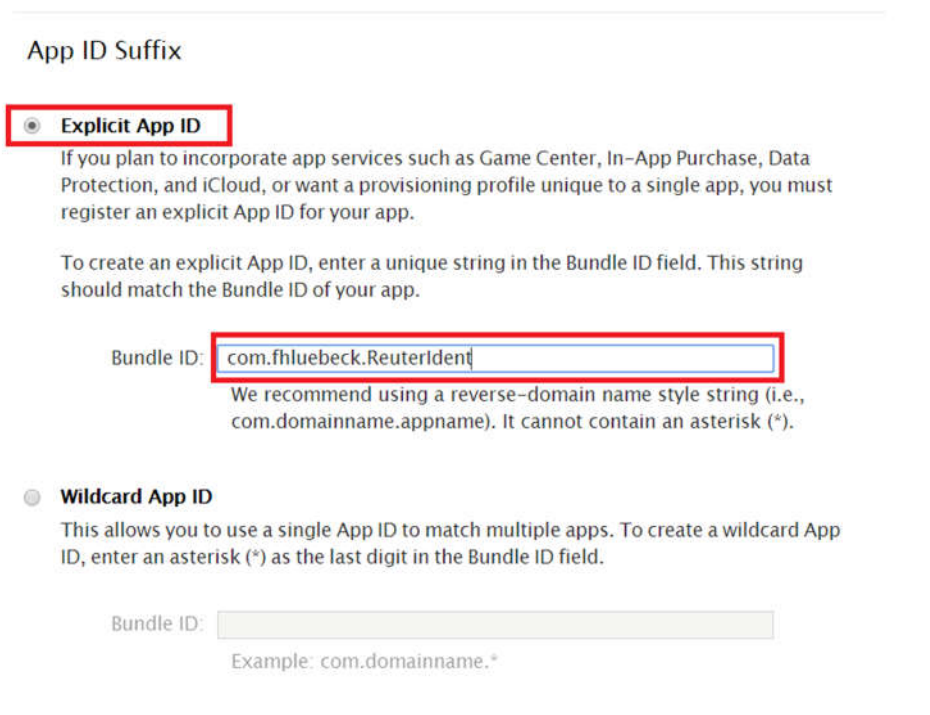
You cannot use special characters such as @, &, *, , , \"

App ID Prefix

Value: TACQ98HFC2 (Team ID)

Figure 5.7.18 Create an App ID 1

In the register page, enter the name of the App ID.



App ID Suffix

☒ **Explicit App ID**

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID:

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

☐ **Wildcard App ID**

This allows you to use a single App ID to match multiple apps. To create a wildcard App ID, enter an asterisk (*) as the last digit in the Bundle ID field.

Bundle ID:

Example: com.domainname.*

Figure 5.7.19 Create an App ID 2

Then choose the Explicit App. Go the Visual Studio, copy the Bundle ID and paste into the entry. The Explicit App ID is used by exact application, while the Wildcard App ID is used for one or more applications. [33]

After that click the register button to create an App ID. Then it can be seen in the App IDs list.

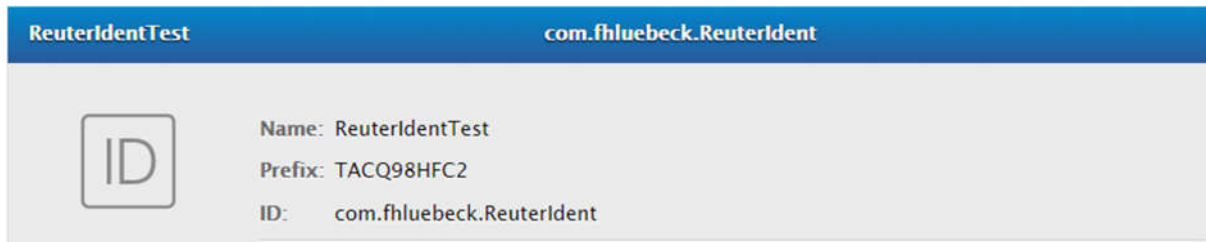


Figure 5.7.20 App ID

Subsequently, create the Distribution Profile. Under the Provisioning Profiles, click the Distribution. Then click the “+” to create a new provision file. [31]

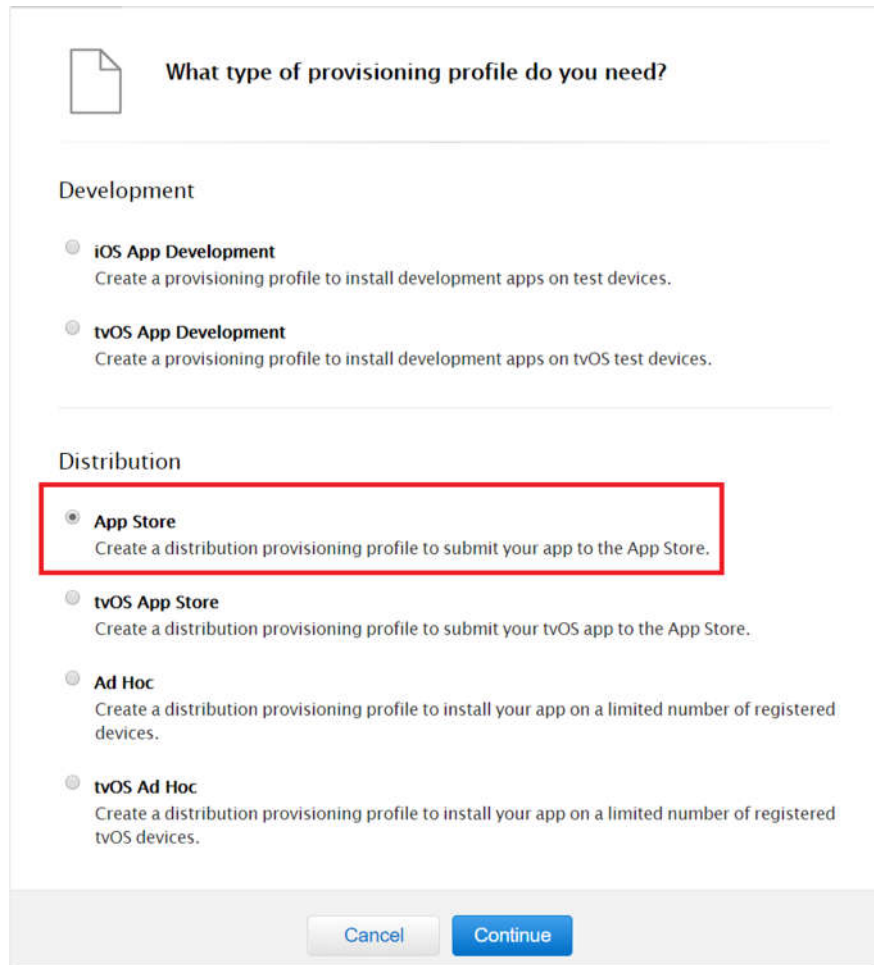
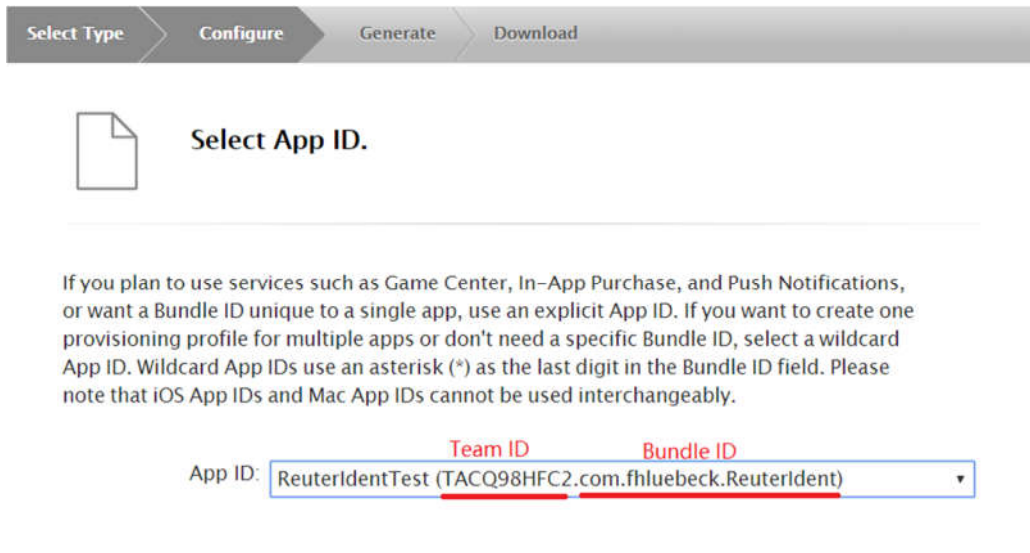



Figure 5.7.21 Create Provision File for App Store Distribution

Choose App Store in the Distribution section. Then click the continue.



The screenshot shows the 'Select App ID' step in the Apple Developer portal. At the top, there is a navigation bar with four tabs: 'Select Type', 'Configure', 'Generate', and 'Download'. The 'Configure' tab is active. Below the navigation bar, there is a document icon and the title 'Select App ID.'. A paragraph of text explains the purpose of App IDs and provides instructions on how to choose one. Below the text, there is a form labeled 'App ID:' with a dropdown menu. The dropdown menu is open, showing the selected App ID: 'ReuterIdentTest (TACQ98HFC2.com.fhluebeck.ReuterIdent)'. Above the dropdown, there are labels for 'Team ID' and 'Bundle ID' in red text.

Select Type Configure Generate Download

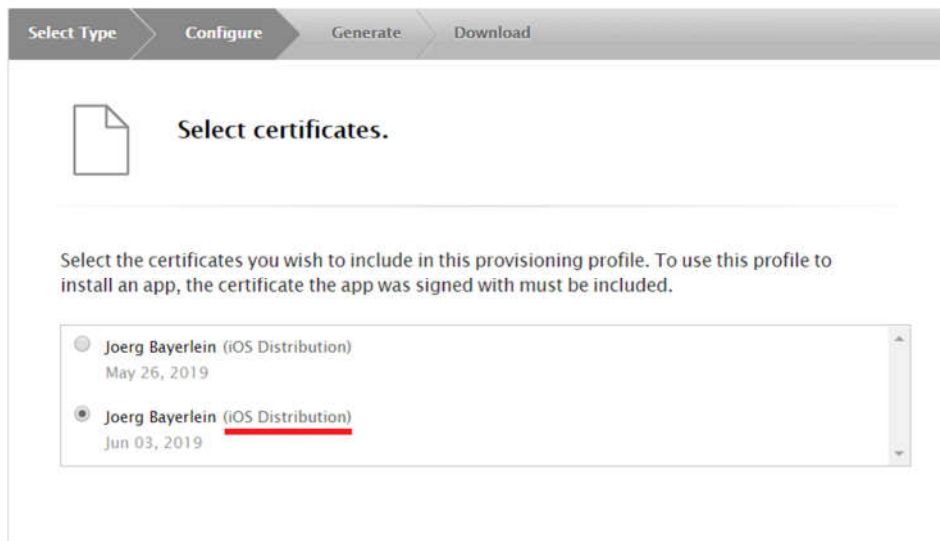
 **Select App ID.**

If you plan to use services such as Game Center, In-App Purchase, and Push Notifications, or want a Bundle ID unique to a single app, use an explicit App ID. If you want to create one provisioning profile for multiple apps or don't need a specific Bundle ID, select a wildcard App ID. Wildcard App IDs use an asterisk (*) as the last digit in the Bundle ID field. Please note that iOS App IDs and Mac App IDs cannot be used interchangeably.

App ID: Team ID Bundle ID
ReuterIdentTest (TACQ98HFC2.com.fhluebeck.ReuterIdent) ▼


Figure 5.7.22 Choose App ID

Choose the App ID in this page. Make sure the App ID is corresponding to the application you want to distribute.



The screenshot shows the 'Select certificates' step in the Apple Developer portal. At the top, there is a navigation bar with four tabs: 'Select Type', 'Configure', 'Generate', and 'Download'. The 'Configure' tab is active. Below the navigation bar, there is a document icon and the title 'Select certificates.'. A paragraph of text explains the purpose of certificates and provides instructions on how to choose one. Below the text, there is a list of certificates. The list contains two entries, both for 'Joerg Bayerlein (iOS Distribution)'. The first entry has a date of 'May 26, 2019' and is not selected. The second entry has a date of 'Jun 03, 2019' and is selected, indicated by a radio button and a red underline.

Select Type Configure Generate Download

 **Select certificates.**

Select the certificates you wish to include in this provisioning profile. To use this profile to install an app, the certificate the app was signed with must be included.

- ☐ Joerg Bayerlein (iOS Distribution)
May 26, 2019
- ☒ Joerg Bayerlein (iOS Distribution)
Jun 03, 2019

Figure 5.7.23 Choose iOS Distribution Certificate

Then choose the iOS Distribution Certificate. Make sure the certificate is used for distribution. Finally, name the file and generate it.

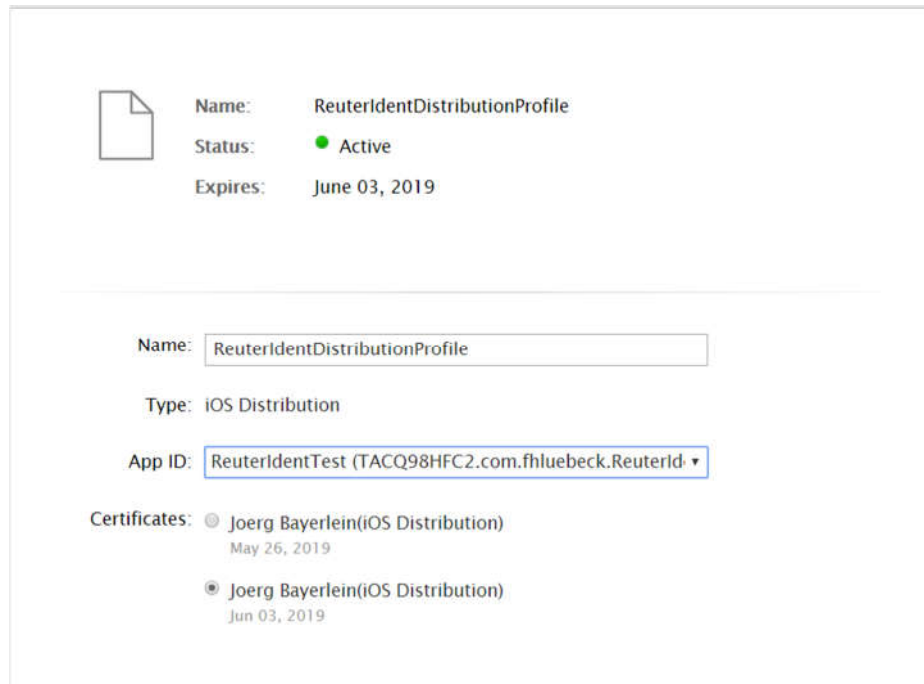


Figure 5.7.24 Generate the Distribution Profile

II Configuration in Visual Studio

In the Visual Studio, the application should be combined with the existed bundle signing. In the property of the iOS project, go to the iOS Bundle Signing. [31]

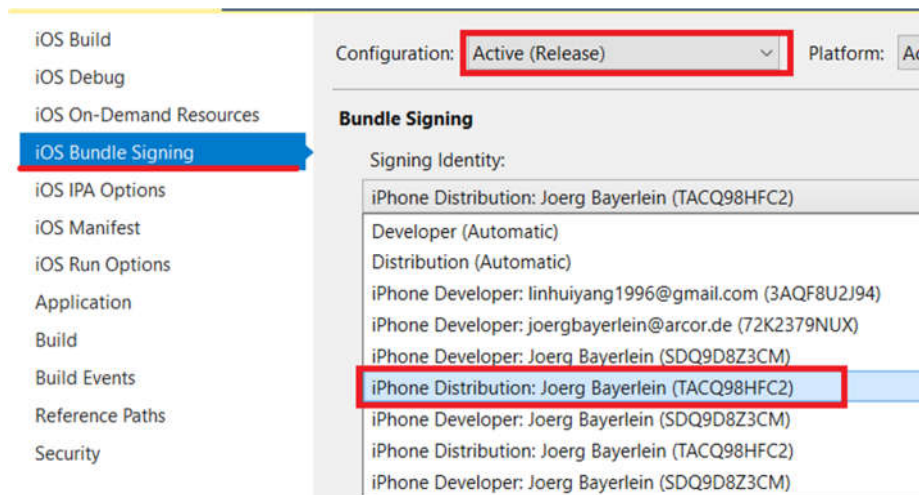


Figure 5.7.25 Configuration in VS 1

Firstly, in the iOS Bundle Signing, change configuration to release. Then choose the signing identity. Once the signing identity has been selected, the provision profile could be chosen. [31]

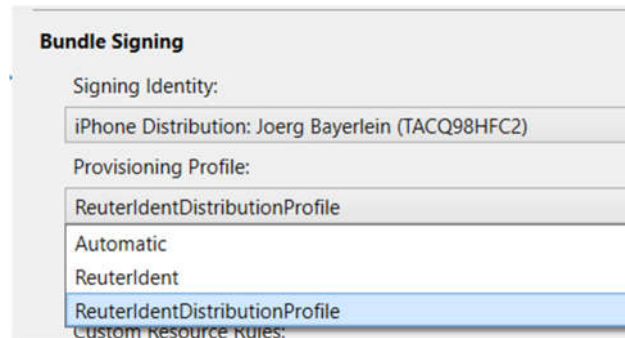


Figure 5.7.26 Configuration in VS 2

The Provision Profile should be the one which was create in the previous steps. Besides, go to the iOS Debug and enable the debugging. Otherwise there will be an error message which will stop the deployment. Subsequently, deploy the device in the Release mode. [31]

Then in the directory “Reuter_Ident.iOS\bin\iPhone\Release” an IPA file is generated. The IPA file is an iOS archived file which store the compressed iOS application. [34] The name of the IFA file can be decided in the iOS IPA Option. Otherwise, the name will be identical to the application name in the info.plist.

III Configuration in iTunes Connect

Once the application has been provisioned, further application configuration should be conducted in iTunes Connect. It is a web-based tool for iOS application management in the App Store. [31]

Then login to the iTunes Connect, create a new iOS application. The most important point is to make sure the Bundle Identifier of the application is the same with the application you want to publish.

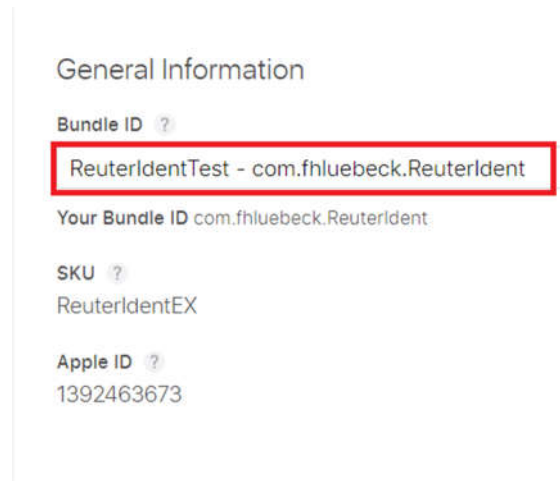


Figure 5.7.27 Information of the application created in iTunes Connect

The following configuration can be completed according to the instruction given in the iTunes Connect. So, here the detail description will not be repeated. More detail information is available on the website [35].

IV Upload the application to iTunes Connect for distribution

Once the IPA file has been created, and the application on iTunes Connect has been configured. The application build can be upload to App Store through iTunes Connect. Then after the review of Apple, this application can be download in the App Store. [31]

Firstly, copy the IPA file from windows to Mac. Then upload by the Application Loader which is provided by Xcode. It can also be downloaded in the App Store. Choose the file in the Application Loader. If the IFA is successfully provisioned, then it can be delivered.



Figure 5.7.28 Deliver IFA file in Application Loader

Then click the Next and wait for uploading.

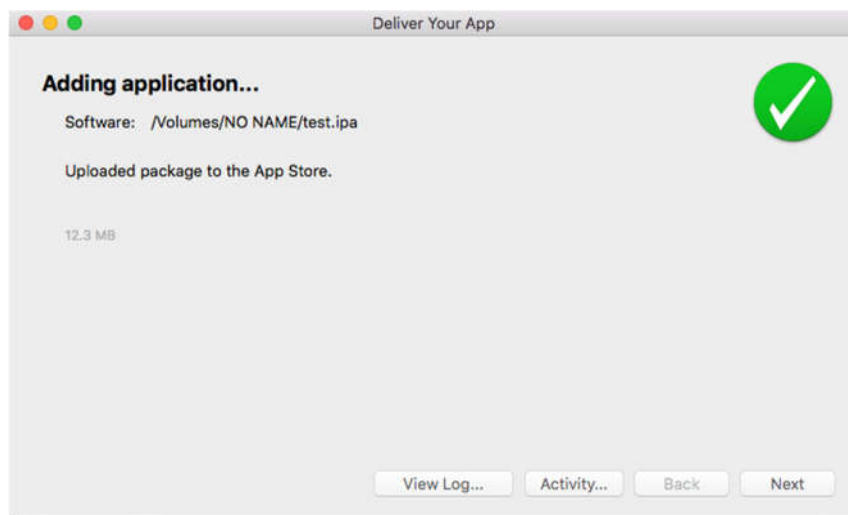


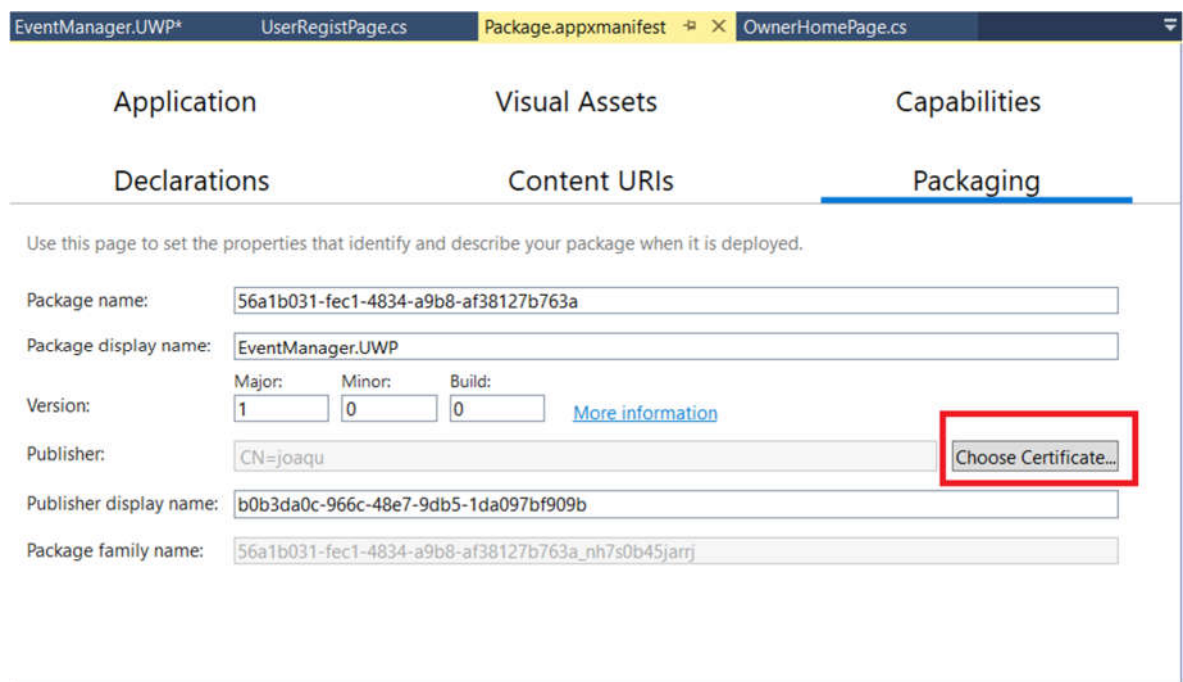
Figure 5.7.29 Successful submission of IFA

5.7.3 Publishing Application Privately in Win10

Microsoft provides three kinds of package: App Package(.appx), App Bundle(.appxbundle) and App Package Upload File(.appxupload). [36] The first package is a format that can be sideloaded on device but it could not be submitted to Microsoft Dev Center.[36] The second package contains multiple application packages, and each of the package is built to support a specific device architecture.[36] The last package is a single file that can contain multiple application packages or an app bundle to support various processor architectures. [36]

In this thesis, the second package was created, and then the application will be sideloaded on other Win 10 machines. Detail explanations are given in the following steps.

Step 1. Configure Package.appxmanifest file [36]



EventManager.UWP* UserRegistPage.cs Package.appxmanifest OwnerHomePage.cs

Application Visual Assets Capabilities

Declarations Content URIs Packaging

Use this page to set the properties that identify and describe your package when it is deployed.

Package name: 56a1b031-fec1-4834-a9b8-af38127b763a

Package display name: EventManager.UWP

Version: Major: 1 Minor: 0 Build: 0 [More information](#)

Publisher: CN=joaqu **Choose Certificate...**

Publisher display name: b0b3da0c-966c-48e7-9db5-1da097bf909b

Package family name: 56a1b031-fec1-4834-a9b8-af38127b763a_nh7s0b45jarj

Figure 5.7.30 Configure Package.appxmanifest [36]

The publisher should create a publishing certificate. Click the Choose Certificate button.

Step 2. Create Publishing Certificate[36]

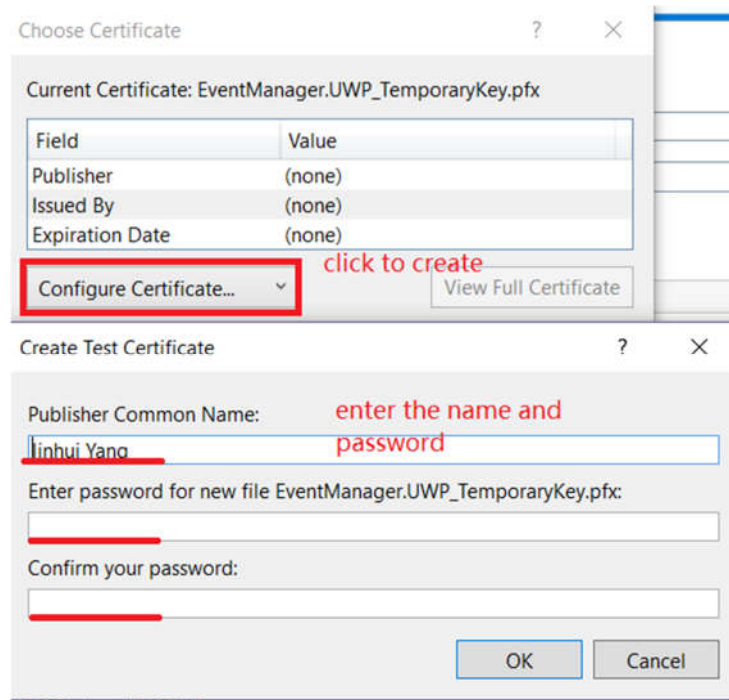


Figure 5.7.31 Create Test Certificate

Then a certificate file will be generated.

Step 3 Create App Package

Click the project, choose store->Create App Package.

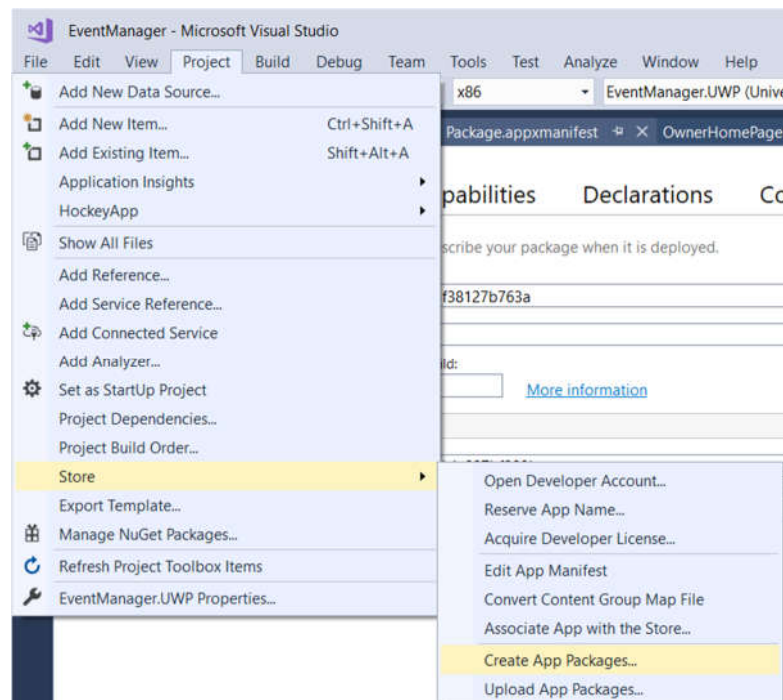


Figure 5.7.32 Create App Package

There are two options for creating a package. If “YES” is selected, a package will be uploaded to the Windows Store. However, it requires the Microsoft developer account, and it is

unnecessary in this thesis. (Xamarin, n.d.) If the package is used for local test or wants to be sideloaded, then “NO” should be selected.



Figure 5.7.33 Create Package Locally

Then the page “Select and Configure Packages” will jump out. In this page, the publishing version and the target version could be chosen.

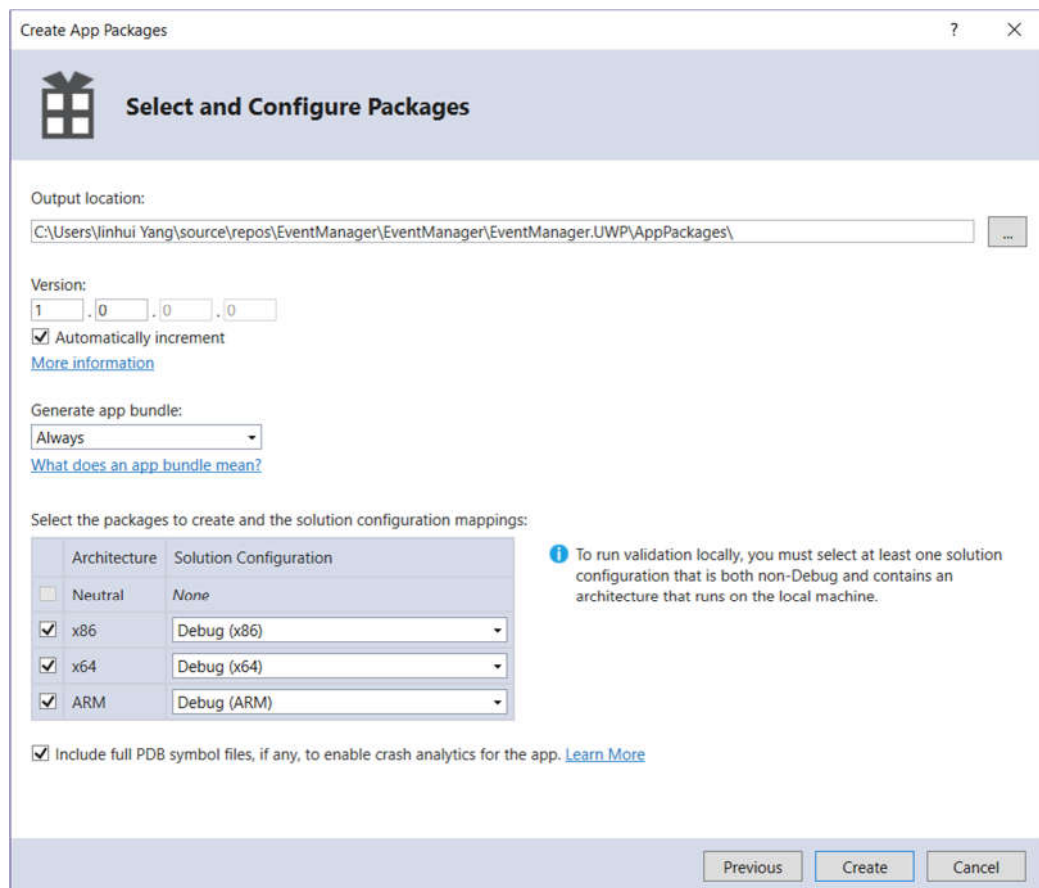


Figure 5.7.34 Select and Configure Packages.

After the configuration and the Certificate checking, the package will be generated in the directory: “EventManager.UWP\AppPackages\EventManager.UWP_1.0.1.0_Debug_Test”.

Then the package could be distributed.

Sideload the application

Once the UWP application has been packaged, it can be installed in other Win 10 machines.

Firstly, run the “Add-AppDevPackage.ps1” with the PowerShell. Following the instruction given by the PowerShell. Allowing all installation conditions. [36]

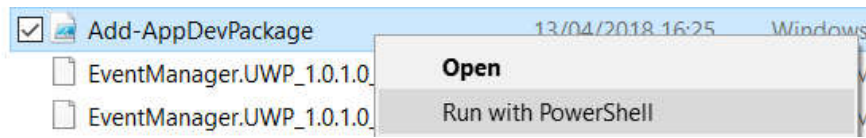


Figure 5.7.35 Run Add-AppDevPackage.ps1 with the PowerShell

Secondly, launch the file ended with “.appxbundle”. Then the application will run on other Win 10 machines.

5.8 Errors reporting

I. MySQL.Data.MySqlClient.MySqlException->”the host does not support SSL Connection”

Due to the UWP use the latest version of MySQL.Data plugin, so the syntax in database connection has changed. In MySQL.Data v8.0.8 or higher, the SSL connection is not supported anymore. It can be solved by adding “SslMode =none” into the database connection string. However, in MySQL.Data.CF which is used in Android and iOS project it is unnecessary to add this condition.

A sample code is shown in the following to indicate how to solve this MySQL.Data problem in UWP project.

```
string connsqlstring = "Server='www.joergbayerlein.de';" +
    "Port=3306;" +
    "User Id=w14515_bav2;" +
    "Database=w14515_loicals;" +
    "charset=utf8;" +
    "SslMode=none";
```

Figure 5.8.1 MySQL.Data error in UWP

II. The crash of Android Emulator

The android emulator stability is always a problem. The version of the plugins will result in the crash of the emulator. Several situations have been tested which would result in the crash of emulator. As a result, some conclusion could be obtained. Summary and Conclusion.

Code sharing project	Android project	Result
MySQL.Data 8.0.11	MySQL.Data.CF(6.9.5.0)	The type MySqlCommand exists in both plugins. Unable to load application.
MySQL.Data 8.0.11	MySQL.Data 8.0.11	Fail to connect the debugger. Unable to load application.
MySQL.Data 6.7.9	MySQL.Data.CF(6.9.5.0)	The emulator is able to launch application load successfully. However, the error message shows

		that MySQL.Data 6.7.9 is not compatible with monoAndroid81.
/	MySQL.Data.CF(6.9.5.0)	Launch successfully.

Table 5.8.1 Version of MySQL.Data in Android and Code sharing projects

III. Java.lang.OutOfMemory

Sometime the application will fail to load due to the shortage of heap memory. It can be solved in the Android Manifest configuration. Click the advance button, set the Java Max Heap Size to 1G.

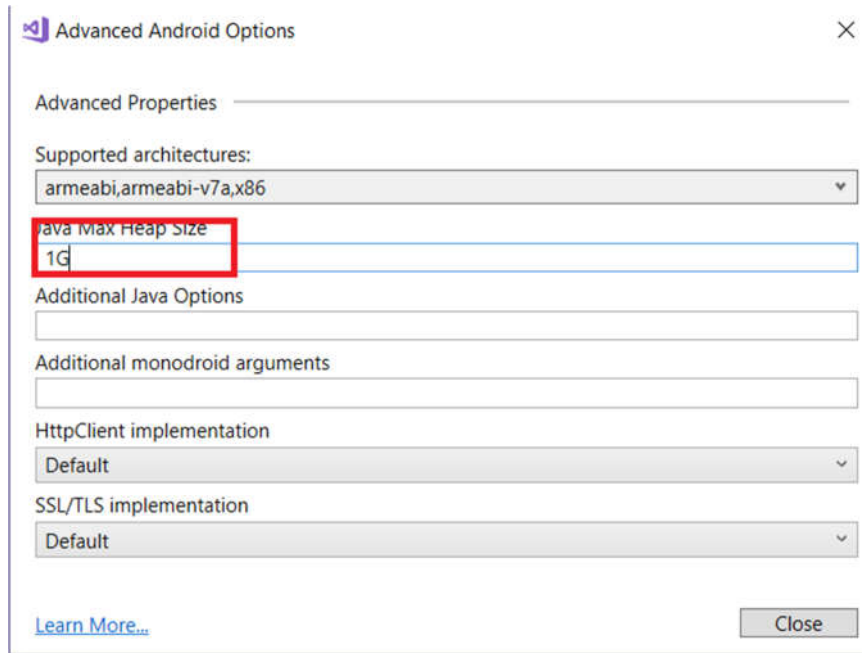


Figure 5.8.2 Java Heap Size Setting

IV. Device.OnPlatform is obsolete

The method `Device.OnPlatform<iOS,Android,WinMobile>` is obsolete due to the elimination of WinMobile API. As a result, it should be replaced. The following example code shows how to fix this problem.

```

1. switch (Device.RuntimePlatform)
2. {
3.     case Device.iOS:
4.         Padding = new Thickness(0, 20, 0, 0);
5.         break;
6.     case Device.Android:
7.         Padding = new Thickness(0, 0, 0, 0);
8.         break;
9.     case Device.UWP:
10.        Padding = new Thickness(0, 0, 0, 0);
11.        break;
12.
13. };

```

Code 5.8.1 Example of Device.RuntimePlatform

V. CornerRadius and BorderRadius

Currently, the `BorderRadius` property of the button is obsolete in the `Xamarin.Forms`. However, when all `BorderRadius` were replaced by `CornerRadius`, the android emulator failed to load the application and crashed. So far, no good solution to solve this problem. It might be solved in the next update of Visual Studio.

VI. Image fails to show

If the picture is added from outside resources or added directly to the source folder, the picture can be seen in the solution, but it can't be shown in the application. So, go to the property of the picture and change the build action to `Embedded resources`.

6 Evaluation

The final goal of this thesis is to discover the possibilities and limitations of the programming tool `Xamarin`. To reach this goal, several tasks were designed: an application development, specific functions development, and the application publishing procedures. In this thesis, the application development was deemed as a method to test the programming tool `Xamarin.Forms`, and it was not planned as a commercial app, so the testing to the application itself in software engineering field is not necessary. The evaluation of this tool is based on the results obtained during the developing time, some parts of assessment have been mentioned in the previous section. In this section, the final evaluation is described from an objective perspective.

Firstly, the realization of the application was successful. The desired function of the components can be realized in the app. Besides, the user interface of the application in three platforms was also implemented successfully. So, it could prove that `Xamarin.Forms` do provide the developers with the abilities to create the cross-platform app which sharing most of the codes.

Secondly, the `Xamarin.Forms` and Visual Studio provides two methods for developers to test their application. The emulators for three platforms are provided. It is a rather cheap and early method to simulate the running of the application. Besides, the physical devices are also available. Currently, the developers could connect an android or iOS device. Testing apps on real devices is faster than testing on emulators. What's more, the physical devices are more stable than emulator because the emulators sometimes would crash due to variety of reasons. Compared with this two testing method, the real devices are more efficient and reliable, but the configuration conditions are more demanding than emulators.

Thirdly, the apps built by `Xamarin.Forms` could be published with the support of `Xamarin.Forms` and Visual Studio. It means that this programming tool provides full app life-cycle development environment. So far, the application published in Google Play Store can be downloaded. The application published to App Store is still being reviewed by Apple. The application published in UWP can be installed on Win 10 machine.

7 Summary

7.1 Possibilities and limitations of Xamarin.Forms

I Possibilities

Firstly, the Xamarin.Forms and Visual Studio provide the opportunity to create the cross-platform application with native appearance. What's more, the Microsoft has unified the all win 10 platforms, which means that the one UWP application can be run on all win10 platforms.

Secondly, the Xamarin itself provides all essential programming functions. The extra functions required by developers could be realized by adding plugins. What's more, the introducing of NuGet Package makes the management of plugins much more manageable.

Thirdly, the database connection function can be realized by adding plugins.

Fourthly, the applications can be distributed on different platforms. Besides, the Microsoft provides comprehensive technical documents which could solve most of common problems.

II Limitation

Firstly, the compatibility of plugins is not so stable. During the development, a routing plugin's update would result in the crash of the emulator or failing to launch the application. Sometimes the problem could be solved in the next update, but some problem still unsolved. For example, Xamarin.Forms 3.x version doesn't support the plugins which run under the Xamarin.Forms 2.x version. As a result, all plugins should be updated synchronously.

Secondly, the performance of database plugin still needs to be improved. It takes rather long time to make the first connection to the database server.

Thirdly, no visual application display during the interface development. The appearance of the application can only be checked when it is debugged which is inconvenient.

Fourthly, the supportive technique documents can't catch the pace of Visual Studio's update rate. Previous documents' instructions can't solve some new problems.

Fifthly, the mac is still needed for iOS application developing and distribution. The developers can't produce the application on only one machine. What's more, the conditions to pair the Win 10 to the Mac are demanding. It requires developers to have a deep understanding of operating Mac OS.

7.2 Conclusion and Outlook

To help readers to understand my thesis, the thesis is introduced progressively. The background information about the Xamarin.Forms are introduced in the second chapter to help readers have a concept what Xamarin.Forms are. Then the software installation and configuration procedures are described in section two and three. Later in the design phase, the approaches designed to be used in the application are explained, so that the readers could have a rough idea about my application.

In the implementation part, basically, all tasks required in the specifications were successfully fulfilled. The old application "Location Finder"[24][25] was successful loaded and integrated into "Event Manager". It can give readers some inspirations for how to transplant a project built

in PCL to the .NET Standard. Some part of the user interfaces were inherited from an old application, and those pages were modified. The participant parts' interfaces were created by the author. The application "Event Manager" could run on Android emulator, iOS emulator/physical device, and UWP emulator. All three layers: represent layer, business logic layer, and database layer was realized by C# codes.

In the following part, the methods of application publishing to multiple platforms were investigated and described. It can give readers instruction how to publish their application in the last part of the application lifecycle.

Finally, the problems faced during the development were concluded which could inspire the readers who might have the same issues.

According to the three months of developing experience on Xamarin.Forms, some conclusion can be drawn. The Xamarin.Forms is a powerful tool for cross-platform application development. With the support of the Visual Studio, it has robust functionality and scalability. Basically, it can bring the majority of the functions required by developers with its increasingly sophisticated plugins packages. What's more, it has a rather high self-update rate. In each update, more capacity will be added to, and the existed bugs could be repaired. Though currently, the compatibility of this tool is not so stable, it will mature shortly with the technical support by not only Microsoft but also millions of enthusiastic cross-platform developers.

However, some improvements could be added to the application "Event Manager" to enhance the performance. What's more, due to the time limitation of the time, some problems haven't been solved yet. Thus, in this section, some potential problem will be stated for further research.

Firstly, the performance of the database connection needs to be improved. It takes about 10 seconds to connect the database and retraced data. But once the database has been connected, the data searching and operations will get faster on a large scale. For example, when the user wants to log in, it will take about 10 seconds in UWP emulator, 12 seconds in Android emulator and about 8 seconds in the iOS emulator. A hypothesis is raised that it might take some time to connect to MySQL Server. If a local database is used, the time consumption might be reduced.

Secondly, because of the low readability of previous students' code, some functions have the problem of ambiguity which might result in the code logic conflict. To solve this problem, most parts of the functions have been annotated and explained to increase the readability. However, the amount of the code is too large; not all codes are covered. What's more, in the integration phase, the codes weren't optimized. To some extends, the performance of the application "Event Manager" can be increased on a large scale.

Thirdly, the stability of the Android Emulator is always a big problem. During the development phase, about 13 working days were spent on investigating the reason for the unreasonable crash. A hypothesis to explain this situation is that the Android emulator is incompatible on author's deploying machine. It might be stable on other computers.

Fourthly, the email verification function in the account registration could be added to increase the security of the application.

Acknowledgment

First and foremost, I want to appreciate sincerely my supervisor Professor Bayerlein who provided me with all necessary facilities for thesis and valuable instructions at each stage of the thesis. Thanks to his enlightened guidance and patience, I can finish my thesis successfully.

Secondly, I would like to show my gratitude to Professor Heeren who is willing to become my second supervisor and gives me instruction about my thesis.

Thirdly, I shall extend my appreciation to Ms. Hanesova for her kindly help when I faced problem in thesis writing.

Last but not least, I would also like to thank previous students Yongsheng Huang and Haocheng Liu for their bachelor thesis which gives me referential instructions.

Appendix 1 List of Figures

Figure 3.1.1 Version of Visual Studio 2017.....	6
Figure 3.1.2 Installed Component in Visual Studio 2017	6
Figure 3.1.3 MySQL Workbench Installation.....	7
Figure 3.1.4 MySQL database Connection	7
Figure 3.2.1 UWP Targeting SDK	8
Figure 3.2.2 UWP Emulator (Win 10 SDK 15063)	8
Figure 3.2.3 Visual Studio Emulator for Android installation.....	9
Figure 3.2.4 Hyper-V manager	9
Figure 3.2.5 Hyper-V manager Emulator Setting	10
Figure 3.2.6 Set Permission for Remote Login.....	10
Figure 3.2.7 Pair to Mac.....	11
Figure 3.2.8 iOS Setting for Remote Simulator to Windows.....	12
Figure 3.3.1 Bundle Identifier in Xcode	13
Figure 3.3.2 Bundle Identifier in Visual Studio	13
Figure 3.3.3 Connect real device.....	13
Figure 3.3.4 Enable Device Debug 1	14
Figure 3.3.5 Enable Device Debug 2	14
Figure 4.3.1 UWP DatePicker.....	17
Figure 4.3.2 UWP TimePicker	17
Figure 4.3.3 UWP time format.....	17
Figure 4.3.4 Android DatePicker	17
Figure 4.3.5 Android TimePicker	17
Figure 4.3.6 Android time format	17
Figure 4.3.7 iOS TimePicker.....	18
Figure 4.3.8 iOS DatePicker	18
Figure 4.3.9 iOS time format	18
Figure 5.2.1 Use case diagram for Event Manager	21
Figure 5.3.1 Create a new Project	22
Figure 5.3.2 Generate a Blank App.....	22
Figure 5.3.3 Copy .cs file from “OwnerBay”	23
Figure 5.3.4 Rename Namespace 1	23
Figure 5.3.5 Rename Namespace 2	24
Figure 5.3.6 Change Namespace and Dependency Service	24
Figure 5.3.7 Plugins in Code Shared Project	25
Figure 5.3.8 Install MySQL.Data into UWP Project	25
Figure 5.3.9 MySQL Plugin for iOS and Android.....	25
Figure 5.3.10 Adding MySQL reference manually.....	26
Figure 5.3.11 Adding System.Data reference	26
Figure 5.3.12 Initialize Maps in UWP	28
Figure 5.3.13 Initialize Maps in iOS[27]	28
Figure 5.3.14 Initialize Maps in MainActivity.....	29
Figure 5.3.15 Configure the permissions for Emulator.....	29

Figure 5.4.1 Choose an Identity in Event Manager.....	30
Figure 5.4.2 Organizer Login Page	31
Figure 5.4.3 Register an account for Organizer	31
Figure 5.4.4 Organizer Event Page	32
Figure 5.4.5 Organizer Location Page	32
Figure 5.4.6 Edit the event	32
Figure 5.4.7 Event search page	33
Figure 5.4.8 Event detail page.....	33
Figure 5.4.9 Event Location Detail	33
Figure 5.5.1 UWP Language Setting Page.....	34
Figure 5.5.2 iOS Language Setting Page	34
Figure 5.5.3 Android Language Setting Page	34
Figure 5.5.4 Autofill account information	36
Figure 5.5.5 Change Decimal symbol.....	38
Figure 5.5.6 UWP Dynamic Event Picker	39
Figure 5.5.7 UWP Dynamic Event Picker Items	39
Figure 5.5.8 New Event type entry page.....	39
Figure 5.5.9 Detail of Event Type table	39
Figure 5.5.10 Event Type Table Sample.....	40
Figure 5.5.11 Event Filter Realizing Logic.....	45
Figure 5.6.1 Android DisplayActionSheet 1	46
Figure 5.6.2 Android DisplayActionSheet 2.....	46
Figure 5.6.3 UWP DisplayActionSheet 1	47
Figure 5.6.4 UWP DisplayActionSheet 2	47
Figure 5.6.5 iOS DisplayActionSheet 1	47
Figure 5.6.6 iOS DisplayActionSheet 2	47
Figure 5.6.7 Install Rg.Plugins.Popup.....	48
Figure 5.6.8 Android PopUp1	49
Figure 5.6.9 Android PopUp2	49
Figure 5.6.10 UWP PopUp1	49
Figure 5.6.11 UWP PopUp2	49
Figure 5.6.12 iOS PopUp1	50
Figure 5.6.13 iOS PopUp2	50
Figure 5.7.1 Android Application icon configuration.....	51
Figure 5.7.2 Set the Version Number.....	51
Figure 5.7.3 Shrink the APK[30]	52
Figure 5.7.4 Protect the Application from unapproved debugging.....	52
Figure 5.7.5 Set Package Property for archiving	52
Figure 5.7.6 Archive the Android application.....	53
Figure 5.7.7 Archived result.....	53
Figure 5.7.8 Select Distribute channel	53
Figure 5.7.9 Register Google API Access.....	54
Figure 5.7.10 Web Browser Confirm window	54
Figure 5.7.11 Choose Google Play Account	55

Figure 5.7.12 Select Track to Upload Application.....	55
Figure 5.7.13 Successful Publishing	55
Figure 5.7.14 Login Apple Developer Account	56
Figure 5.7.15 Create a Distribution Provision File	57
Figure 5.7.16 Download the Certificate	58
Figure 5.7.17 Install Certificate into Keychain Access.....	58
Figure 5.7.18 Create an App ID 1	59
Figure 5.7.19 Create an App ID 2	59
Figure 5.7.20 App ID	60
Figure 5.7.21 Create Provision File for App Store Distribution	60
Figure 5.7.22 Choose App ID	61
Figure 5.7.23 Choose iOS Distribution Certificate	61
Figure 5.7.24 Generate the Distribution Profile	62
Figure 5.7.25 Configuration in VS 1	62
Figure 5.7.26 Configuration in VS 2	63
Figure 5.7.27 Information of the application created in iTunes Connect	63
Figure 5.7.28 Deliver IFA file in Application Loader	64
Figure 5.7.29 Successful submission of IFA.....	64
Figure 5.7.30 Configure Package.appxmanifest [36].....	65
Figure 5.7.31 Create Test Certificate	66
Figure 5.7.32 Create App Package.....	66
Figure 5.7.33 Create Package Locally	67
Figure 5.7.34 Select and Configure Packages.....	67
Figure 5.7.35 Run Add-AppDevPackage.ps1 with the PowerShell.....	68
Figure 5.8.1 MySQL.Data error in UWP	68
Figure 5.8.2 Java Heap Size Setting.....	69

Appendix 2 List of Table

Table 2.3.1.NET Implementation Support [7]	5
Table 4.4.1 App class properties [21]	19
Table 4.5.1 Methods in IDictionary Interface [22]	20
Table 5.3.1 Plugins needed for each project	27
Table 5.5.1 Database Table Description	40
Table 5.5.2 Sample of adding assembly in the implementation class.....	40
Table 5.8.1 Version of MySQL.Data in Android and Code sharing projects	69

Appendix 3 List of Code

Code 4.1.1 Create a tabbed page [13]	15
Code 4.3.1 Sample code of Picker	16
Code 4.3.2 Syntax for DatePicker [20]	18

Code 4.3.3 Syntax for Time Picker	18
Code 4.3.4 Get value from TimePicker and DataPicker	19
Code 4.5.1 Syntax for IDictionary Interface	19
Code 4.6.1 Page Navigation Method [23]	20
Code 5.3.1 Initialize Maps in UWP [27]	28
Code 5.3.2 Initialize Maps in iOS [27]	28
Code 5.3.3 Initialize Maps in Info.plist [27]	29
Code 5.3.4 Initialize Maps in Android [27]	29
Code 5.5.1 Retrieve switch status	35
Code 5.5.2 Entry instantiation of Username and Password	36
Code 5.5.3 Store the parameter into the local file	37
Code 5.5.4 Load User Account Information	37
Code 5.5.5 Get Location Information	38
Code 5.5.6 using MySQL Plugin namespace	40
Code 5.5.7 Register Main information [24]	41
Code 5.5.8 Get Event Detail Information [24]	42
Code 5.5.9 Update Event Detail [24]	43
Code 5.5.10 SQL Time filter searching sentence	43
Code 5.5.11 SQL Type filter searching sentence	43
Code 5.5.12 Range filter code fragment [25]	44
Code 5.5.13 SQL Range Filter Searching Sentence	44
Code 5.6.1 Initialize a Popup Menu by DisplayActionSheet	46
Code 5.6.2 Using Plugin's Namespace	48
Code 5.6.3 Create new Pop-up Page	48
Code 5.8.1 Example of Device.RuntimePlatform	69

Appendix 4 Content of USB Sticker

1. database tables
2. the source code of "Event Manager"
3. Linhui Yang thesis paper (Word version)
4. Linhui Yang thesis paper (PDF version)
5. The source code of "ReuterIdent"
6. Specification of Event Manager(word)
7. Screenshots of "EventManager" (Marked with the name of View Components)

Appendix 5 Reference

- [1] Microsoft, Programming Concept(C#)(07/20/2015). Contributors: Bill Wagner, Maira Wenzel, Mike B, Luke Latham. Retrieved from URL: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/index> [accessed at 7th.06.2018]
- [2] Nat Friedeman(May 28,2014). Announcing Xamarin 3. Retrieved from URL: <https://blog.xamarin.com/announcing-xamarin-3/> [accessed at 6th.06.2018]

- [3] Microsoft, An introduction to Xamarin.Forms(12/02/2016), contributors: David Britch, Matt Cooper, Craig Dunn, Brad Umbaugh. Retrieved from URL: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/get-started/introduction-to-xamarin-forms> [Accessed at 23th.05.2018]
- [4] Petzold,C.(2016), Creating Mobile Apps with Xamarin.Forms, Redmond: Microsoft Press, Chapter 7, Page 132
- [5] Microsoft, Share Code Overview (23th.03.2017). Contributors: Amy Burns, Brad Umbaugh, Craig Dunn. Retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/code-sharing>. [Accessed at 3rd.05.2018]
- [6] Immo Landwerth. Introducing .NET Standard(Sep 26, 2016). Retrieved from URL :<https://blogs.msdn.microsoft.com/dotnet/2016/09/26/introducing-net-standard/> [accessed at 21th.05.2018]
- [7] Microsoft, .NET Standard. Contributors: Maira Wenzel, Timotai Dolean, Mike Jones, Macro Rossignoli, Tompratt-AQ. Retrieved from URL: <https://docs.microsoft.com/en-us/dotnet/standard/net-standard> [Accessed at 22th.05.2018]
- [8] Joseph Hill, Hello NuGet! A New Home for Xamarin Components (Nov.20.2017). Retrieved from URL: <https://blog.xamarin.com/hello-nuget-new-home-xamarin-components/> [Accessed at 9th.05.2018]
- [9] Microsoft, Visual Studio Download. Retrieved from URL: <https://www.visualstudio.com/downloads/> [accessed at 25th.05.2018]
- [10] MySQL, Download MySQL Workbench. Retrieved from URL : https://dev.mysql.com/downloads/workbench/?utm_source=tuicool [accessed at 25.05.2018]
- [11] Microsoft, Choose a UWP Version(04/10/2018). Contributor: Quinn Radich, Michael Satran, Mike Jacobs, Tyler Whitney, Thomas Claudius Huber, Alexander Koren. Retrieved from URL : <https://docs.microsoft.com/en-us/windows/uwp/updates-and-versions/choose-a-uwp-version> [accessed at 6th.06.2018]
- [12] Microsoft, Xamarin.Forms.Page(01/12/2016), Contributors: David Britch, Craig Dunn, Charles Petzold. Retrieved from URL: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/controls/pages> [accessed at 6th.06.2018]
- [13] Microsoft, Tabbed Page(07/10/2017). contributor: David Britch, Craig Dunn, Charles Petzold, Retrieved from <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/navigation/tabbed-page> [Accessed at 23th.05.2018]
- [14] Microsoft, Layouts(10/26/2017). Contributor: David Britch, Craig Dunn, Brad Umbaugh. Retrieved from URL: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/layouts/> [accessed at 6th.06.2018]

- [15] Microsoft, Stack Layout(11/25/2015), contributor: David Britch, Crag Dunn, Brad Umbaugh, retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/layouts/stack-layout> [accessed at 09.05.2018]
- [16] Microsoft, ScrollView(04/22/2016). Contributors: David Britch, Craig Dunn, Brad Umbaugh. Retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/layouts/scroll-view> [accessed at 6th.06.2018]
- [17] Microsoft,Grid(10/26/2017). Contributors: David Britch, Gerald Versluis, Craig Dunn, Brad Umbaugh,Charles Petzold. Retrieved from URL: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/layouts/grid> [accessed at 6th.06.2018]
- [18] Petzold, C.(2016), Creating Mobile Apps with Xamarin.Forms, Redmond: Microsoft Press, p384.
- [19] Microsoft, Picker Class, Retrieved from URL : <https://docs.microsoft.com/en-us/dotnet/api/Xamarin.Forms.Picker?view=xamarin-forms> [accessed at 01.06.2018]
- [20] Microsoft, Using DatePicker(03/12/2018). Contributors: Charles Petzold, Craig Dunn, Retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/datepicker> [Accessed on 7th.05.2018]
- [21] Microsoft, App Class(02/19/2018). Contributors: David Britch, bestcoder, Craig Dunn, Charles Petzold, Retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/application-class> [accessed at 9th.06.2018]
- [22] Microsoft, IDictionary<TKey,TValue> interface. Retrieved from URL : <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.idictionary-2?view=netframework-4.7.2> [accessed at 7th.06.2018]
- [23] Petzold,C.(2016), Creating Mobile Apps with Xamarin.Forms, Redmond: MicrosoftPress, Chapter 24, Page 921-920
- [24] Yongshen Huang, Thesis paper_YongshengHuang , Fachhochschule luebeck lernraum [Online] . Available at URL : <https://lernraum.fh-luebeck.de/mod/folder/view.php?id=85523> [accessed at 06.06.2018]
- [25] Haocheng Liu, bachelorThesis_Haocheng Liu, Fachhochschule Luebeck lernraum [Online], Available at URL : <https://lernraum.fh-luebeck.de/mod/folder/view.php?id=85523> [accessed at 06.06.2018]
- [26] Being maps | Dev Center, Available at URL : <https://www.bingmapsportal.com/?lc=1033>
- [27] Microsoft, Map(04/27/2016). Contributor: David Britch, Charles Petzold, Brad Umbaugh, Houssem Dbria, Paramjit singh, Craig Dunn. Retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/map> [Accessed at 24.05.2018]

- [28] Microsoft, Display Pop-ups (12/01/2017). Contributors: David Britch, Craig Dunn, Brad Umbaugh, Charles Petzold, Retrieved from URL: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/navigation/pop-ups> [accessed at 15.05.2018]
- [29] GitHub, What is Rg.Plugins.Popup?. Kirill edited on Feb 13, 2018. Retrieved from URL : <https://github.com/rotoorgames/Rg.Plugins.Popup/wiki> [accessed at 13.05.2018]
- [30] Microsoft, Preparing an application for release(03/21/2018). Contributors: Mark Mclemore, Leah Plett, Craig Dunn, Brad Umbaugh. Retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/android/deploy-test/release-prep/?tabs=vswin> [accessed at 17.05.2018]
- [31] Microsoft, App Store Distribution(08/23/2017). Contributors: Brad Umbaugh, Craig Dunn. Retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/ios/deploy-test/app-distribution/app-store-distribution/?tabs=vsmac>. [accessed at 7th.06.2018]
- [32] Microsoft, Manual Provision for Xamarin.Forms. Contributors(07/15/2017): Amy Burns, Brad Umbaugh, Craig Dunn, retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/ios/get-started/installation/device-provisioning/manual-provisioning?tabs=vsmac> [accessed at 06.06.2018]
- [33] Bart Jacobs, What Are App IDs and Bundle Identifiers(Oct 10, 2017). Retrieved from URL : <https://cocoacasts.com/what-are-app-ids-and-bundle-identifiers/> [accessed at 04.06.2018]
- [34] Wikipedia, .ipa(03 Feb,2018). Retrieved from URL : <https://en.wikipedia.org/wiki/.ipa> [accessed at 04.06.2018]
- [35] Microsoft, Configuring an App in iTunes Connect(03/19/2017). Retrieved from URL : <https://docs.microsoft.com/en-us/xamarin/ios/deploy-test/app-distribution/app-store-distribution/itunesconnect> . [Available at 4th.06.2018]
- [36] Microsoft. Package a UWP application with Visual Studio(11/16/2017). Contributors: Lauren Hughes, Genevieve Warren, Michael Satran, Mike Jacobs, Alexander Koren, Retrieved from URL : <https://docs.microsoft.com/en-us/windows/uwp/packaging/packaging-uwp-apps>, [Accessed at 29th.04.2018]