Zusammenfassung der Arbeit Abstract of Thesis	
Fachbereich: Department:	**Electrical Engineering and Computer Science**
Studiengang: University course:	**Information Technology**
Thema: Subject:	**Investigation and Test of Xamarin.Essentials in Visual Studio on Android Phone**

Zusammenfassung:
Abstract:

This thesis intends to investigate the possibilities and limitations of Xamarin.Essentials on Android phone. Xamarin.Essentials is published to simplify the access to the features of different platforms, which gives a huge support to the cross-platform development. A mobile application including the introduction and presentation of the features is implemented to achieve the goal. This thesis serves as both a guideline for developing applications with Xamarin.Essentials and a documentation for investigating different performance of Xamarin.Essentials on various platforms. In this thesis, half of the features are tested and an image picker is implemented without Xamarin.Essentials. The introduction of replacing the application icon is contained as well. The investigation of this NuGet package shows that it reduces the workload to get access of sensors and information on various devices. The features of the package can be available as long as they are supported on the device. However, the huge package will be the burden of the application if only one single feature is used. Despite of this limitation and the deficiency of functions in some features, Xamarin.Essentials has shown the potential to be a suitable tool to implement cross-platform applications. More useful features might be supplemented into the package.

Verfasser: Author:	**Qin Lu**
Betreuender Professor/in: Attending Professor:	**Prof. Dr. Jörg Bayerlein**
WS / SS:	**SS <2019>**

Table of Contents

1 Introduction

1.1 Motivation

The world currently confronts the fast development of mobile smartphones. Meanwhile, iOS, Android, UWP become the biggest and most critical mobile platforms. In order to reduce the duplication and effort on programming on different platforms, the investigation on cross-platform application development keeps expanding. According to Alberto Furlan, the cross-platform application market has exceeded 7.5 billion dollars until 2018 [1]. As one of the most powerful cross-platform developing tool, Xamarin provides the possibility that applications on various platforms can be developed by shared codes without considering different user interfaces [23]. At the same time, Xamarin.Essentials is published to simplify the access to the features of different platforms, including accelerometer, barometer and so on. Therefore, it is highly desirable that definite documentation about Xamarin.Essentials can be provided.

1.2 Goal

This thesis intends to investigate the possibilities and limitations of Xamarin.Essentials on Android phone. To realize this intention, this thesis tries to find a good demonstration of the features in Xamarin.Essentials. In other words, a mobile application including the introduction and presentation of the features is implemented. Additionally, this thesis can offer as a guideline for those who aim at developing with Xamarin.Essentials or as the documentation for those who focus on investigating the different behaviors of Xamarin.Essentials on various platforms.

1.3 Organization

This thesis contains five essential sections. The first section aims at the introduction of the whole thesis. In the second section, some basic knowledge and background information about associated software and API (application programming interface) are presented. In the next section, the preparation steps of software installation and emulator configuration are provided. In the fourth section, the detailed specification and behaviors of the first half of the features of Xamarin.Essentials on different platforms is documented, which is the most essential part of the thesis. Several additional tasks are also included in this section. In the last section, the conclusion and evaluation of this thesis are presented, and suggestions for further research are mentioned as well.

2 Background information and basic knowledge

Xamarin is a software company of Microsoft, which was founded in May 2011. Xamarin.Forms was introduced on May 28, 2014 (Petzold, 2016). Xamarin is a suitable tool for projects about C# and .NET (Reynolds, 2014). One of the most important reasons is that developers do not need to waste their time learning Java or Objective-C to develop iOS or Android applications. In this case, Xamarin can be a substitute for development. Additionally, Xamarin has the benefit of code reuse. It is estimated that codes regarding user interface or device capabilities can be recycled on the platform of Android, iOS and WP8 at most 80 percent (Reynolds, 2014).

2.1 Xamarin.Forms and C#

Xamarin.Forms is a cross-platform development framework, which can be used to develop mobile applications for Android, iOS and Universal Windows Platform (UWP). As mentioned before, codes about the definition of the user interface can be reused due to the user interface abstraction of Xamarin.Forms.

C# is an articulate programming language. However, it is uncomplicated for developers who have an intimate knowledge of C, C++ or Java [7]. C# syntax not only abandons the intricacy part of C++ but also provides new impressive features which are not contained in Java [7]. One of the most typical examples is lambda expression, which can reduce the code needed to be written. Certainly, C# is an objective-oriented programming language like Java. In addition, it can be used to write programs that can run on the .NET Framework. That means, the .NET Framework class library can be utilized by the programs if .NET is accessible on different platforms. As a result, code written in C# based on Xamarin.Forms is platform independent (Petzold, 2016).

2.2 Xamarin.Essentials

Xamarin.Essentials provides various API (application programming interface) on Android, iOS and UWP platforms. Although each platform has a distinct operating system, the cross-platform mobile application developed with these APIs can be accessed from shared code independent of the user interface [23].

2.3 Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) which can be used to develop variable applications including mobile ones [22]. It supports 36 different programming languages including C#. Moreover, there exists plenty of convenient features, which make the development of application efficient. For instance, the refactoring feature helps developers to change all the names of identical parameters or methods at the same time. Besides, it is allowed for developers to choose and install workloads in line with their needs. In this thesis, different emulators with different Android APIs can be chosen, which is the most vital reason to develop with Visual Studio.

2.4 Windows IoT and Raspberry Pi

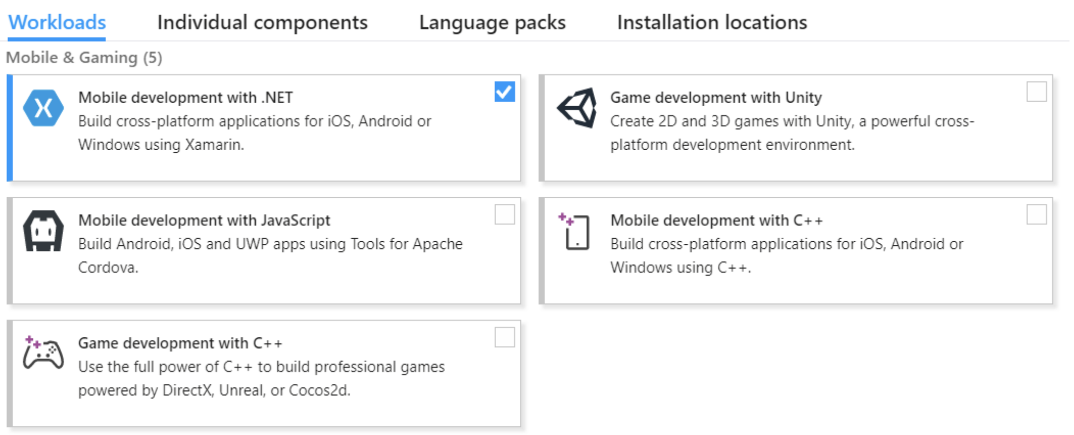
The Raspberry Pi is a single-board computer which supports peripheral devices such as keyboard and mouse. It supports various operating systems including Windows IoT. Windows IoT is designed by Microsoft for embedded systems. It allows an additional task that is testing the developed demonstration application of Xamarin.Essentials on Windows IoT based on Raspberry Pi.

3 Installation and Configuration

In this section, the installation of Visual Studio and the configuration of the Android emulator will be introduced. The way to improve the performance of the emulator will be contained in the configuration part as well.

3.1 Visual Studio Installation

Firstly, Visual Studio IDE can be downloaded from the web page of Microsoft Visual Studio [44]. After running the installation application, the workload which named as “Mobile development with .NET” should be selected to develop a cross-platform application with Xamarin in Figure 3.1.

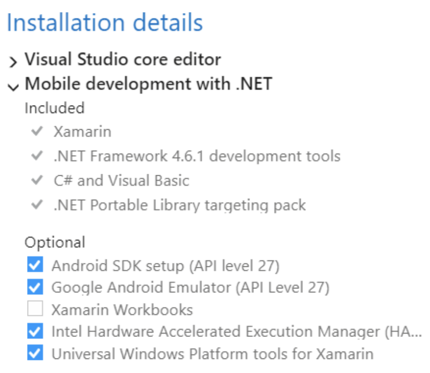


Mobile & Gaming (5)

- [✓] Mobile development with .NET — Build cross-platform applications for iOS, Android or Windows using Xamarin.
- [ ] Game development with Unity — Create 2D and 3D games with Unity, a powerful cross-platform development environment.
- [ ] Mobile development with JavaScript — Build Android, iOS and UWP apps using Tools for Apache Cordova.
- [ ] Mobile development with C++ — Build cross-platform applications for iOS, Android or Windows using C++.
- [ ] Game development with C++ — Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Figure 3.1 Workload selection

As a default, “Universal Windows Platform tools for Xamarin” is a not chosen option in the installation details. Since Xamarin.Essentials should be investigated on UWP as well, this option should be selected to guarantee the access of the build of the test application on UWP (Figure 3.2). After these selections, the installation can be started.



Installation details

> Visual Studio core editor
⌄ Mobile development with .NET

Included
- ✓ Xamarin
- ✓ .NET Framework 4.6.1 development tools
- ✓ C# and Visual Basic
- ✓ .NET Portable Library targeting pack

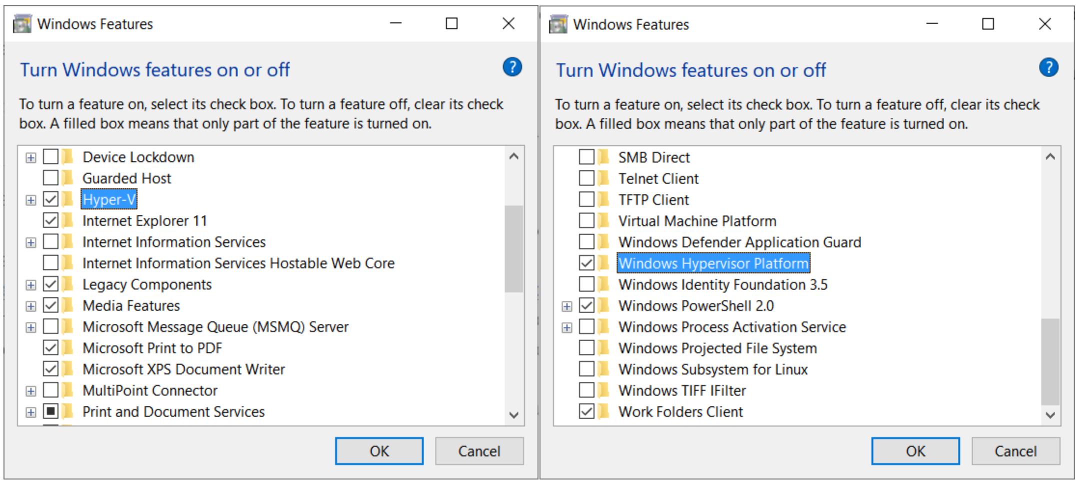
Optional
- [✓] Android SDK setup (API level 27)
- [✓] Google Android Emulator (API Level 27)
- [ ] Xamarin Workbooks
- [✓] Intel Hardware Accelerated Execution Manager (HA...
- [✓] Universal Windows Platform tools for Xamarin

Figure 3.2 Installation details

3.2 Emulator Configuration

Using emulator is a good approach to check written programs before running them on physical devices. Nevertheless, it usually needs great effort for computers to run an emulator. These computers usually seem to be high configuration to ensure the affluence of the running of the emulator. Currently, with the help of the announcement of Windows Hypervisor Platform in the update of Windows on 10th of April in 2018 [47], it is possible to handle Hyper-V as a hardware accelerator for Android emulators. This would improve the performance of the emulators running on low-configuration computers to some extent.

In order to realize this, the Hyper-V and Windows Hypervisor Platform should be turned on in Windows Features interface. Figure 3.3 shows an example of this page. After selected these two features, the computer should be restarted to allow the system to enable them.


Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ☐ Device Lockdown
- ☐ Guarded Host
- ☑ Hyper-V
- ☑ Internet Explorer 11
- ☐ Internet Information Services
- ☐ Internet Information Services Hostable Web Core
- ☑ Legacy Components
- ☑ Media Features
- ☐ Microsoft Message Queue (MSMQ) Server
- ☑ Microsoft Print to PDF
- ☑ Microsoft XPS Document Writer
- ☐ MultiPoint Connector
- ■ Print and Document Services

Windows Features

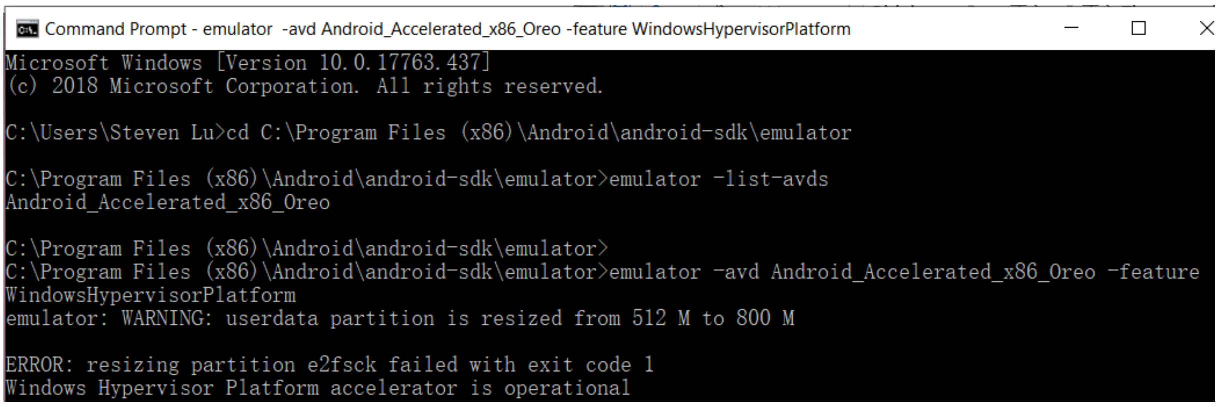
Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ☐ SMB Direct
- ☐ Telnet Client
- ☐ TFTP Client
- ☐ Virtual Machine Platform
- ☐ Windows Defender Application Guard
- ☑ Windows Hypervisor Platform
- ☐ Windows Identity Foundation 3.5
- ☑ Windows PowerShell 2.0
- ☐ Windows Process Activation Service
- ☐ Windows Projected File System
- ☐ Windows Subsystem for Linux
- ☐ Windows TIFF IFilter
- ☑ Work Folders Client

Figure 3.3 Windows Features

Secondly, after clicking Tools > Android > Android SDK managers, the Android SDK Tools and Android Emulators should be updated to the latest version. After that, an emulator flag is required to ensure that the features are used while the emulator is running. This can be realized in two ways. The first one is to create a file as advancedFeatures.ini in the path of C:\Users\<your-username>\.android\. The second one is to start the emulator in the "Command Prompt" with the command "-feature WindowsHypervisorPlatform". The detailed steps are as following. Entering the installation path of the Android emulator is the first step. In my computer, the command should be "cd C:\Program Files (x86)\Android\android-sdk\emulator". Then the name of the Android SDK should be found with the command "emulator -list-avds". Finally, one of the lists should be chosen and the emulator can be started with the command "emulator -avd Android_Accelerated_x86_Oreo -feature WindowsHypervisorPlatform" as an example in Figure 3.4. The emulator started with the latter method will be terminated if the "Command Prompt" window is closed.


Command Prompt - emulator -avd Android_Accelerated_x86_Oreo -feature WindowsHypervisorPlatform

```
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Steven Lu>cd C:\Program Files (x86)\Android\android-sdk\emulator

C:\Program Files (x86)\Android\android-sdk\emulator>emulator -list-avds
Android_Accelerated_x86_Oreo

C:\Program Files (x86)\Android\android-sdk\emulator>
C:\Program Files (x86)\Android\android-sdk\emulator>emulator -avd Android_Accelerated_x86_Oreo -feature
WindowsHypervisorPlatform
emulator: WARNING: userdata partition is resized from 512 M to 800 M

ERROR: resizing partition e2fsck failed with exit code 1
Windows Hypervisor Platform accelerator is operational
```

Figure 3.4 Command Prompt View

In my personal computer, the Android emulator needs 24 seconds to be started without Hyper-V, while only 13 seconds with the help of the hardware acceleration. The affluence while running applications in the emulator is apparently improved as well.

3.3 Physical Devices Prepared

A smartphone “Xiaomi 5” and a tablet “Samsung SM-T585” are used for the test on the Android platform. A device called “iPhone 6” is prepared for the test on the iOS platform. A device called “Surface” is prepared for the test on the UWP platform. Additionally, Raspberry Pi with the system “Windows IoT” is also used to investigate the availability of Xamarin.Essentials on other platforms.

4 Features and additional tasks

This section will show the first half features of Xamarin.Essentials and implementation processes and behaviors on different platforms of these features. An additional task will be included as well. The rest of the features is contained in another thesis written by Mr. Wang [50]. To provide a better specification, a demonstration mobile application called "EssentialsDemo" is developed. Since the purpose is to provide a better understanding of the features, the application is designed with "Master-Detail Page" in Xamarin.Forms. In this presentation form, a list of features shows on the left side of the screen, and the detail of each feature shows on the right side. A slight performance difference of the "Master-Detail Page" among three platforms is shown in Figure 4.1 Steps of the implementation of the "Master-Detail Page" can be found in the official document [9].

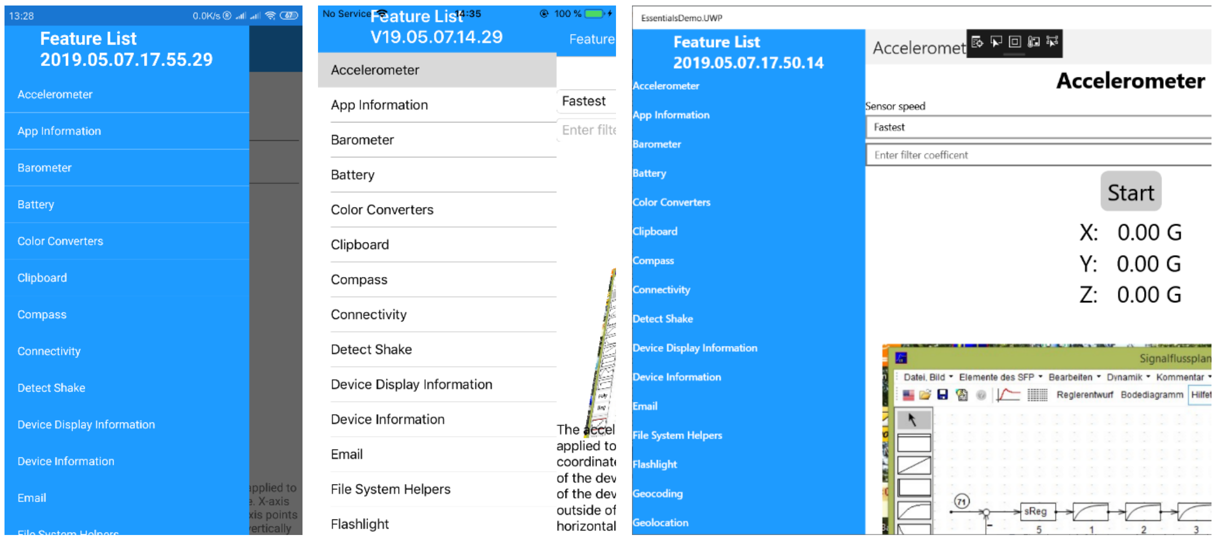


Figure 4.1 Master-Detail Page on Android, iOS & UWP

In this application, no MVVM (Model-View-ViewModel) pattern is used, which means the user interface, presentation logic and business logic are written in one file. This pattern is a variation of MVC (Model-View-Controller) pattern. User interfaces are generally determined in Views, and ViewModels determine the presentation logic of Views. Other logic, such as business logic, is determined in Models. Models also provide model data support [46]. Each feature has its own demonstration class "ContentPage". It is probably crucial for large projects to clarify them, but for this test project, there is no need to adapt this pattern to each feature demonstration, which would increase duplication of codes.

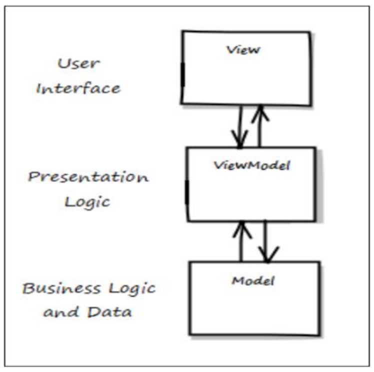
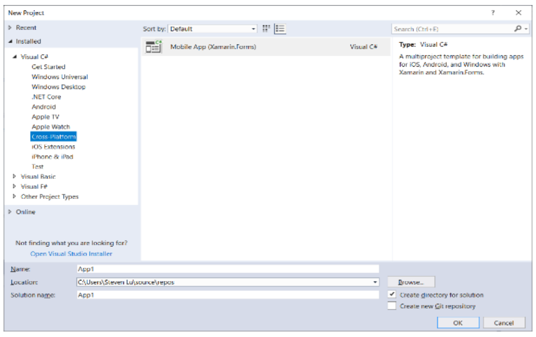


Figure 4.2 MVVM pattern [46]

Before using these features, one important step is to install Xamarin.Essentials NuGet package to projects of Android, iOS and UWP [10]. Although the behavior of these features on the Android platform is primarily considered in this thesis, it is necessary for those who want to develop a cross-platform application to add the package to all platform projects. Otherwise, the application will not run on a specific platform without the package. The detail of the installation is below. First, create a mobile application project as shown in Figure 4.3. Next, right click on the solution and select "Manage NuGet Packages for Solution". Finally, search for the package named "Xamarin.Essentials" and add it to all projects as shown in Figure 4.4.



Mobile App (Xamarin.Forms) Visual C#

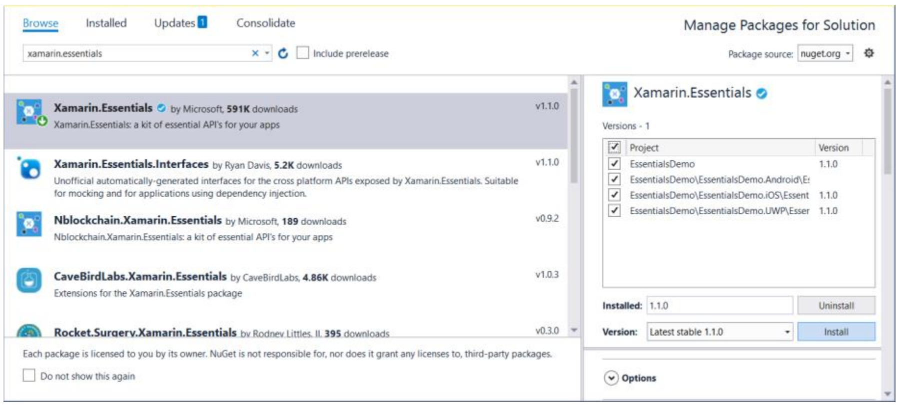
Type: Visual C#

A multiproject template for building apps for iOS, Android, and Windows with Xamarin and Xamarin.Forms.

Not finding what you are looking for?
Open Visual Studio Installer

Name: App1
Location: C:\Users\Steven Lu\source\repos
Solution name: App1
[✓] Create directory for solution
[ ] Create new Git repository

Figure 4.3 Create Mobile Apps Project



xamarin.essentials

Package source: nuget.org

Xamarin.Essentials by Microsoft, **591K** downloads v1.1.0
Xamarin.Essentials: a kit of essential API's for your apps

Xamarin.Essentials.Interfaces by Ryan Davis, **5.2K** downloads v1.1.0
Unofficial automatically-generated interfaces for the cross platform APIs exposed by Xamarin.Essentials. Suitable for mocking and for applications using dependency injection.

Nblockchain.Xamarin.Essentials by Microsoft, **189** downloads v0.9.2
Nblockchain.Xamarin.Essentials: a kit of essential API's for your apps

CaveBirdLabs.Xamarin.Essentials by CaveBirdLabs, **4.86K** downloads v1.0.3
Extensions for the Xamarin.Essentials package

Rocket.Surgery.Xamarin.Essentials by Rodney Littles, II, **395** downloads v0.3.0

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

[ ] Do not show this again

Xamarin.Essentials

Versions - 1

	Project	Version
✓	EssentialsDemo	1.1.0
✓	EssentialsDemo\EssentialsDemo.Android\Es	
✓	EssentialsDemo\EssentialsDemo.iOS\Essent	1.1.0
✓	EssentialsDemo\EssentialsDemo.UWP\Esser	1.1.0

Installed: 1.1.0
Version: Latest stable 1.1.0

Figure 4.4 Install Xamarin.Essentials

On Android platform, it is necessary to initialize all activities in the method “OnCreate” in the file “MainActivity.cs” and to handle the permissions by adding method “OnRequestPermissionsResult” to the file as shown in Listing 4.1.

```
protected override void OnCreate(Bundle savedInstanceState)
{
    base.OnCreate(savedInstanceState);
    Xamarin.Essentials.Platform.Init(this, savedInstanceState);
}
public override void OnRequestPermissionsResult(int requestCode, string[] pe
rmissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
{
    Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, perm
issions, grantResults);
    base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

Listing 4.1 Codes added in the “MainActivity.cs”

4.1 Accelerometer

An accelerometer is a hardware-based motion sensor on the Android platform [2]. It directly measures acceleration force on the three-dimensional coordinate axes with the physical elements on the device. There is one property “IsMonitoring”, two methods “Start” and “Stop” and two events “ReadingChanged” and “ShakeDetected” in the class “Accelerometer”. The event “ShakeDetected” will be described later in Section 4.9.

The speed of the sensor should be determined first. There are four kinds of speed which are “Default”, “UI”, “Game” and “Fastest”. The measurement frequency of them is from low to high as shown in Table 4.1. The frequency means that the sensor detects changes at a speed of how many times per second.

Type of Speed	Default	UI	Game	Fastest
Frequency (s^{-1})	5	18	53	205

Table 4.1 Frequency of the sensor speed

The data are measured with codes in Listing 4.2 on the author’s personal device. However, the frequency could be influenced by the performance of the device. Namely, there might be a difference between various devices with different platforms.

```
void Accelerometer_ReadingChanged(object sender, AccelerometerChangedEventArgs e) {
    count++; // Sum when there is new reading
}
if (Accelerometer.IsMonitoring) {
    Accelerometer.Stop();
    stopWatch.Stop();
    TimeSpan timeSpan = stopWatch.Elapsed; // Duration of measurement
    label.Text = Math.Round(timeSpan.TotalSeconds, 2) + "; " + count + "; "
        + Math.Round((count / Math.Round(timeSpan.TotalSeconds, 2)), 0);
    stopWatch = new Stopwatch();
    count = 0; // Counters for frequency measurement
} else {
    Accelerometer.Start(speed);
    stopWatch.Start();
}
```

Listing 4.2 Frequency measurement

The “ReadingChanged” event should be registered in the constructor of the class. To better present the feature, a rotation picture is used instead of three data on three coordinate axes. In this coordinate, the direction of the X-axis is horizontally towards the right side of the device. The direction of the Y-axis is vertically towards the top of the device. The direction of Z-axis is vertical to the X-axis and Y-axis, towards outside of the device (Figure 4.5).

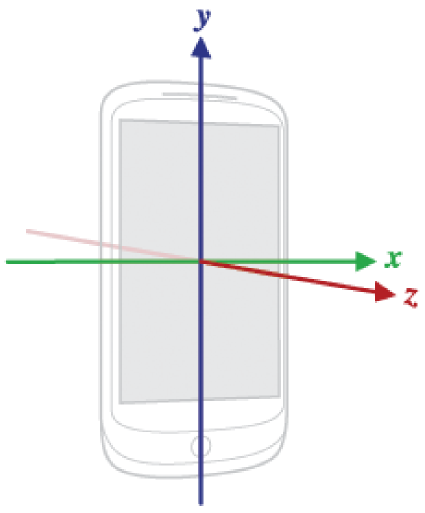


Figure 4.5 Device coordinate [3]

According to the acceleration of the phone, the picture will always be horizontal to the earth, which can show the gravity orientation (Figure 4.6). The basic implementation of sensor speed definition, event registration and utilization of “IsMonitoring”, “Start” and “Stop” methods are shown in Listing 4.3. The presentation logic of the picture is also included from the 11th to the 18th line in Listing 4.3. It uses the values from the event “ReadingChanged” to calculate the rotation angle of the image, which makes the picture seems to be always horizontal to the ground. The calculation is based on the inner product principle of Euclidean space. If the speed of the sensor is set to “Game” or “Fastest”, class “MainThread” should be used to avoid the event handler not to return on UI thread. In other words, the manipulation on the UI controller might be ignored [24]. The detailed implementation of “MainThread” can be seen in the other thesis from Mr. Wang [50].

```
SensorSpeed speed = SensorSpeed.Fastest; // Set speed delay for monitoring changes.
public AccelerometerTest() {
    // Register for reading changes
    Accelerometer.ReadingChanged += Accelerometer_ReadingChanged;
}
void Accelerometer_ReadingChanged(object sender, AccelerometerChangedEventArgs e) {
    // Process Acceleration X, Y, and Z
    var x = e.Reading.Acceleration.X;
    var y = e.Reading.Acceleration.Y;
    var z = e.Reading.Acceleration.Z;
    if (y >= 0)
    image.RotationX = Math.Acos(z / Math.Sqrt(y * y + z * z)) * 180 / Math.PI;
    else
        image.RotationX = 360 - Math.Acos(z / Math.Sqrt(y * y + z * z)) * 180 / Math.PI;
   if (x >= 0)
       image.RotationY = Math.Acos(z / Math.Sqrt(x * x + z * z)) * 180 / Math.PI;
    else
        image.RotationY = 360 - Math.Acos(z / Math.Sqrt(x * x + z * z)) * 180 / Math.PI;
}
// Switch on / off the sensor
public void ToggleAccelerometer() {
    if (Accelerometer.IsMonitoring)
       Accelerometer.Stop();
   else
        Accelerometer.Start(speed);
}
```

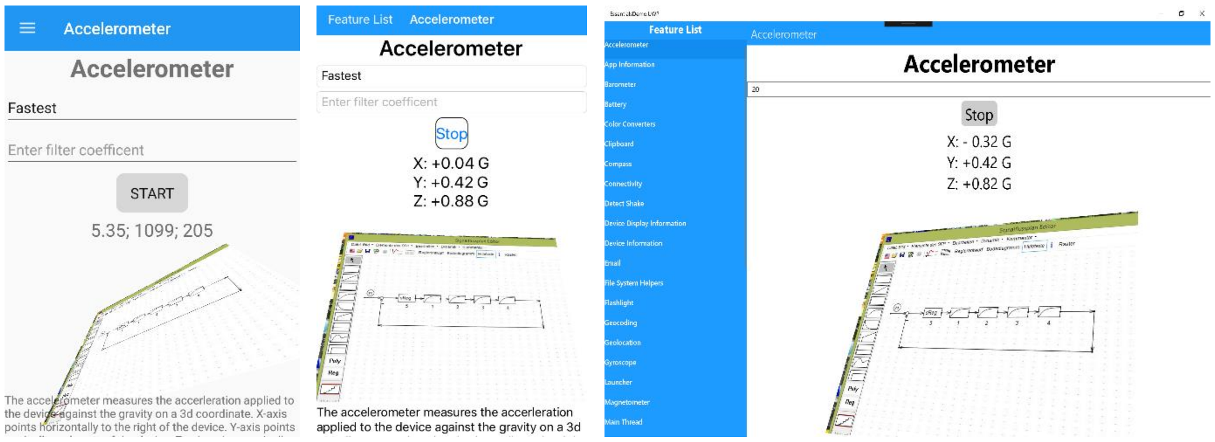
Listing 4.3 Basic implementation of Accelerometer [24]

The reading data is measured with unit G, which equals to 9.81 m/s^2, the gravity from the earth. For instance, when the device lies stably on a table, the value of X-axis and Y-axis should be zero, and the value of Z-axis should be a positive one. This indicates that the accelerometer measures the acceleration force against the earth gravity. When using the data from the accelerometer, a big noise can be found through the data. To eliminate the noise, one of the low-pass filters is used as shown in Listing 4.4. The new value generated from this filter is the average of ten previous values. A list is used to manipulate the number of previous values. There will be a delay in the output value if the amount is set too large. The amount can be could be changed with the entry as shown in Figure 4.6.

```
List<float> list = new List<float>();
int N = 10;
while (list.Count > N - 1) {
    list.RemoveAt(0);
}
list.Add(data);
data = filter(data);
float filter(List<float> list) {
    float sum = 0;
    foreach (float element in list) {
        sum += element;
    }
    return sum / list.Count;
}
```

Listing 4.4 Arithmetic average low-pass filter

The performance of the accelerometer on the Android, iOS and UWP platform are shown in order in Figure 4.6. As mentioned before, the rotation of the picture indicates the orientation of gravity after the accelerometer is started. Meanwhile, the acceleration force on the three axes is shown on the screen. After the accelerometer stops measuring, the measurement period, the times of the event "ReadingChange" triggered and the frequency are shown in order in a string. There also exists a picker to select different sensor speed. It is necessary to restart the accelerometer after changing the speed.



Accelerometer

Fastest

Enter filter coefficent

5.35; 1099; 205

The accelerometer measures the accerleration applied to the device against the gravity on a 3d coordinate. X-axis points horizontally to the right of the device. Y-axis points

Accelerometer

Fastest

Enter filter coefficent

X: +0.04 G
Y: +0.42 G
Z: +0.88 G

The accelerometer measures the accerleration applied to the device against the gravity on a 3d

Accelerometer

20

X: - 0.32 G
Y: +0.42 G
Z: +0.82 G

Figure 4.6 Performance of Accelerometer on Android, iOS & UWP

This feature gives no response instead of an error message after clicking the button on Windows IoT base on Raspberry Pi.

4.2 Application information

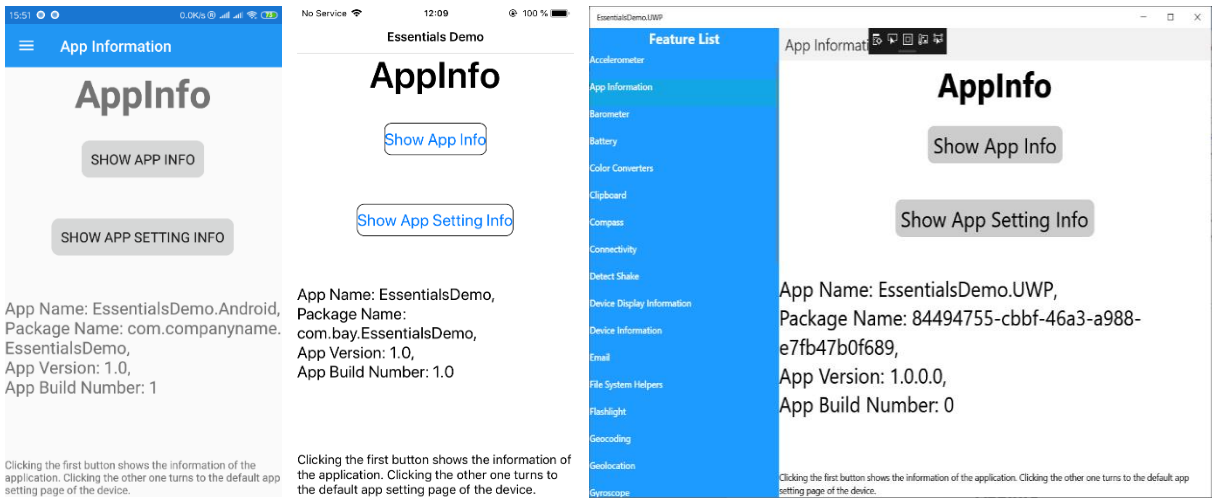
There are five properties which are “BuildString”, “Name”, “PackageName”, “Version” and “VersionString” in the class “AppInfo”. For the Android platform, these properties are retrieved from nodes in the file “AndroidManifest.xml” [25]. The only method contained in the class is “ShowSettingsUI”. The properties can be simply accessed as shown in Listing 4.5. The default setting page of the platform will be shown with the method. Some permissions of the application can be given on this page.

```
var appName = AppInfo.Name; // Application Name
var packageName = AppInfo.PackageName; // Package Name/Application Identifier
var version = AppInfo.VersionString; // Application Version
var build = AppInfo.BuildString; // Application Build Number
AppInfo.ShowSettingsUI(); // Display settings page
```

Listing 4.5 AppInfo implementation [25]

For the Android platform, the content of the five properties can be changed manually in the file “AndroidManifest.xml”. Each property is related to an attribute of a node in the file. For the iOS platform, the content of the five properties can be changed in the file “Info.plist”. The management file of UWP is “Package.appxmanifest”.

The performance of this feature on Android, iOS and UWP are shown in order in Figure 4.7. The default package name and app name on these platforms are different. The default setting page of the application will be shown after clicking the second button. This feature is valid on Windows IoT based on Raspberry Pi as well and the performance is the same as that on UWP. The probable reason is that they are both Windows platforms.



App Name: EssentialsDemo.Android,
Package Name: com.companyname.EssentialsDemo,
App Version: 1.0,
App Build Number: 1

Clicking the first button shows the information of the application. Clicking the other one turns to the default app setting page of the device.

App Name: EssentialsDemo,
Package Name:
com.bay.EssentialsDemo,
App Version: 1.0,
App Build Number: 1.0

Clicking the first button shows the information of the application. Clicking the other one turns to the default app setting page of the device.

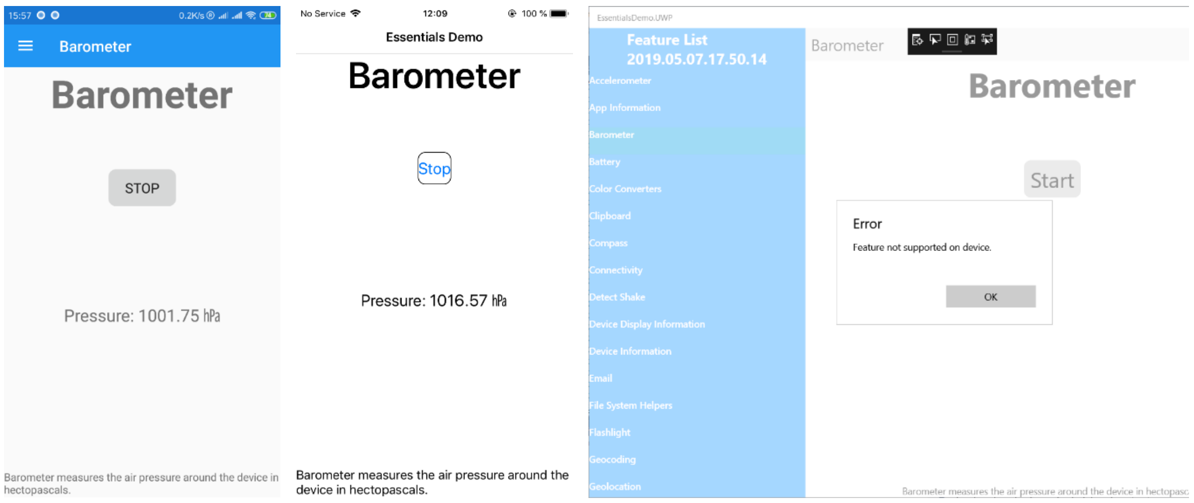
App Name: EssentialsDemo.UWP,
Package Name: 84494755-cbbf-46a3-a988-e7fb47b0f689,
App Version: 1.0.0.0,
App Build Number: 0

Clicking the first button shows the information of the application. Clicking the other one turns to the default app setting page of the device.

Figure 4.7 Performance of Application Information on Android, iOS & UWP

4.3 Barometer

Barometer belongs to the category of environmental sensors, which is based on hardware [3]. Namely, a barometer measures the environment pressure directly through a physical segment in the device. Similar to the accelerometer, there exists one property “IsMonitoring”, two methods “Start” and “Stop” and one event “ReadingChanged” in the class “Barometer”. The speed of the sensor and the registration of reading changes can be set as described in Section 4.1. The utilization of the property and two methods can be referred to the official document [13]. Since the sensor measures the air pressure around the device, there might be a difference between the pressure in the weather forecast and the data. It could also be influenced by the accurateness of the physical segment in the device.



Pressure: 1001.75 hPa

Barometer measures the air pressure around the device in hectopascals.

Pressure: 1016.57 hPa

Barometer measures the air pressure around the device in hectopascals.

Error

Feature not supported on device.

Barometer measures the air pressure around the device in hectopasc

Figure 4.8 Performance of Barometer on Android, iOS & UWP

The performance of this feature on Android, iOS and UWP are shown in order in Figure 4.8. Unfortunately, the author has not a device with the UWP platform which can validate this feature. In addition, this feature is not valid on Windows IoT based on Raspberry Pi. It will throw a “FeatureNotSupported” exception similar to that on UWP in Figure 4.8.

4.4 Battery

Before implementing the feature, the permission to get access to the information on the battery should be added manually. The code in Listing 4.6 should be added in the file “AssemblyInfo.cs” under the “Properties” in the Android project. Another way to add permission is to add the code in Listing 4.7 into the file “AndroidManifest.xml” under the “Properties”.

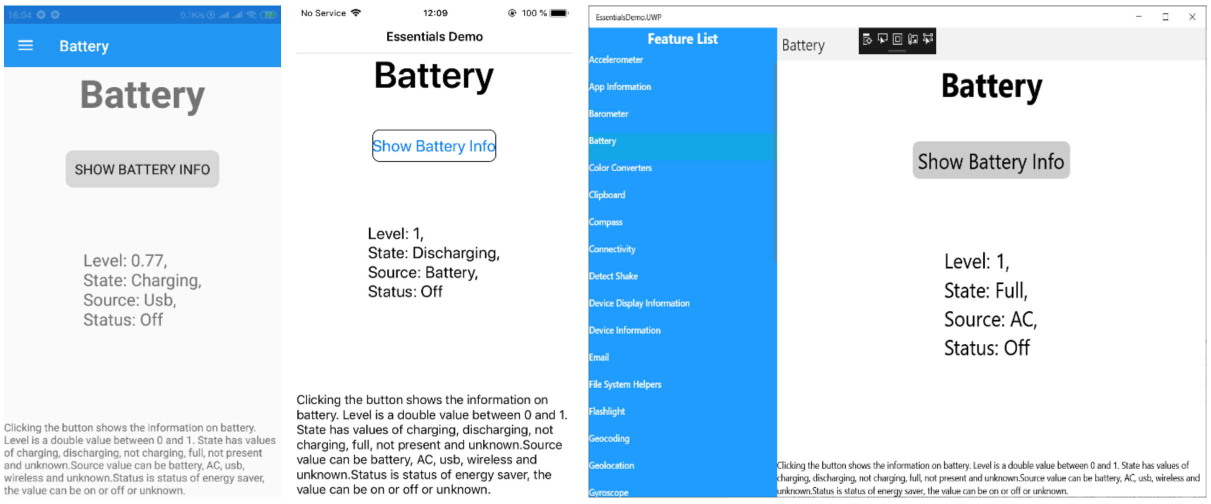
```
[assembly: UsesPermission(Android.Manifest.Permission.BatteryStats)]
```

Listing 4.6 Battery permission in C# [26]

```xml
<uses-permission android:name="android.permission.BATTERY_STATS" />
```

Listing 4.7 Battery permission in XML [26]

There exist four properties which are “ChargeLevel”, “EnergySaverStatus”, “PowerSource” and “State” in the class “Battery”. Two events which are “BatteryInfoChanged” and “EnergySaverStatusChanged” are also included in this class. The property “ChargeLevel” can return a double value between 0.0 and 1.0. “EnergySaverStatus” has enumeration values whose fields are “On”, “Off” and “Unknown”. “PowerSource” has enumeration values whose fields are “AC”, “Battery”, “Unknown”, “Usb” and “Wireless”. “State” has enumeration values whose fields are “Charging”, “Discharging”, “Full”, “Notcharging”, “Notpresent” and “Unknown” [30]. “Notcharging” means a state which is neither charging nor discharging. According to the official document [26], “PowerSource” can only return values of “AC” or “Battery” on the iOS and UWP platform. These properties can be set as described in Section 4.2 or can be implemented as shown in the official document [26]. The registration of two events is also the same as mentioned before in Section 4.1.



Level: 0.77,
State: Charging,
Source: Usb,
Status: Off

Clicking the button shows the information on battery. Level is a double value between 0 and 1. State has values of charging, discharging, not charging, full, not present and unknown.Source value can be battery, AC, usb, wireless and unknown.Status is status of energy saver, the value can be on or off or unknown.

Level: 1,
State: Discharging,
Source: Battery,
Status: Off

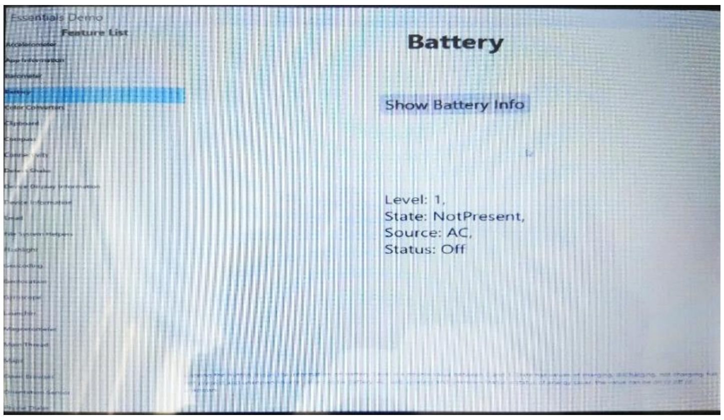
Clicking the button shows the information on battery. Level is a double value between 0 and 1. State has values of charging, discharging, not charging, full, not present and unknown.Source value can be battery, AC, usb, wireless and unknown.Status is status of energy saver, the value can be on or off or unknown.

Level: 1,
State: Full,
Source: AC,
Status: Off

Clicking the button shows the information on battery. Level is a double value between 0 and 1. State has values of charging, discharging, not charging, full, not present and unknown.Source value can be battery, AC, usb, wireless and unknown.Status is status of energy saver, the value can be on or off or unknown.

Figure 4.9 Performance of Battery on Android, iOS & UWP

The performance of this feature on Android, iOS and UWP are shown in order in Figure 4.9. The information of the battery is shown by clicking the button. It will update automatically if the state of the battery is changed. The performance on Windows IoT based on Raspberry Pi is shown in Figure 4.10. The state is “NotPresent”, which means that there is no battery in the device.



Level: 1,
State: NotPresent,
Source: AC,
Status: Off

Figure 4.10 Performance of Battery on Windows IoT

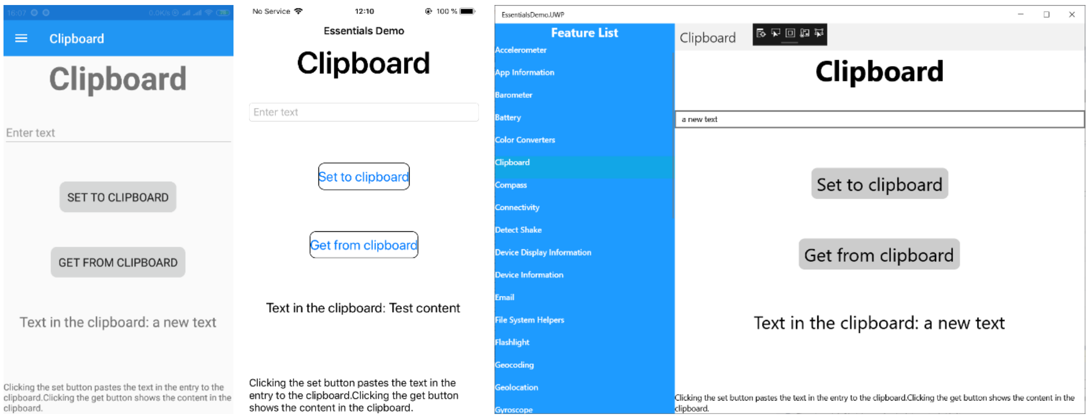
4.5 Clipboard

There exists only one property “HasText” and two methods called “GetTextAsync” and “SetTextAsync” in the class “Clipboard” [31]. The property has a boolean value which shows whether there is text in the clipboard or not. These two methods can be simply implemented as shown in Listing 4.8.

```
// Set text in the entry to the clipboard
await Clipboard.SetTextAsync(entry.Text);
// Get text from the clipboard
label.Text = await ShowClipboardTextAsync();
```

Listing 4.8 Implementation of Clipboard

To better demonstrate the behavior, an entry field is provided in the clipboard page. Any text in the entry field can be set to the clipboard with the button “Set”. Button “Get” can show the text in the clipboard with a label. The performance of this feature on Android, iOS and UWP are shown in order in Figure 4.11.



Text in the clipboard: a new text

Clicking the set button pastes the text in the entry to the clipboard.Clicking the get button shows the content in the clipboard.

Text in the clipboard: Test content

Clicking the set button pastes the text in the entry to the clipboard.Clicking the get button shows the content in the clipboard.

Text in the clipboard: a new text

Clicking the set button pastes the text in the entry to the clipboard.Clicking the get button shows the content in the clipboard.

Figure 4.11 Performance of Clipboard on Android, iOS & UWP

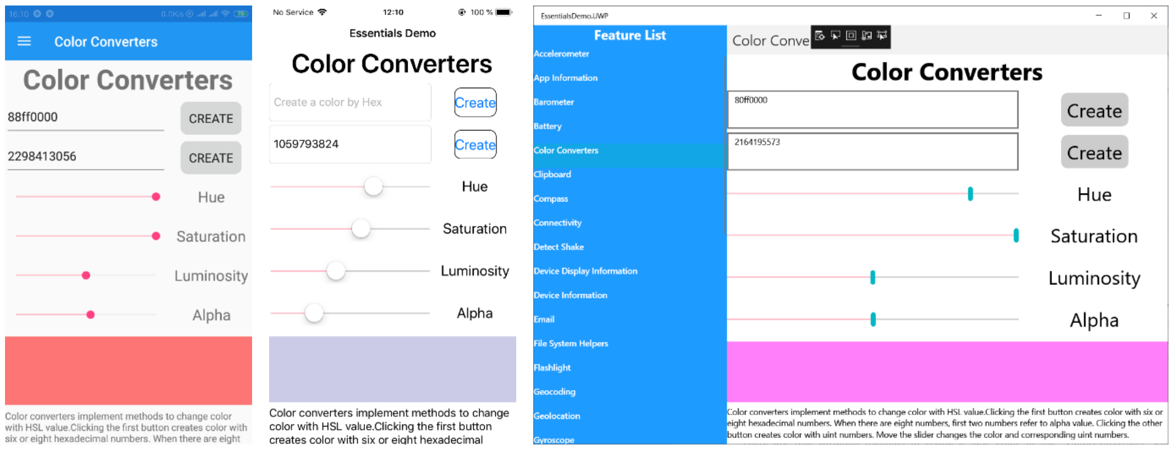
4.6 Color Converters

There are four different methods to create a new color in the "ColorConverters" class [32]. The color can be created with a hexadecimal string or with HSLA (hue, saturation, luminosity and alpha) values or with a UInt value. The hexadecimal string consists of six numbers. Each pair of the six numbers respectively refer to RGB (red, green and blue) values. The values can be eight numbers as well. In this condition, the first pair of the eight numbers should be the alpha value. To supplement the color converters, "ColorExtensions" class provide more methods to control the color. For instance, it can create a new color based on the original color with a new value of alpha, hue, saturation or luminosity [33]. The basic implementation of these two classes is shown in Listing 4.9.

```csharp
box.Color = ColorConverters.FromHex(entry_hex.Text);
box.Color = ColorConverters.FromUInt(uint.Parse(entry_uint.Text));
box.Color = ColorExtensions.WithHue(box.Color, (float)slider1.Value);
box.Color = ColorExtensions.WithSaturation(box.Color, (float)slider2.Value);
box.Color = ColorExtensions.WithLuminosity(box.Color, (float)slider3.Value);
box.Color = ColorExtensions.WithAlpha(box.Color, (int)slider4.Value);
entry_uint.Text = ColorExtensions.ToUInt(box.Color).ToString();
```

Listing 4.9 Implementation of Color Converters & Color Extensions

To better present this feature, a color box with four slider controllers is set in the page. New colors can be created in the box by clicking the "Create" button with hexadecimal value or UInt value in the entries. In addition, the four sliders can change the HSLA values of the color in the box with the methods mentioned before. To give a direct view of UInt value, the value in the second entry will change if the color in the color box changes. The performance of this feature on Android, iOS and UWP are shown in order in Figure 4.12. This feature is also valid on Windows IoT based on Raspberry Pi as shown in Figure 4.13.



Color converters implement methods to change color with HSL value.Clicking the first button creates color with six or eight hexadecimal numbers. When there are eight

Color converters implement methods to change color with HSL value.Clicking the first button creates color with six or eight hexadecimal

Color converters implement methods to change color with HSL value.Clicking the first button creates color with six or eight hexadecimal numbers. When there are eight numbers, first two numbers refer to alpha value. Clicking the other button creates color with uint numbers. Move the slider changes the color and corresponding uint numbers.

Figure 4.12 Performance of Color Converters & Color Extensions on Android, iOS & UWP

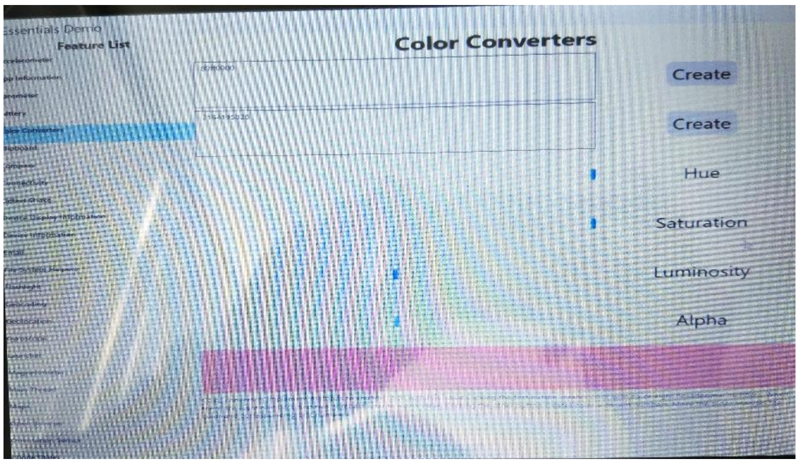


Figure 4.13 Performance of Color Converters & Color Extensions on Windows IoT

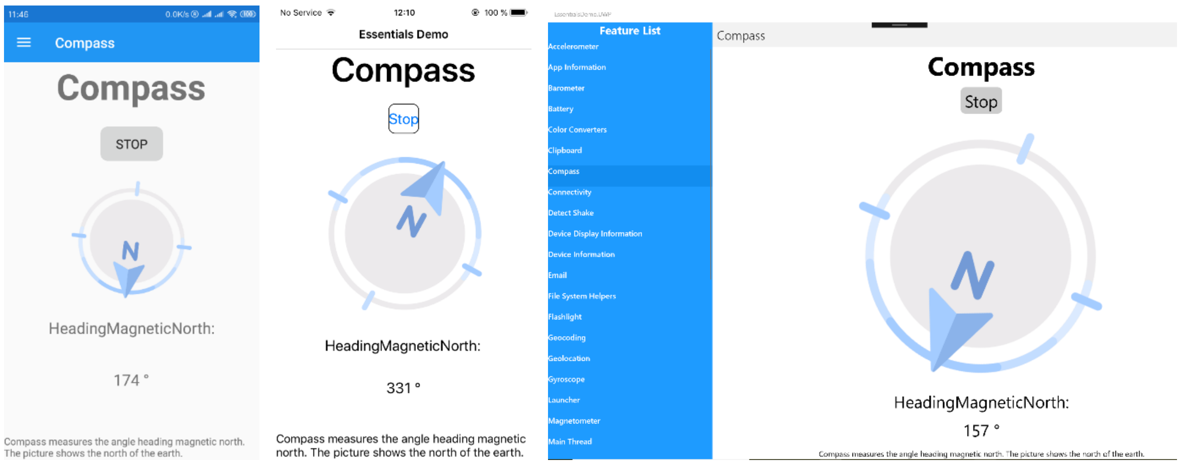
4.7 Compass

According to the official document of Android [14], there is no definition of API (application programming interface) to get the heading of compass for Android. The value of heading magnetic north is measured by exploiting accelerometer and magnetometer. Thus, compass is a software-based sensor. There exists one property "IsMonitoring", two "Start" methods, one "Stop" method and one "ReadingChanged" Event in the "Compass" class [34]. The distinction between two "Start" methods is whether to use a low-pass filter or not. However, this low-pass filter is only suitable for Android devices because compasses on the other two platforms can be accessed directly instead of with the calculation from accelerometer and magnetometer. The speed of the sensor and the registration of "ReadingChanged" can be set as described in Section 4.1. To better present the feature, a picture of north arrow rotates with the angle of heading magnetic north. The implementation of the rotation logic of the picture is shown in Listing 4.10. The 3rd line in Listing 4.10 is used to avoid event handler not to return on UI thread as mentioned in Section 4.1.

```
1.  void Compass_ReadingChanged(object sender, CompassChangedEventArgs e) {
2.      //To avoid not able to return on UI thread
3.      MainThread.BeginInvokeOnMainThread(() => {
4.      label.Text = String.Format("{0,0:000} °", e.Reading.HeadingMagneticNorth);
5.      image.Rotation = -e.Reading.HeadingMagneticNorth;
6.      });
7.  }
```

Listing 4.10 Implementation of Compass

The performance of this feature on Android, iOS and UWP are shown in order in Figure 4.14. The button is used to start or stop the running of the feature. This feature throws a "FeatureNotSupported" exception on Windows IoT based on Raspberry Pi.



Compass measures the angle heading magnetic north. The picture shows the north of the earth.

Figure 4.14 Performance of Compass on Android, iOS & UWP

4.8 Connectivity

Before implementing the feature, the permission to get access to the information on the connectivity should be added manually [27]. This step is only required on the Android platform. The code in Listing 4.11 should be added in the file “AssemblyInfo.cs” under the “Properties” in the Android project. Another way to add the permission is to add code in Listing 4.12 into the file “AndroidManifest.xml” under the “Properties”.

```
[assembly: UsesPermission(Android.Manifest.Permission.AccessNetworkState)]
```

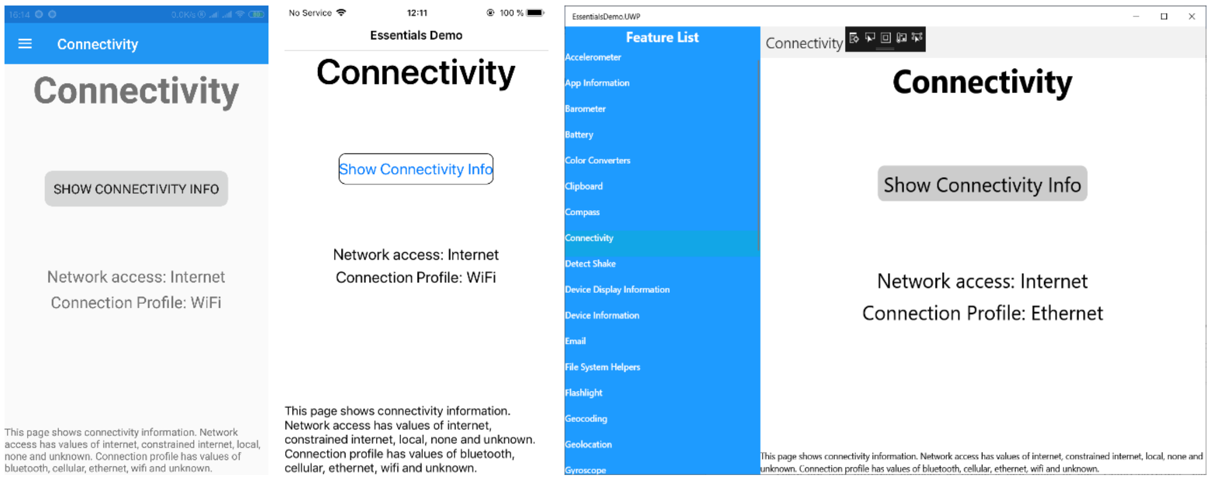
Listing 4.11 Connectivity permission in C# [27]

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Listing 4.12 Connectivity permission in XML [27]

There exists two properties “ConnectionProfiles” and “NetworkAccess” and one event “ConnectivityChanged” in the “Connectivity” class. In the former property, the fields contain “Bluetooth”, “Cellular”, “Ethernet”, “Wifi” and “Unknown”. In the other property, the fields contain “Internet”, “ConstrainedInternet”, “Local”, “None” and “Unknown”. The implementation of this feature can be referred to the official document [27].

The performance of this feature on Android, iOS and UWP are shown in order in Figure 4.15. The information will be shown after clicking the button. This feature is also valid on Windows IoT based on Raspberry Pi, whose display of network access and connection profile is the same as that on UWP.



Network access: Internet
Connection Profile: WiFi

This page shows connectivity information. Network access has values of internet, constrained internet, local, none and unknown. Connection profile has values of bluetooth, cellular, ethernet, wifi and unknown.

Network access: Internet
Connection Profile: WiFi

This page shows connectivity information. Network access has values of internet, constrained internet, local, none and unknown. Connection profile has values of bluetooth, cellular, ethernet, wifi and unknown.

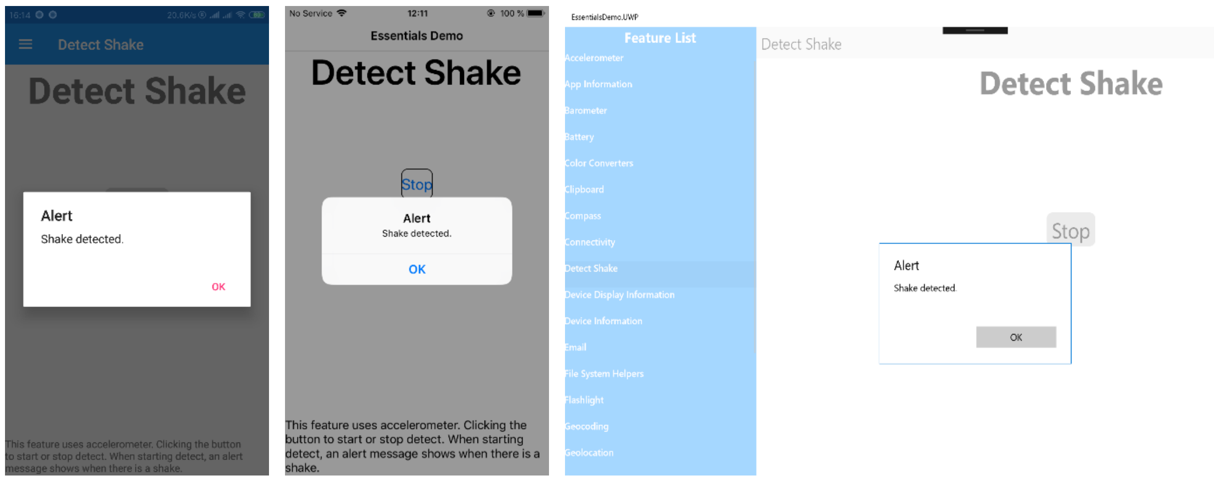
Network access: Internet
Connection Profile: Ethernet

This page shows connectivity information. Network access has values of internet, constrained internet, local, none and unknown. Connection profile has values of bluetooth, cellular, ethernet, wifi and unknown.

Figure 4.15 Performance of Connectivity on Android, iOS & UWP

4.9 Detect Shake

This feature utilizes the event "ShakeDetected" in the class "Accelerometer" to detect a shake of the device. The shake is detected with a queue mechanism which examines if there exists over 75 percent of recent events during the latest 0.5 seconds [15]. Furthermore, according to the source code [5], a threshold of 169 is set to compare with the sum of the square of the acceleration value on X, Y and Z axis. The speed of the sensor, the registration of "ReadingChanged" and the basic implementation of the event can be completed as described in Section 4.1.



Alert
Shake detected.

This feature uses accelerometer. Clicking the button to start or stop detect. When starting detect, an alert message shows when there is a shake.

Figure 4.16 Performance of Detect Shake on Android, iOS & UWP

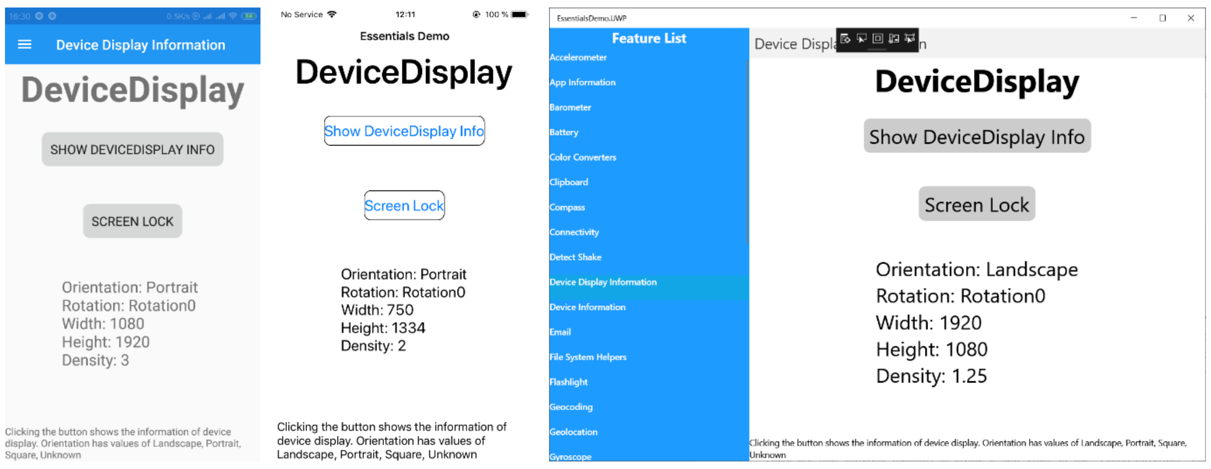
In this content page, a button is used to control the sensor. An alert message will display if a shake is detected after the sensor is started. The performance of this feature on Android, iOS and UWP are shown in order in Figure 4.16. This feature gives no response instead of an error message after clicking the button on Windows IoT base on Raspberry Pi.

4.10 Device Display Information

There exist two properties “MainDisplayInfo” and “KeepScreenOn” and one event “MainDisplayInfoChanged” in the class “DeviceDisplay” [35]. In the “MainDisplayInfo”, there are properties “Orientation”, “Rotation”, “Width”, “Height” and “Density”. The property “Orientation” has values of “Landscape”, “Portrait”, “Square”, and “Unknown”. The property “Rotation” has values of 0, 90, 180 and 270. The “DisplayInfo” of the event can be read only after “MainDisplayInfoChanged” is registered. “KeepScreenOn” can be set true to keep the screen of the device from switching off [16]. The registration and implementation of “MainDisplayInfoChanged” can be completed as described in Section 4.1. The basic implementation of this class is shown in Listing 4.13. According to the official document [16], it should be implemented on the UI thread on the iOS platform to avoid exceptions.

```
var mainDisplayInfo = DeviceDisplay.MainDisplayInfo;
var orientation = mainDisplayInfo.Orientation
var rotation = mainDisplayInfo.Rotation;
var width = mainDisplayInfo.Width; // Width (in pixels)
var height = mainDisplayInfo.Height; // Height (in pixels)
var density = mainDisplayInfo.Density; // Screen density
DeviceDisplay.KeepScreenOn = !DeviceDisplay.KeepScreenOn; // Keep Screen on or not
```

Listing 4.13 Implementation of Device Display Information [16]



Orientation: Portrait
Rotation: Rotation0
Width: 1080
Height: 1920
Density: 3

Clicking the button shows the information of device display. Orientation has values of Landscape, Portrait, Square, Unknown

Orientation: Portrait
Rotation: Rotation0
Width: 750
Height: 1334
Density: 2

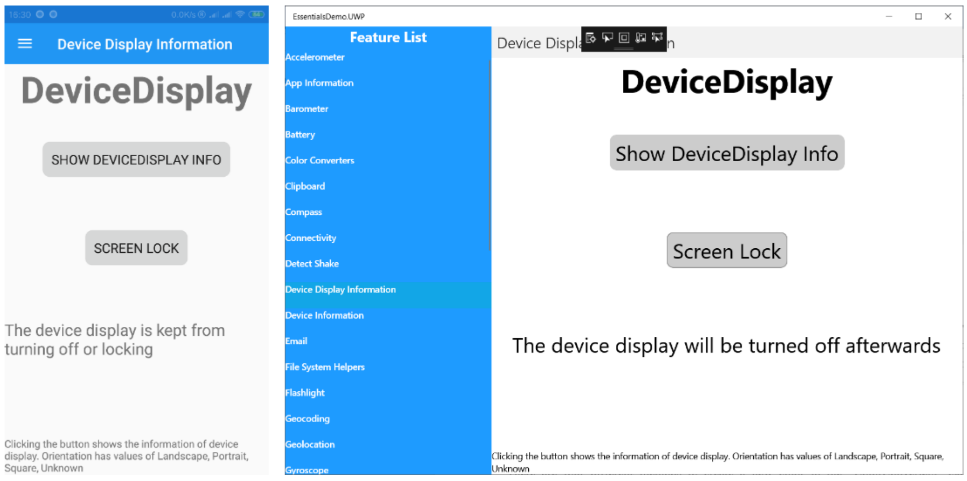
Clicking the button shows the information of device display. Orientation has values of Landscape, Portrait, Square, Unknown

Orientation: Landscape
Rotation: Rotation0
Width: 1920
Height: 1080
Density: 1.25

Clicking the button shows the information of device display. Orientation has values of Landscape, Portrait, Square, Unknown

Figure 4.17 Performance of Device Display Information on Android, iOS & UWP - 1

In this page, the first button is used to display information. The performance of clicking the first button on Android, iOS and UWP are shown in order in Figure 4.17. The other button is used to keep the screen from switching off. A hint is displayed to tell whether the screen will be shut down afterward or not (Figure 4.18). The feature is supported on Windows IoT based on Raspberry Pi.



The device display is kept from turning off or locking

Clicking the button shows the information of device display. Orientation has values of Landscape, Portrait, Square, Unknown

The device display will be turned off afterwards

Clicking the button shows the information of device display. Orientation has values of Landscape, Portrait, Square, Unknown

Figure 4.18 Performance of Device Display Information on Android, iOS & UWP - 2

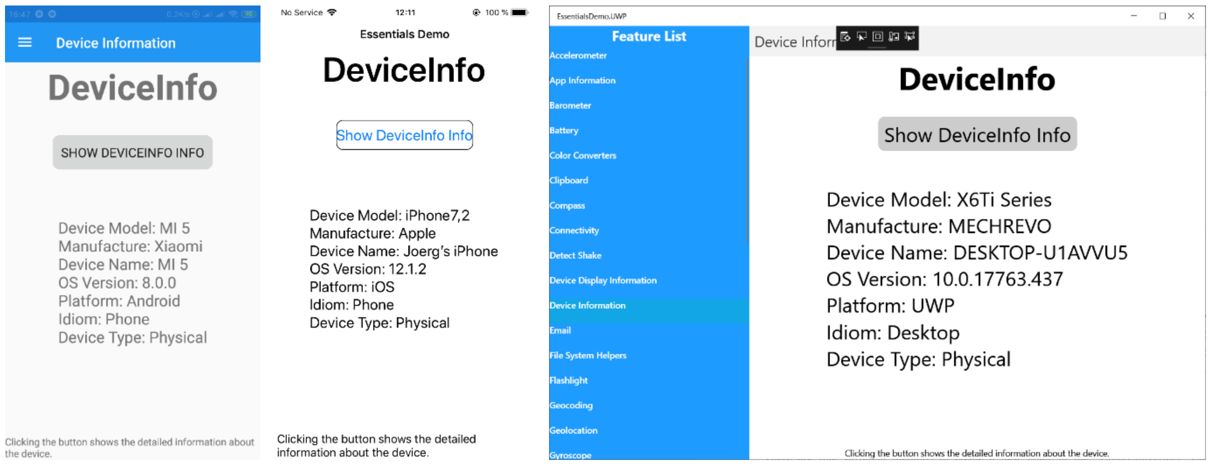
4.11 Device Information

There exist eight properties in the “DeviceInfo” class, which are “DeviceType”, “Idiom”, “Manufacturer”, “Model”, “Name”, “Platform”, “Version” and “VersionString” [36]. The “Platform” property contains values of “iOS”, “Android”, “UWP” and “Unknown”. The “Idiom” property contains values of “Phone”, “Tablet”, “Desktop”, “TV”, “Watch” and “Unknown” according to various devices. The “DeviceType” property can be “Physical”, “Virtual” or “Unknown”, which means if the application runs on a physical device or an emulator. The basic implementation codes are shown in Listing 4.14.

```
var device = DeviceInfo.Model; // Device Model
var manufacturer = DeviceInfo.Manufacturer; // Manufacturer
var deviceName = DeviceInfo.Name; // Device Name
var version = DeviceInfo.VersionString; // Operating System Version Number
var platform = DeviceInfo.Platform; // Platform
var idiom = DeviceInfo.Idiom; // Idiom
var deviceType = DeviceInfo.DeviceType; // Device Type
```

Listing 4.14 Implementation of Device Information

A button is defined to display the information of the device in the demonstration page. The performance of clicking the button on Android, iOS and UWP are shown in order in Figure 4.19.



```
Device Model: MI 5
Manufacture: Xiaomi
Device Name: MI 5
OS Version: 8.0.0
Platform: Android
Idiom: Phone
Device Type: Physical
```

Clicking the button shows the detailed information about the device.

```
Device Model: iPhone7,2
Manufacture: Apple
Device Name: Joerg's iPhone
OS Version: 12.1.2
Platform: iOS
Idiom: Phone
Device Type: Physical
```

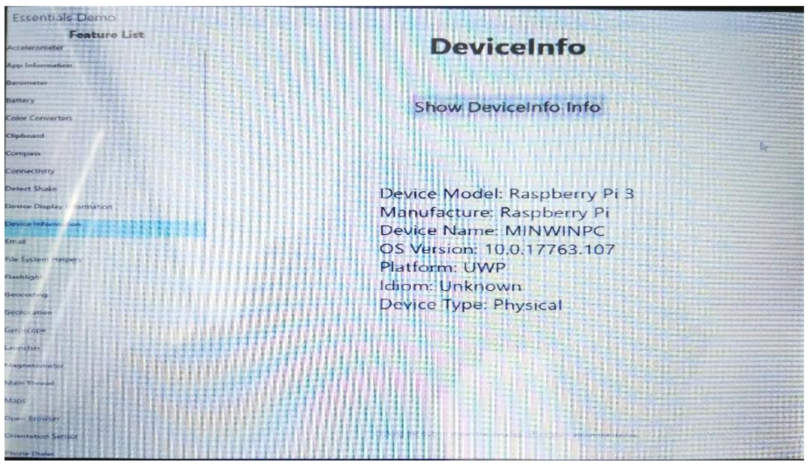
Clicking the button shows the detailed information about the device.

```
Device Model: X6Ti Series
Manufacture: MECHREVO
Device Name: DESKTOP-U1AVVU5
OS Version: 10.0.17763.437
Platform: UWP
Idiom: Desktop
Device Type: Physical
```

Clicking the button shows the detailed information about the device.

Figure 4.19 Performance of Device Information on Android, iOS & UWP

This feature is valid on Windows IoT based on Raspberry Pi as well. The performance is shown in Figure 4.20.



```
Device Model: Raspberry Pi 3
Manufacture: Raspberry Pi
Device Name: MINWINPC
OS Version: 10.0.17763.107
Platform: UWP
Idiom: Unknown
Device Type: Physical
```

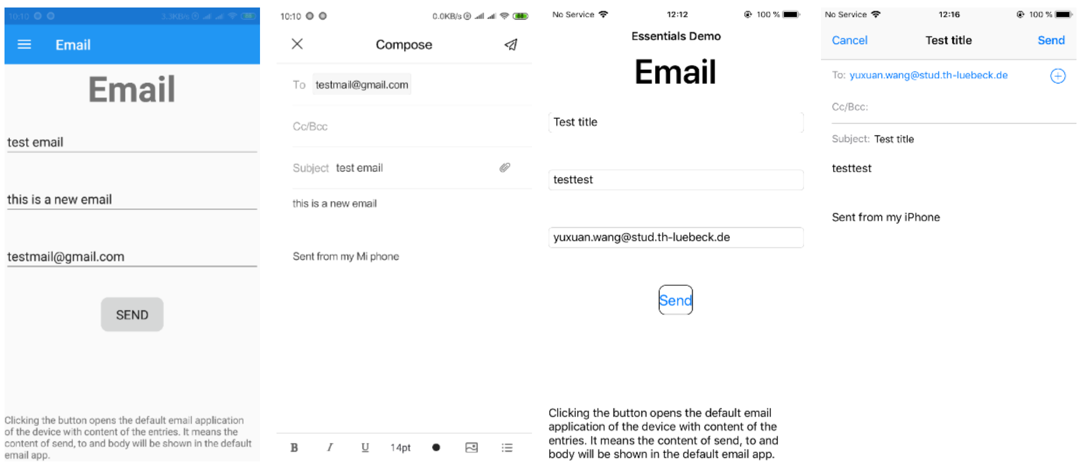
Figure 4.20 Performance of Device Information on Windows IoT

4.12 Email

There is only one method “ComposeAsync” in the class “Email”. Nevertheless, it has three different kinds of variables including “Null”, “EmailMessage” and “String, String, String[]” [37]. All of them tries to open a default email application on the device. The only difference is that the content of the email message can be edited only in the default email application when the method with “Null” variable is used. The other two methods allow users to prepare their email message within both this application and the default email application. The implementation code of the feature is shown in Listing 4.15.

```
public async Task SendEmail(string subject, string body, List<string> recipients) {
    try {
        var message = new EmailMessage {
            Subject = subject,
            Body = body,
            To = recipients,
            //Cc = ccRecipients,
            //Bcc = bccRecipients
    };
    await Email.ComposeAsync(message);
    } catch (FeatureNotSupportedException fnsEx) {
        await DisplayAlert("Error", "Feature not supported on device.", "OK");
    } catch (Exception ex) {
        await DisplayAlert("Error", "Other error has occurred.", "OK");
    }
}
```

Listing 4.15 Implementation of Email



Email

test email

this is a new email

testmail@gmail.com

Clicking the button opens the default email application of the device with content of the entries. It means the content of send, to and body will be shown in the default email app.

To testmail@gmail.com

Cc/Bcc

Subject test email

this is a new email

Sent from my Mi phone

Essentials Demo

Email

Test title

testtest

yuxuan.wang@stud.th-luebeck.de

Clicking the button opens the default email application of the device with content of the entries. It means the content of send, to and body will be shown in the default email app.

Test title

To: yuxuan.wang@stud.th-luebeck.de

Cc/Bcc:

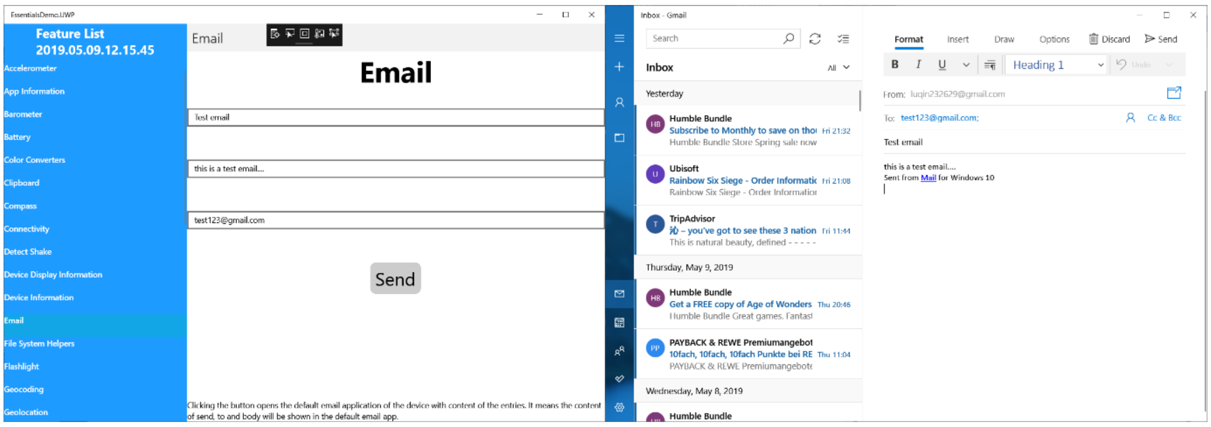
Subject: Test title

testtest

Sent from my iPhone

Figure 4.21 Performance of Email on Android & iOS

The fields “Cc” and “Bcc” in the class “EmailMessage” are not implemented in this demonstration application [28]. The performance of this feature on the Android and iOS platform is shown in order in Figure 4.21. The default email application including the entered text is opened after clicking the button.



Email

Test email

this is a test email...

test123@gmail.com

Clicking the button opens the default email application of the device with content of the entries. It means the content of send, to and body will be shown in the default email app.

Yesterday

Humble Bundle — Subscribe to Monthly to save on tho — Fri 21:32 — Humble Bundle Store Spring sale now

Ubisoft — Rainbow Six Siege - Order Informatic — Fri 21:08 — Rainbow Six Siege - Order Informatior

TripAdvisor — you've got to see these 3 nation — Fri 11:44 — This is natural beauty, defined - - - - -

Thursday, May 9, 2019

Humble Bundle — Get a FREE copy of Age of Wonders — Thu 20:46 — Humble Bundle Great games. Fantas!

PAYBACK & REWE Premiumangebot — 10fach, 10fach, 10fach Punkte bei RE — Thu 11:04 — PAYBACK & REWE Premiumangebote

Wednesday, May 8, 2019

Humble Bundle

From: luqin232629@gmail.com

To: test123@gmail.com; Cc & Bcc

Test email

this is a test email....
Sent from Mail for Windows 10

Figure 4.22 Performance of Email on UWP

The performance of the feature on the UWP platform is shown in Figure 4.22. Unfortunately, this feature is not supported on Windows IoT based on Raspberry Pi (Figure 4.23).

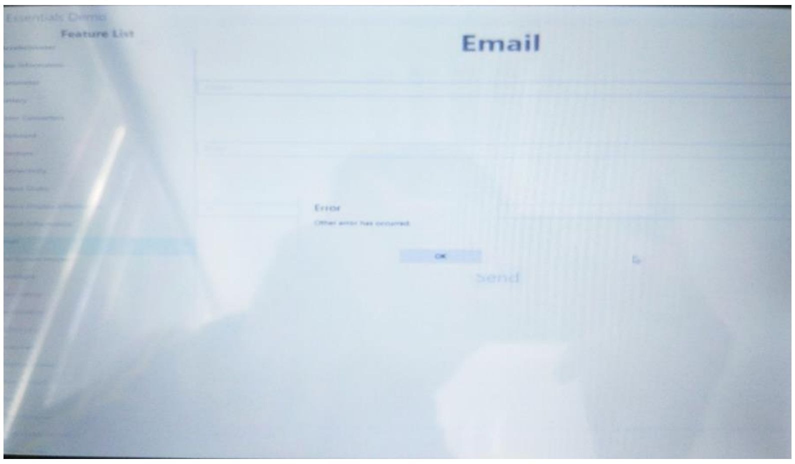


Figure 4.23 Performance of Email on Windows IoT

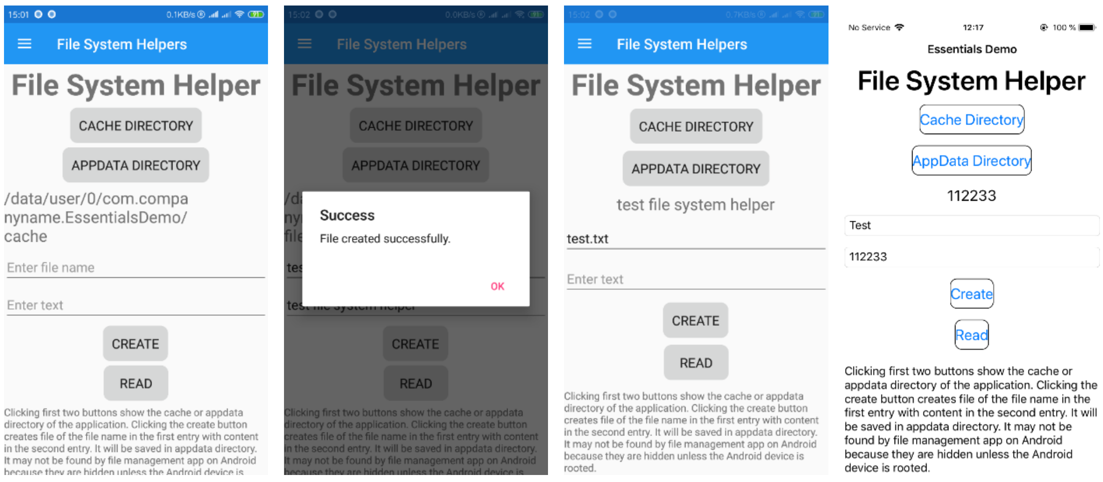
4.13 File System Helpers

There exist two properties “AppDataDirectory” and “CacheDirectory” and one method “OpenAppPackageFileAsync” in the class “FileSystem” [38]. These two properties only provide a simpler way to get the directory of app data and cache data, and the method can only open the files built in the app package. Therefore, this feature is not capable of creating or managing files. In order to manage files, another class “System.IO.File” should be used instead. On the Android platform, the method can open files added in the folder “Assets” with the build action “AndroidAsset”. On the iOS platform, the method can open files added in the folder “Resources” with the build action “BundledResource”. On the UWP platform, the method can open files added in the root folder with the build action “Content”. [18].

```
void OnButtonClicked1(object sender, EventArgs e) {
    label.Text = FileSystem.CacheDirectory;
}
void OnButtonClicked2(object sender, EventArgs e) {
    label.Text = FileSystem.AppDataDirectory;
}
void OnButtonClicked3(object sender, EventArgs e) {
    try {
        var mainDir = FileSystem.AppDataDirectory;
        var fileName = Path.Combine(mainDir, entry_path.Text);
        File.WriteAllText(fileName, entry_content.Text);
        DisplayAlert("Success", "File created successfully.", "OK");
    } catch (Exception) {
        DisplayAlert("Error", "Error has occured.", "OK");
    }
}
void OnButtonClicked4(object sender, EventArgs e) {
    try {
        var mainDir = FileSystem.AppDataDirectory;
        var fileName = Path.Combine(mainDir, entry_path.Text);
        label.Text = File.ReadAllText(fileName);
    } catch (Exception) {
        DisplayAlert("Error", "Error has occured.", "OK");
    }
}
```

Listing 4.16 Implementation of File System Helpers

The implementation of this feature is combined with the utilization of “System.IO.File” to better demonstrate it. The code is shown in Listing 4.16. The presentation page on the Android and iOS platform is shown in Figure 4.24. The first two buttons are designed to show the string of directory of app data or cache data of the current application. The create button can create a file named in the first entry with the content entered in the second entry. This file is created in the app data directory. By clicking the read button, the content of the file named in the first entry which located in the app data directory can be shown on the screen. However, it might be impossible to check the created file with a default file manager of an Android device because it might be hidden unless the device is rooted. The performance of the feature on UWP is shown in Figure 4.25.



/data/user/0/com.companyname.EssentialsDemo/cache

Clicking first two buttons show the cache or appdata directory of the application. Clicking the create button creates file of the file name in the first entry with content in the second entry. It will be saved in appdata directory. It may not be found by file management app on Android because they are hidden unless the Android device is

Success

File created successfully.

Clicking first two buttons show the cache or appdata directory of the application. Clicking the create button creates file of the file name in the first entry with content in the second entry. It will be saved in appdata directory. It may not be found by file management app on Android because they are hidden unless the Android device is

test file system helper

test.txt

Clicking first two buttons show the cache or appdata directory of the application. Clicking the create button creates file of the file name in the first entry with content in the second entry. It will be saved in appdata directory. It may not be found by file management app on Android because they are hidden unless the Android device is rooted.

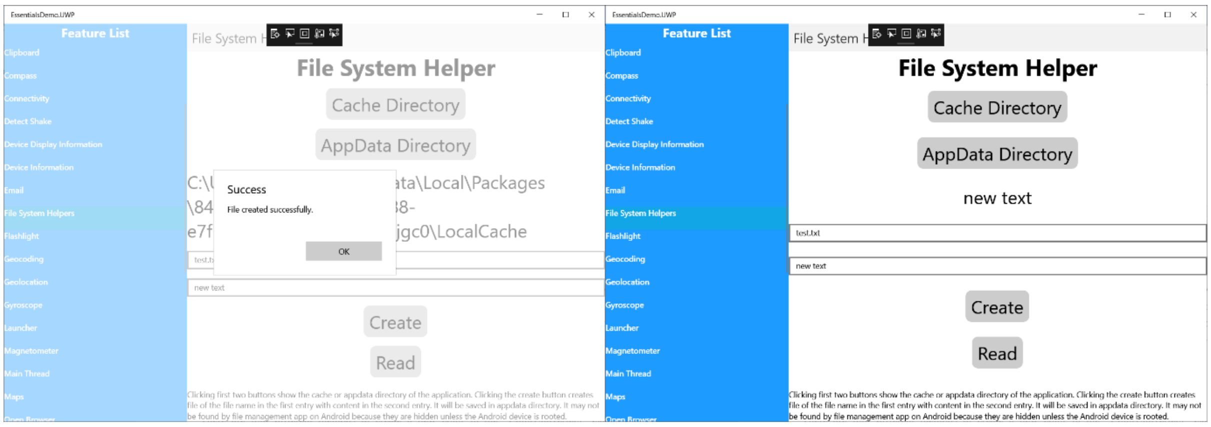
112233

Test

112233

Clicking first two buttons show the cache or appdata directory of the application. Clicking the create button creates file of the file name in the first entry with content in the second entry. It will be saved in appdata directory. It may not be found by file management app on Android because they are hidden unless the Android device is rooted.

Figure 4.24 Performance of File System Helpers on Android & iOS



C:\U ata\Local\Packages

\84 38-

e7f jgc0\LocalCache

Success

File created successfully.

test.t

new text

Clicking first two buttons show the cache or appdata directory of the application. Clicking the create button creates file of the file name in the first entry with content in the second entry. It will be saved in appdata directory. It may not be found by file management app on Android because they are hidden unless the Android device is rooted.

new text

test.txt

new text

Clicking first two buttons show the cache or appdata directory of the application. Clicking the create button creates file of the file name in the first entry with content in the second entry. It will be saved in appdata directory. It may not be found by file management app on Android because they are hidden unless the Android device is rooted.

Figure 4.25 Performance of File System Helpers on UWP

This feature is also valid on Windows IoT based on Raspberry Pi as shown in Figure 4.26.

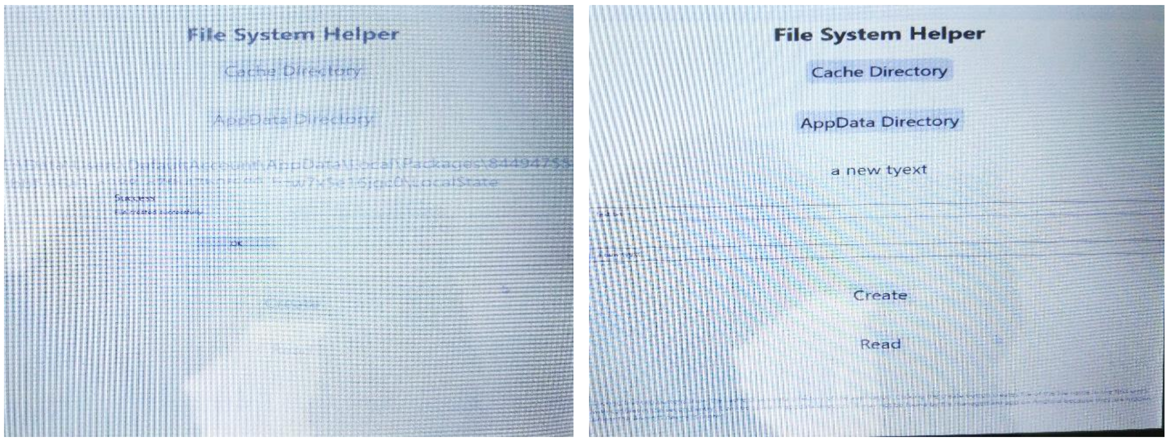


Figure 4.26 Performance of File System Helpers on Windows IoT

4.14 Flashlight

Before implementing the feature, the permission to get access to the camera and the flashlight should be added manually for the Android platform [12]. The code in Listing 4.17 should be added in the file “AssemblyInfo.cs” under the “Properties” in the Android project.

```
[assembly: UsesPermission(Android.Manifest.Permission.Flashlight)]
[assembly: UsesPermission(Android.Manifest.Permission.Camera)]
```

Listing 4.17 Camera & flashlight permission in C# [12]

Another way to add permission is to add the code in Listing 4.18 into the file “AndroidManifest.xml” under the “Properties”.

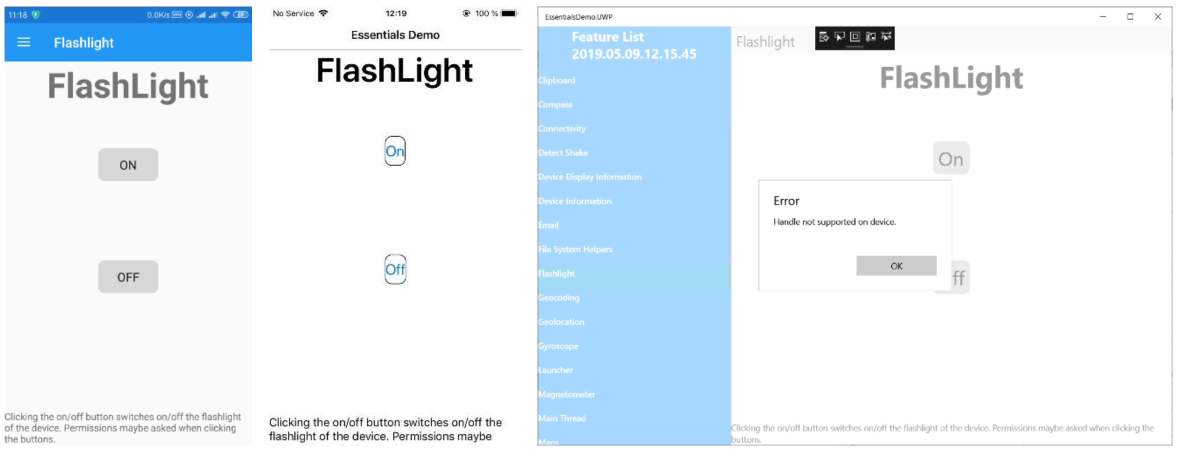
```
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.CAMERA" />
```

Listing 4.18 Camera & flashlight permission in XML [12]

There are two methods “TurnOnAsync” and “TurnOffAsync” in the class “Flashlight”. These two methods reduce codes need to be implemented on each platform. In addition, the feature does not contain any methods to control the luminosity of the flashlight.

```
await Flashlight.TurnOnAsync();
await Flashlight.TurnOffAsync();
```

Listing 4.19 Implementation of Flashlight [12]



Clicking the on/off button switches on/off the flashlight of the device. Permissions maybe asked when clicking the buttons.

Clicking the on/off button switches on/off the flashlight of the device. Permissions maybe

Error

Handle not supported on device.

Clicking the on/off button switches on/off the flashlight of the device. Permissions maybe asked when clicking the buttons.

Figure 4.27 Performance of Flashlight on Android, iOS & UWP

The performance of this feature on Android, iOS & UWP is shown in order in Figure 4.27. Unfortunately, the author does not have a UWP device which supports the feature. Additionally, the feature is not supported on Windows IoT based on Raspberry Pi either. It throws the same exception as that on UWP.

4.15 Geocoding

Geocoding is a procedure to translate a literal address into numerical coordinates on the earth. This coordinate can be used as a placemark on a map application. The procedure can also be reversed, which is called reverse geocoding. There are two methods “GetLocationAsync” and “GetPlacemarkAsync” in the class “Geocoding” [40]. The implementation of the method “GetLocationAsync” is shown in Listing 4.20.

```
var address = entry.Text.ToString();
var locations = await Geocoding.GetLocationsAsync(address);
var location = locations?.FirstOrDefault();
if (location != null) {
    label.Text = "Latitude: " + location.Latitude + "\nLongitude: "
         + location.Longitude + "\nAltitude: " + location.Altitude;
}
```

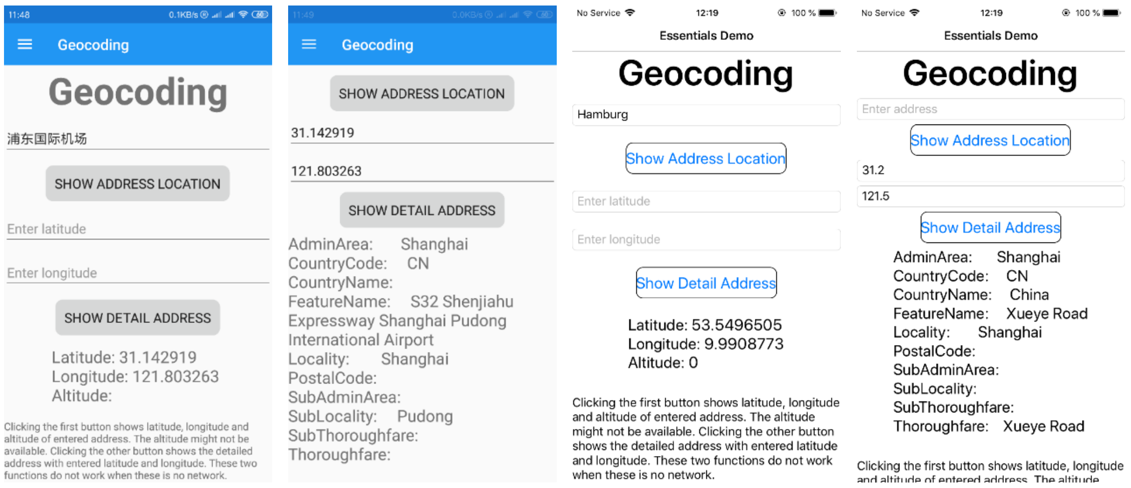
Listing 4.20 Implementation of geocoding

The method “GetPlacemarkAsync” has two kinds of variables which are one “Location” value or two double values. Since an object with the type of “Location” can be constructed with two double values, these two methods are virtually the same. The implementation of the method “GetPlacemarkAsync” is shown in Listing 4.21.

```
var lat = Convert.ToDouble(entry_lat.Text);
var lon = Convert.ToDouble(entry_lon.Text);
var placemarks = await Geocoding.GetPlacemarksAsync(lat, lon);
var placemark = placemarks?.FirstOrDefault();
if (placemark != null) {
    var geocodeAddress =
        $"AdminArea:       {placemark.AdminArea}\n" +
        $"CountryCode:     {placemark.CountryCode}\n" +
        $"CountryName:     {placemark.CountryName}\n" +
        $"FeatureName:     {placemark.FeatureName}\n" +
        $"Locality:        {placemark.Locality}\n" +
        $"PostalCode:      {placemark.PostalCode}\n" +
        $"SubAdminArea:    {placemark.SubAdminArea}\n" +
        $"SubLocality:     {placemark.SubLocality}\n" +
        $"SubThoroughfare: {placemark.SubThoroughfare}\n" +
        $"Thoroughfare:    {placemark.Thoroughfare}\n";
    label.Text = geocodeAddress;
}
```

Listing 4.21 Implementation of reverse geocoding [21]

The examples of the demonstration on Android and iOS are shown in order in Figure 4.28. It is necessary to connect the device to the network. Since this feature does not provide an offline database, the procedure of geocoding or reverse geocoding is processed online. The two buttons used the two functions of this feature respectively. The text in the first entry in the presentation of the Android platform is the Chinese characters of “Pudong International Airport”. The altitude might be unavailable according to different devices [21]. After entering the results of the geographic location, reverse geocoding is done by clicking the second button. The result on the screen of the Android platform can prove that the conclusion of geocoding is true.



Geocoding

浦东国际机场

Enter latitude

Enter longitude

Latitude: 31.142919
Longitude: 121.803263
Altitude:

Clicking the first button shows latitude, longitude and altitude of entered address. The altitude might not be available. Clicking the other button shows the detailed address with entered latitude and longitude. These two functions do not work when these is no network.

31.142919

121.803263

AdminArea: Shanghai
CountryCode: CN
CountryName:
FeatureName: S32 Shenjiahu Expressway Shanghai Pudong International Airport
Locality: Shanghai
PostalCode:
SubAdminArea:
SubLocality: Pudong
SubThoroughfare:
Thoroughfare:

Geocoding

Hamburg

Enter latitude

Enter longitude

Latitude: 53.5496505
Longitude: 9.9908773
Altitude: 0

Clicking the first button shows latitude, longitude and altitude of entered address. The altitude might not be available. Clicking the other button shows the detailed address with entered latitude and longitude. These two functions do not work when these is no network.

Geocoding

Enter address

31.2

121.5

AdminArea: Shanghai
CountryCode: CN
CountryName: China
FeatureName: Xueye Road
Locality: Shanghai
PostalCode:
SubAdminArea:
SubLocality:
SubThoroughfare:
Thoroughfare: Xueye Road

Clicking the first button shows latitude, longitude
and altitude of entered address. The altitude

Figure 4.28 Performance of Geocoding on Android & iOS

The performance of the feature on UWP is not available because of the lack of authentication from the API of Bing Maps. The author did not find a way to get the authentication through the code in the official document [21]. This feature is not valid on Windows IoT as well.

4.16 Geolocation

Before implementing the feature, the permission to get access to the information on the location should be added manually to different platforms [29]. The code in Listing 4.22 should be added in the file “AssemblyInfo.cs” under the “Properties” in the Android project.

```
[assembly: UsesPermission(Android.Manifest.Permission.AccessCoarseLocation)]
[assembly: UsesPermission(Android.Manifest.Permission.AccessFineLocation)]
[assembly: UsesFeature("android.hardware.location", Required = false)]
[assembly: UsesFeature("android.hardware.location.gps", Required = false)]
[assembly: UsesFeature("android.hardware.location.network", Required = false)]
```

Listing 4.22 Geolocation permission on Android in C# [29]

Another way to add the permission is to add the code in Listing 4.23 into the file "AndroidManifest.xml" under the "Properties" in the Android project.

```xml
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location" android:required="false" />
<uses-feature android:name="android.hardware.location.gps" android:required="false" />
<uses-feature android:name="android.hardware.location.network" android:required="false" />
```

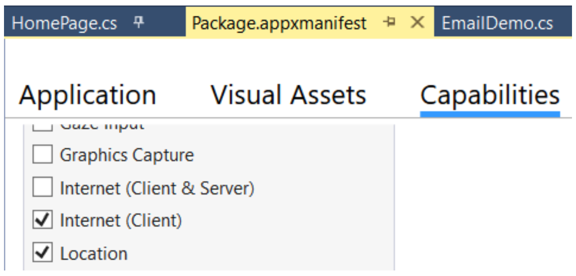
Listing 4.23 Geolocation permission on Android in XML [29]

For the iOS platform, the code in Listing 4.24 should be added to the file "Info.plist" in the iOS project.

```xml
<key>NSLocationWhenInUseUsageDescription</key>
<string>This app needs location access when open.</string>
```

Listing 4.24 Geolocation permission on iOS [29]

For the UWP platform, the checkbox of "Location" should be clicked under the tag "Capabilities" in the file "Package.appxmanifest" in the UWP project as shown in Figure 4.29.



☐ Graphics Capture
☐ Internet (Client & Server)
☑ Internet (Client)
☑ Location

Figure 4.29 Geolocation permission on UWP

There are two methods "GetLastKnownLocation" and "GetLocationAsync" in the class "Geolocation" [41]. The first method returns the last known location of the device. The second method has three different types of input values which are "Null", "GeolocationRequest" and "GeolocationRequest, CancellationToken". The variable "GeolocationRequest" can be defined by "GeolocationAccuracy" which has five categories of accuracy. They are "Lowest", "Low", "Medium", "High" and "Best". More detailed statistics can be referred from the official document [29]. The variable "CancellationToken" can be set as a time period so that the request can be terminated after waiting for a long time without response. The detailed implementation of the two methods is shown in Listing 4.25 and Listing 4.26.

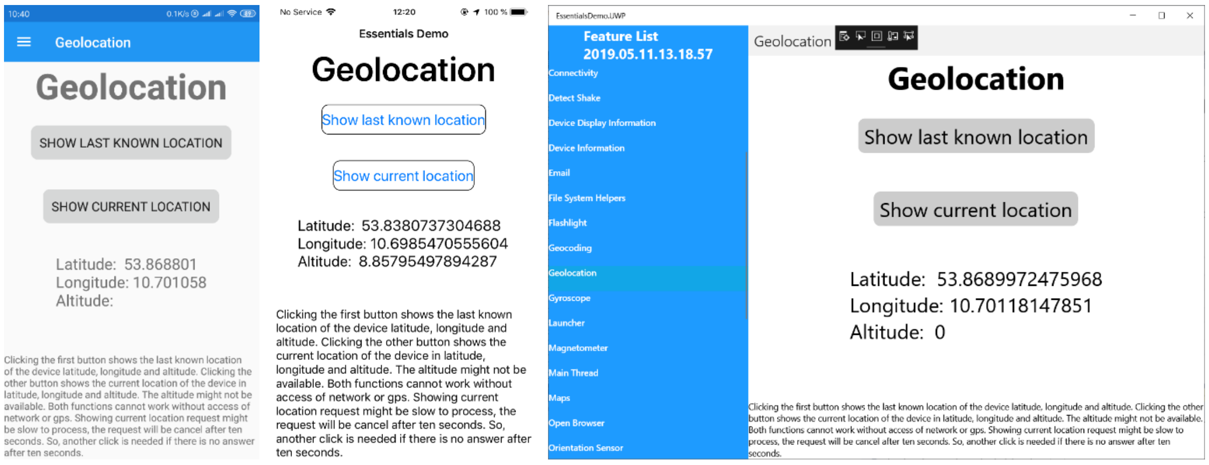
```
var location = await Geolocation.GetLastKnownLocationAsync();
if (location != null) {
label.Text = $"Latitude:  {location.Latitude}\n" +
             $"Longitude: {location.Longitude}\n" +
             $"Altitude:  {location.Altitude}";
}
```

Listing 4.25 Implementation of Geolocation – 1 [29]

```
var request = new GeolocationRequest(GeolocationAccuracy.Me-
dium, TimeSpan.FromSeconds(10));
var location = await Geolocation.GetLocationAsync(request);
if (location != null) {
label.Text = $"Latitude:  {location.Latitude}\n" +
             $"Longitude: {location.Longitude}\n" +
             $"Altitude:  {location.Altitude}";
}
```

Listing 4.26 Implementation of Geolocation – 2 [29]

As the example presentation of Android, iOS and UWP shown in Figure 4.30, the current location will show after clicking the second button. The accuracy can be proved by entering the latitude and longitude in the geocoding page mentioned in Section 4.15. This feature is not available on Windows IoT based on Raspberry Pi.



Geolocation

Latitude: 53.868801
Longitude: 10.701058
Altitude:

Clicking the first button shows the last known location of the device latitude, longitude and altitude. Clicking the other button shows the current location of the device in latitude, longitude and altitude. The altitude might not be available. Both functions cannot work without access of network or gps. Showing current location request might be slow to process, the request will be cancel after ten seconds. So, another click is needed if there is no answer after ten seconds.

Geolocation

Latitude: 53.8380737304688
Longitude: 10.6985470555604
Altitude: 8.85795497894287

Clicking the first button shows the last known location of the device latitude, longitude and altitude. Clicking the other button shows the current location of the device in latitude, longitude and altitude. The altitude might not be available. Both functions cannot work without access of network or gps. Showing current location request might be slow to process, the request will be cancel after ten seconds. So, another click is needed if there is no answer after ten seconds.

Geolocation

Latitude: 53.8689972475968
Longitude: 10.70118147851
Altitude: 0

Clicking the first button shows the last known location of the device latitude, longitude and altitude. Clicking the other button shows the current location of the device in latitude, longitude and altitude. The altitude might not be available. Both functions cannot work without access of network or gps. Showing current location request might be slow to process, the request will be cancel after ten seconds. So, another click is needed if there is no answer after ten seconds.

Figure 4.30 Performance of Geolocation on Android, iOS & UWP

4.17 Gyroscope

Gyroscope is a motion sensor which measures the rotation speed in radian per second around x, y and z-axis. The coordinate is equivalent to that mentioned in the accelerometer class in

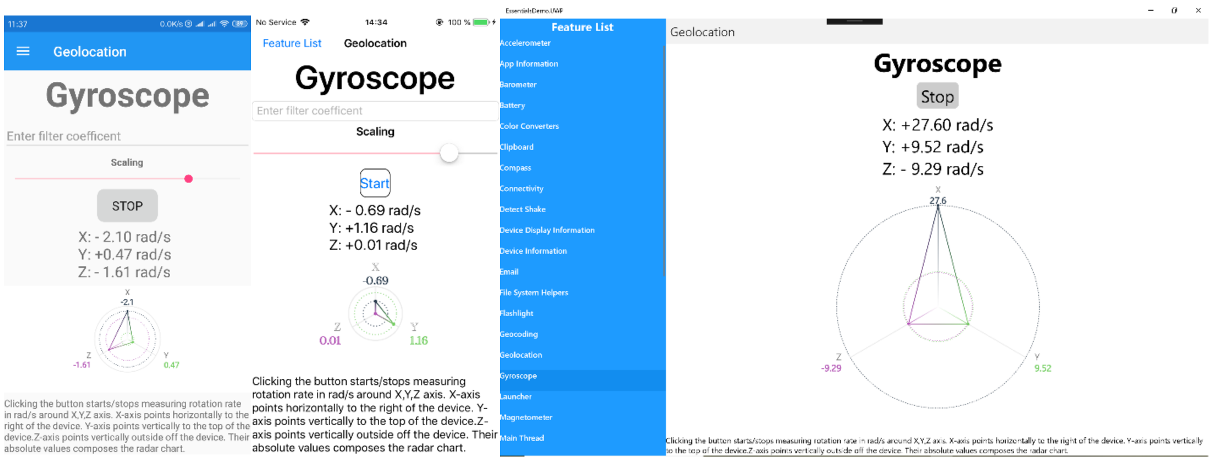
Section 4.1. There exists one property “IsMonitoring”, two methods “Start” and “Stop” and one event “ReadingChanged” in the class “Gyroscope” [42]. The implementation of the two methods is similar to that in Listing 4.3. To eliminate noise, an average low-pass filter is used, which is mentioned in Section 4.1. The values of rotation speed are positive in the counter-clockwise direction. The speed of the sensor, the registration of “ReadingChanged” and the implementation of the two methods are similar to those of the accelerometer in Section 4.1, which is omitted here.

```
using Entry = Microcharts.Entry;
using Microcharts;
using Microcharts.Forms;
double scaling = Math.Pow(10, slider.value);
slider = new Slider {
    Maximum = 2,
    Minimum = -2
}
ChartView chartView = new ChartView();
var entries = new[] {
    new Entry((float)Math.Abs(Math.Round(data_X * scaling, 2))) {
        Label = "X",
        ValueLabel = Math.Round(data_X, 2).ToString(),
        Color = SKColor.Parse("#2c3e50")
    },
    new Entry((float)Math.Abs(Math.Round(data_Y * scaling, 2))) {
        Label = "Y",
        ValueLabel = Math.Round(data_Y, 2).ToString(),
        Color = SKColor.Parse("#77d065")
    },
    new Entry((float)Math.Abs(Math.Round(data_Z * scaling, 2))) {
        Label = "Z",
        ValueLabel = Math.Round(data_Z, 2).ToString(),
        Color = SKColor.Parse("#b455b6")
    }
};
chartView.Chart = new RadarChart() {
    Entries = entries,
    PointSize = 10,
    MaxValue = 30,
    MinValue = 0
};
```

Listing 4.27 Implementation of Mirocharts

To improve the demonstration, instead of three single numbers shown on the screen, Microcharts is used. It is a NuGet package that is available for the cross-platform application. There are different types of charts in this package. In this case, the radar chart is used. Although it can only display absolute values, the demonstration of radar on three axes can probably improve the imagination of users on the three-dimensional coordinate. The detail implementation of Microcharts is shown in Listing 4.27. The code in the first three lines of Listing 4.27 should be added at the beginning of the file. From the 5th to 9th line should be contained in the constructor of the content page. Rest of the code defines the entries and the style of the chart.

The variable “scaling” is used to adjust the size of the diagram in the radar chart. One of the reasons to use this variable is that the sensors in different devices can probably return disparate magnitude of values. For instance, when rotating a tablet, the value is usually around thirty, while it is only around 3 on a phone. Since the maximum of the chart is fixed, a coefficient is needed to adjust the values. As shown in Listing 4.27, an exponential method is used. With this method, a small range of the slider from -2 to +2 can lead to a huge range of coefficient from 0.01 to 100. It is much better than a simple linear coefficient slider for users to control.



X: - 2.10 rad/s
Y: +0.47 rad/s
Z: - 1.61 rad/s

Clicking the button starts/stops measuring rotation rate in rad/s around X,Y,Z axis. X-axis points horizontally to the right of the device. Y-axis points vertically to the top of the device.Z-axis points vertically outside off the device. Their absolute values composes the radar chart.

X: - 0.69 rad/s
Y: +1.16 rad/s
Z: +0.01 rad/s

Clicking the button starts/stops measuring rotation rate in rad/s around X,Y,Z axis. X-axis points horizontally to the right of the device. Y-axis points vertically to the top of the device.Z-axis points vertically outside off the device. Their absolute values composes the radar chart.

X: +27.60 rad/s
Y: +9.52 rad/s
Z: - 9.29 rad/s

Clicking the button starts/stops measuring rotation rate in rad/s around X,Y,Z axis. X-axis points horizontally to the right of the device. Y-axis points vertically to the top of the device.Z-axis points vertically outside off the device. Their absolute values composes the radar chart.

Figure 4.31 Performance of Gyroscope on Android, iOS & UWP

The performance of this feature on Android, iOS and UWP is shown in order in Figure 4.31. This feature is not available on Windows IoT based on Raspberry Pi without throwing an exception.

4.18 Launcher

The class “Launcher” can open a URI (uniform resource identifier) by the application. A uniform resource identifier determines an explicit resource with uniform syntax rules. According to RFC (Request for Comments) 2396, the rules which are called URI generic syntax has the format as “scheme:[//authority]path[?query][#fragment]” to define all the URIs. Each part of the format has its specific meaning. They can be referred to RFC 3986, which is the ultimate version of the specification of URI.

There exist two methods “CanOpenAsync” and “OpenAsync” in the class “Launcher” [43]. Both of them can be executed with input values of type “String” or “Uri”. The former one returns a task with type “boolean” to check if the URI scheme is supported on the current device. The other method just opens a URI to launch the specific application. Some examples are provided here. “smsto:” can open the default SMS application of the device. “tel:” can launch the default telephone application of the device. “http://[website]” can open the default browser of the device. On the Android platform, “content://contacts/people” can launch the default contact application of the device while “market://search” can launch the default app market of the device. The detail implement codes are shown in Listing 4.28.

```
try {
    Uri uri = new Uri(text.Text);
    //var supportsUri = await Launcher.CanOpenAsync(uri);
    await Launcher.OpenAsync(uri);
} catch (UriFormatException) {
    await DisplayAlert("Alert", "The entered uri is not supported.", "OK");
} catch (Exception) {
    await DisplayAlert("Alert", "Error has occured.", "OK");
}
```

Listing 4.28 Implementation of Launcher

The performance of the launcher on Android and iOS is shown in order in Figure 4.32. On the Android platform, the default telephone application will be launched after clicking the button “Launch”. Meanwhile, a message is prompted to let users call with the number.

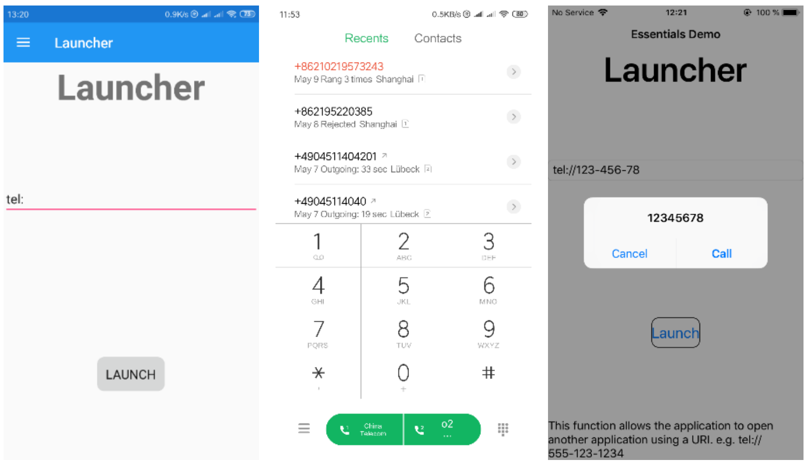
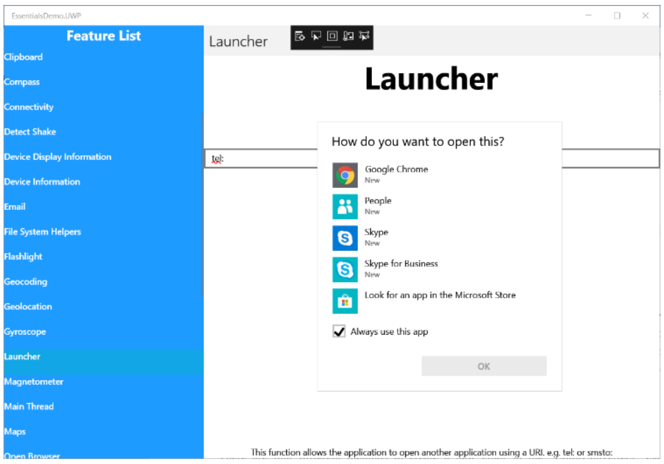


Figure 4.32 Performance of Launcher on Android & iOS

The performance of the feature on UWP is shown in Figure 4.33. A selection message is given that users can select a suitable application to execute the URI. This feature gives no response after clicking the button on Windows IoT based on Raspberry Pi.



How do you want to open this?

This function allows the application to open another application using a URI. e.g. tel: or smsto:

Figure 4.33 Performance of Launcher on UWP

4.19 Additional Task

Although Xamarin.Essentials has lots of features for developers to utilize, there does not exist a feature to access the library or the camera of the device. This section will demonstrate how to implement this on Android, iOS and UWP.

4.19.1 Image Picker

Firstly, an interface containing a method with a return type of "Task<Stream>" should be created in the shared project. This method is used to get the stream of the image and should be implemented in Android, iOS and UWP platforms respectively. To realize this, a vital class "DependencyService" is needed, which enables the application to call the respective functions related to the interface implemented on each platform from the shared code [11]. The simpler explanation is shown as in Figure 4.34.

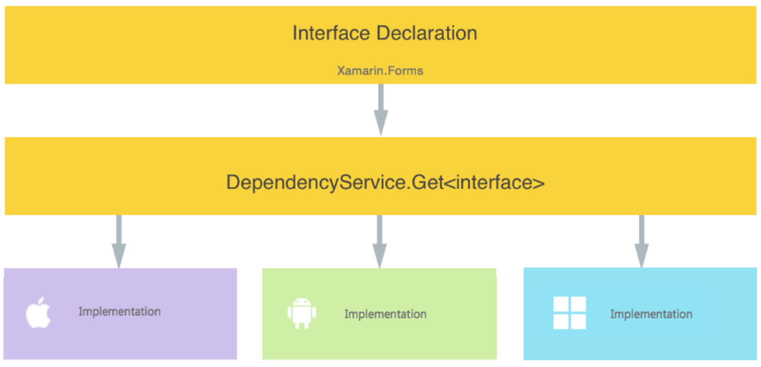


Figure 4.34 Structure of DependencyService [11]

The detailed implementation on each platform can be referred to the official document [8]. On the iOS platform, the permission should be granted by developers. To realize this, the code in Listing 4.29 should be added to the node “dict” of the file “Info.plist” in the iOS project.

```
<key>NSPhotoLibraryUsageDescription</key>
<string>Image Picker uses photo library</string>
```

Listing 4.29 Permission of Image Picker on iOS

On the other hand, additional field, property and method related to activity should be added in the class “MainActivity” in the Android project. The implementation is shown in Listing 4.30.

```
// Field, property, and method for Picture Picker
public static readonly int PickImageId = 1000;
public TaskCompletionSource<Stream> PickImageTaskCompletion-
Source { set; get; }
protected override void OnActivityResult(int requestCode, Result re-
sultCode, Intent intent) {
base.OnActivityResult(requestCode, resultCode, intent);
if (requestCode == PickImageId) {
    if ((resultCode == Result.Ok) && (intent != null)) {
        Android.Net.Uri uri = intent.Data;
        Stream stream = ContentResolver.OpenInputStream(uri);
        // Set the Stream as the completion of the Task
        PickImageTaskCompletionSource.SetResult(stream);
    } else {
        PickImageTaskCompletionSource.SetResult(null);
        }
    }
}
```

Listing 4.30 Implementation of MainActivity on Android [8]

The final step is to implement the shared code in the content page of the image picker. The implementation is contained in the click action of the button “Pick”. To show the image, a new instance of the image should be created with the returned stream, and the content of the demonstration page should be recreated [8].

The performance this image picker on Android, iOS and UWP are shown in order in Figure 4.35. The photo library is shown after clicking the first button. After picking a photo, the photo is presented in the application. It can be removed by tapping on it. The function of the second button is described in Section 4.19.2. Image picker is not available on Windows IoT based on Raspberry Pi.

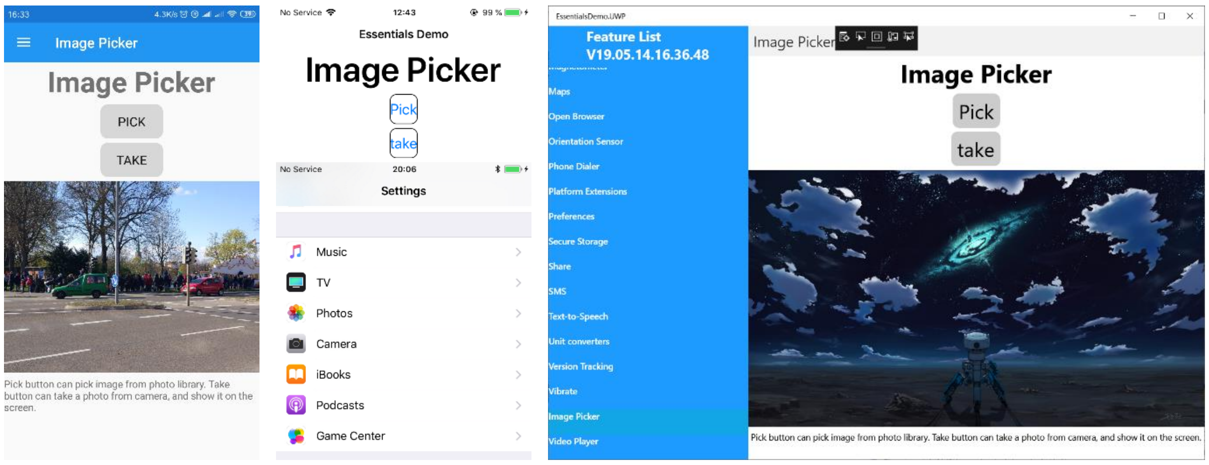


Figure 4.35 Performance of Image Picker on Android, iOS & UWP

4.19.2 Photo from Camera

To pick the photo captured by the camera of the device, a NuGet package called "Xam.Plugin.Media" can be used to simplify the implementation. The package should be installed on all projects. The installation of the package is similar to that of "Xamarin.Essentials" in Section 4.

Before the implementation, several setup preparations should be done on Android, iOS and UWP [6]. On the Android platform, the permission plugin code in Listing 4.31 should be added to the class "MainActivity". Since this NuGet Package is based on "Current Activity Plugin", a NuGet package called "Plugin.CurrentActivity" should be installed only on the Android project. After the installation, there exist two methods for initialization [4]. In this thesis, the first one is recommended because it is unnecessary to create a new class to increase the load of the application. Therefore, one single line of Listing 4.32 should be added to the method "OnCreate" in the class "MainActivity".

```
public override void OnRequestPermissionsResult(int requestCode, string[] permis-
sions, Android.Content.PM.Permission[] grantResults) {
    Plugin.Permissions.PermissionsImplementation.Current.OnRequestPermis-
sionsResult(requestCode, permissions, grantResults);
}
```

Listing 4.31 Permission plugin code [6]

```
CrossCurrentActivity.Current.Init(this, bundle);
```

Listing 4.32 Initialization of CurrentActivity [6]

To totally guarantee the camera permission on Android, the code in Listing 4.33 should be added to the file “AssemblyInfo.cs”. After that, some configuration code in

Listing **4.34** related to Android file provider should be added to the node “application” in the file “AndroidManifest.xml”. Furthermore, a new folder “xml” with build action “AndroidResource” including a file “file_paths.xml” should be created under the folder “Resources” in the Android project. The file should include the code in Listing 4.35 [6].

```
[assembly: UsesFeature("android.hardware.camera", Required = false)]
[assembly: UsesFeature("android.hardware.camera.autofocus", Required = false)]
```

Listing 4.33 Permission of camera on Android [6]

```
<provider android:name="android.support.v4.content.FileProvider"
          android:authorities="${applicationId}.fileprovider"
          android:exported="false"
          android:grantUriPermissions="true">
      <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
                 android:resource="@xml/file_paths"></meta-data>
</provider>
```

Listing 4.34 Configuration code for Android file provider – 1 [6]

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-files-path name="my_images" path="Pictures" />
    <external-files-path name="my_movies" path="Movies" />
</paths>
```

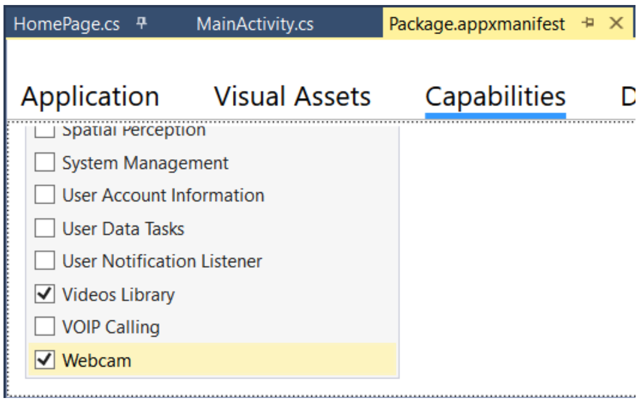
Listing 4.35 Configuration code for Android file provider – 2 [6]

The process of permission on the iOS platform is simpler. The code in Listing 4.36 should be added to the file “Info.plist” in the iOS project [6].

```
<key>NSCameraUsageDescription</key>
<string>This app needs access to the camera to take photos.</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>This app needs access to photos.</string>
<key>NSPhotoLibraryAddUsageDescription</key>
<string>This app needs access to the photo gallery.</string>
```

Listing 4.36 Permission of camera on iOS [6]

The process of permission on the UWP platform is the simplest. The checkbox of “Webcam” should be clicked under the tag “Capabilities” in the file “Package.appxmanifest” in the UWP project as shown in Figure 4.36 [6].



- [ ] Spatial Perception
- [ ] System Management
- [ ] User Account Information
- [ ] User Data Tasks
- [ ] User Notification Listener
- [x] Videos Library
- [ ] VOIP Calling
- [x] Webcam

Figure 4.36 Permission of camera on UWP

The detailed implementation of picking a photo from the camera is shown in Listing 4.37 and Listing 4.38. Since the permission requirement on each platform is completely different, a permission check is recommended before every access to the camera [6]. The code in Listing 4.37 show the permission check process. The code in Listing 4.38 implement the function and recreate the content page.

```
// Check permission before every access
if (!CrossMedia.Current.IsCameraAvailable || !CrossMedia.Current.IsTakePhotoSupported) {
    await DisplayAlert("No Camera", ":( No camera avaialble.", "OK");
    return;
}
var cameraStatus = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Camera);
var storageStatus = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Storage);

if (cameraStatus != PermissionStatus.Granted || storageStatus != PermissionStatus.Granted) {
    var results = await CrossPermissions.Current.RequestPermissionsAsync(new[] { Permission.Camera, Permission.Storage });
    cameraStatus = results[Permission.Camera];
    storageStatus = results[Permission.Storage];
}
if (cameraStatus == PermissionStatus.Granted && storageStatus == PermissionStatus.Granted) {
```

Listing 4.37 Implementation of picking a photo from the camera - 1 [6]

```
    // Codes to get access to camera and take photos
    var file = await CrossMedia.Current.TakePhotoAsync(new Plugin.Media.Abstractions.S
toreCameraMediaOptions {
        PhotoSize = Plugin.Media.Abstractions.PhotoSize.Medium,
        Directory = "Sample",
        Name = "test.jpg"
    });
    if (file == null)
        return;
    // create image with the photo just took and stored
    image = new Image {
        Source = ImageSource.FromStream(() => {
            var stream = file.GetStream();
            file.Dispose();
            return stream;
        }),
        BackgroundColor = Color.Gray,
        VerticalOptions = LayoutOptions.Center,
        HorizontalOptions = LayoutOptions.Center
    };
    // Recreate the content page
    scrollView = new StackLayout { Children = { image } };
}
else {
    await DisplayAlert("Permissions Denied", "Unable to take photos.", "OK");
}
```

Listing 4.38 Implementation of picking a photo from the camera - 2 [6]

The performance of this task in the application is the same as the image picker as shown in Figure 4.35. The default camera application of the device will be launched after clicking the button “Take”. The taken photo will be shown on the screen afterward.

4.19.3 Other Implementation

The Icon of the demonstration application is designed by the author. It utilizes the first letter of the word “Essentials” as shown in Figure 4.37. To adapt the icon to the application, the default icons in the Android, iOS and UWP project should be replaced manually. The storage places of these icons are slightly different. In the Android project, it is the folder “Resouces”, while it is the folder “Assets.xcassets” in the iOS project, and it is the folder “Assets” in the UWP project. It is necessary to resize the designed icon to make it available to these projects. In this thesis, a resize application is used, which is designed by Jörg Bayerlein. It can generate all the needed sizes of an icon automatically [49]. The detailed replacement rule can be referred to the document written by Jörg Bayerlein [48].

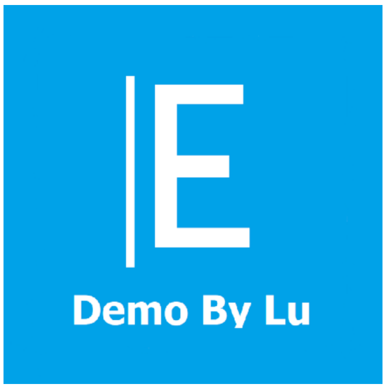


Figure 4.37 Icon of the application

In addition, a version number is added in the feature list as shown in Figure 4.1. It is implemented by using the build time of the application (Listing 4.39). However, the actual version of the application is not changed at the same time. The author does not find the approach to synchronize the versions on different platforms and to automatically increase them, which is a limitation of this application.

```
System.IO.File.GetLastWriteTime(System.Reflection.Assembly.GetExecutingAssem
bly().Location).ToString("yy.MM.dd.HH.mm.ss");
```

Listing 4.39 Get the build time of the project

The introduction of each feature is described in each presentation page as shown in the figures in Section 4. It can help users and other developers to have a better understanding of the feature.

4.19.4 Problems of the other half of the features on Android

Firstly, the preferences are removed when the application is uninstalled unless “Auto Backup” is used. This feature is enabled by default on Android 6.0 or later versions. Additionally, the performance of the feature “Preferences” will be poor if large amount of text is stored. This phenomenon appears in the feature “Secure Storage” as well.

5 Conclusion and outlook

The purpose of this study is to investigate the possibilities and limitations of Xamarin.Essentials on Android phone. In this paper, a mobile application is developed to demonstrate the features in Xamarin.Essentials. Performance of half of the features on different platforms is evaluated.

The performance has shown that virtually all the features are capable of executing the shared code on Android, iOS and UWP. Nevertheless, some of the features considering the utilization of sensors might not be supported as a result of lacking a specific sensor on the device. For instance, the tablet “Samsung SM-T585” does not contain most of the sensors and cannot support those features as a result. The application has a slight difference in the demonstration due to the style of the platform. Moreover, the information provided by the features can be processed in a more understandable way with the help of Xamarin.Forms. Additionally, there exist other NuGet packages, such as “Microcharts”, to make a good presentation of the features.

Xamarin.Essentials gives the possibility to get access to sensors, information and default applications of the device based on different platforms with least implementation of the code. It reduces the workload of developers to develop cross-platform applications.

On the other hand, there still exists some limitations of Xamarin.Essentials. Firstly, some features do not provide complete functions. For instance, it is impossible to change the luminosity of the flashlight with the feature “Flashlight”. There does not exist any method to create files in the feature “File System Helpers”. Secondly, although it reduces the workload of developers, it increases the load of the application to some extent. If only one feature of Xamarin.Essentials is used during cross-platform development, it is unnecessary to use such a large package. Instead, developers can make the implementation with their own code. At last but not least, Xamarin.Essentials lacks some useful functions, such as image picker, which is the additional task of this study. It requires great effort to implement such a function on different platforms as shown in Section 4.19.1. The implementation would be simpler if it is contained in Xamarin.Essentials.

For further research on Xamarin.Essentials, several factors can be further investigated. Firstly, some meaningful cross-platform mobile applications can be developed with the use of Xamarin.Essentials. For instance, an application with the utilization of the accelerometer can be developed to measure the force of a punch of one person. Secondly, as mentioned in the limitations, more useful functions should be developed and be accumulated in Xamarin.Essentials. Finally, the approach to increase the version number automatically in the cross-platform projects should be investigated.

Acknowledgments

First and foremost, I want to appreciate my supervisor Prof. Bayerlein for providing me the thesis topic and all the related facilities, devices and vital instructions. With the help of these, I can finish my thesis successfully.

Secondly, I would like to present my gratitude to Mrs. Kleinau who gives me information on thesis writing. Her efforts of teaching me during my study in THL also laid foundations for my basic knowledge on implementation of applications.

Thirdly, I want to express my thankfulness to my friend Yuxuan Wang who gives me support on the iOS platform.

Last but not least, I would like to give the acknowledgment to my families, classmates and anyone who gave me a hand during my study, which helps me to overcome all the difficulties and keep progressing.

Bibliography

[1] Furlan, A., 2018. *Cross Platform Mobile App Development Guide (2017).* [Online] Available at: http://www.businessofapps.com/guide/cross-platform-mobile-app-development/ [Accessed 30 April 2019].

[2] Google Developers, n.d. *Motion sensors.* [Online] Available at: https://developer.android.com/guide/topics/sensors/sensors_motion [Accessed 2 May 2019].

[3] Google Developers, n.d. *Sensors Overview.* [Online] Available at: https://developer.android.com/guide/topics/sensors/sensors_overview.html [Accessed 2 May 2019].

[4] jamesmontemagno, 2018. *CurrentActivityPlugin.* [Online] Available at: https://github.com/jamesmontemagno/CurrentActivityPlugin/blob/master/README.md [Accessed 4 May 2019].

[5] jamesmontemagno, 2019. *Accelerometer.shared.cs.* [Online] Available at: https://github.com/xamarin/Essentials/blob/master/Xamarin.Essentials/Accelerometer/Accelerometer.shared.cs [Accessed 4 May 2019].

[6] jamesmontemagno, n.d. *MediaPlugin.* [Online] Available at: https://github.com/jamesmontemagno/MediaPlugin [Accessed 4 May 2019].

[7] Microsoft, 2015. *Introduction to the C# Language and the .NET Framework.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework [Accessed 1 May 2019].

[8] Microsoft, 2017. *Picking a Photo from the Picture Library.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/dependency-service/photo-picker [Accessed 4 May 2019].

[9] Microsoft, 2017. *Xamarin.Forms Master-Detail Page.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/navigation/master-detail-page [Accessed 4 May 2019].

[10] Microsoft, 2018. *Get Started with Xamarin.Essentials.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/get-started?context=xamarin%2Fandroid&tabs=windows%2Candroid [Accessed 2 May 2019].

[11] Microsoft, 2018. *Introduction to DependencyService.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/dependency-service/introduction [Accessed 4 May 2019].

[12] Microsoft, 2018. *Xamarin.Essentials.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/flashlight?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019].

[13] Microsoft, 2018. *Xamarin.Essentials: Barometer.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/barometer?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019].

[14] Microsoft, 2018. *Xamarin.Essentials: Compass.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/compass?context=xamarin%2Fandroid&tabs=android [Accessed 3 May 2019].

[15] Microsoft, 2018. *Xamarin.Essentials: Detect Shake.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/detect-shake?context=xamarin/android [Accessed 4 May 2019].

[16] Microsoft, 2018. *Xamarin.Essentials: Device Display Information.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/device-display?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019]

[17] Microsoft, 2018. *Xamarin.Essentials: Device Information.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/device-information?context=xamarin%2Fandroid&tabs=ios [Accessed 4 May 2019].

[18] Microsoft, 2018. *Xamarin.Essentials: File System Helpers.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/file-system-helpers?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019].

[19] Microsoft, 2018. *Xamarin.Essentials: Gyroscope.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/gyroscope?context=xamarin/android [Accessed 4 May 2019].

[20] Microsoft, 2018. *Xamarin.Essentials: Launcher.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/launcher?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019].

[21] Microsoft, 2018. *Xamarin.Essentials:Geocoding.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/geocoding?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019].

[22] Microsoft, 2019. *Welcome to the Visual Studio IDE.* [Online] Available at: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2017 [Accessed 1 May 2019].

[23] Microsoft, 2019. *Xamarin.Essentials.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/ [Accessed 30 April 2019].

[24] Microsoft, 2019. *Xamarin.Essentials: Accelerometer.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/accelerometer?context=xamarin/android [Accessed 4 May 2019].

[25] Microsoft, 2019. *Xamarin.Essentials: App Information.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/app-information?context=xamarin%2Fandroid&tabs=android [Accessed 3 May 2019].

[26] Microsoft, 2019. *Xamarin.Essentials: Battery.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/battery?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019].

[27] Microsoft, 2019. *Xamarin.Essentials: Connectivity.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/connectivity?context=xamarin%2Fandroid&tabs=ios [Accessed 4 May 2019].

[28] Microsoft, 2019. *Xamarin.Essentials: Email.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/email?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019].

[29] Microsoft, 2019. *Xamarin.Essentials: Geolocation.* [Online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/geolocation?context=xamarin%2Fandroid&tabs=android [Accessed 4 May 2019].

[30] Microsoft, n.d. *BatteryState Enum.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.batterystate?view=xamarin-essentials [Accessed 3 May 2019].

[31] Microsoft, n.d. *Clipboard Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.clipboard?view=xamarin-essentials [Accessed 4 May 2019].

[32] Microsoft, n.d. *ColorConverters Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.colorconverters?view=xamarin-essentials [Accessed 3 May 2019].

[33] Microsoft, n.d. *ColorExtensions Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.colorextensions?view=xamarin-essentials [Accessed 3 May 2019].

[34] Microsoft, n.d. *Compass Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.compass?view=xamarin-essentials [Accessed 4 May 2019].

[35] Microsoft, n.d. *DeviceDisplay Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.devicedisplay?view=xamarin-essentials [Accessed 4 May 2019].

[36] Microsoft, n.d. *DeviceInfo Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.deviceinfo?view=xamarin-essentials [Accessed 4 May 2019].

[37] Microsoft, n.d. *Email Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.email?view=xamarin-essentials [Accessed 4 May 2019].

[38] Microsoft, n.d. *FileSystem Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.filesystem?view=xamarin-essentials [Accessed 4 May 2019].

[39] Microsoft, n.d. *Flashlight Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.flashlight?view=xamarin-essentials [Accessed 4 May 2019].

[40] Microsoft, n.d. *Geocoding Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.geocoding?view=xamarin-essentials [Accessed 4 May 2019].

[41] Microsoft, n.d. *Geolocation Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.geolocation?view=xamarin-essentials [Accessed 4 May 2019].

[42] Microsoft, n.d. *Gyroscope Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.gyroscope?view=xamarin-essentials [Accessed 4 May 2019].

[43] Microsoft, n.d. *Launcher Class.* [Online] Available at: https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.launcher?view=xamarin-essentials [Accessed 4 May 2019].

[44] Microsoft, n.d. *Visual Studio.* [Online] Available at: https://visualstudio.microsoft.com/ [Accessed 1 May 2019].

[45] Petzold, C., 2016. How does Xamarin.Forms fit in?. In: *Creating Mobile Apps with Xamarin.Forms.* s.l.:Microsoft Press, pp. 6-8.

[46] Reynolds, M., 2014. Xamarin and Mono – a Pathway to the Unnatural. In: *Xamarin Essentials.* s.l.:Packt Publishing Ltd., pp. 9-10, 126-127.

[47] Visual Studio, 2018. *Hyper-V Android emulator support.* [Online] Available at: https://devblogs.microsoft.com/visualstudio/hyper-v-android-emulator-support/ [Accessed 1 May 2019].

[48] Bayerlein, J., 2014. *Xamarin Forms Docs : App Icons.* [Online] Available at: http://www.joergbayerlein.de/wp-content/uploads/2019/02/App-Icons-US-V1.pdf [Accessed 1 May 2019].

[49] Bayerlein, J., 2014. *App Icons for Xamarin Forms.* [Online] Available at: http://www.joergbayerlein.de/?page_id=446 [Accessed 1 May 2019].

[50] Wang, Y., 2019. *Investigation and Test of Xamarin.Essentials in Visual Studio on iPhone*

Appendix A - List of figures

Appendix B - List of listings

Appendix C - List of tables